An Investigation of the Modelling-Animation Relationship in Computer Graphics

by

Stephen C. Maddock

THESIS SUBMITTED TO THE FACULTY OF PURE SCIENCE
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE
DECEMBER 1998

© Stephen C. Maddock, 1998
An Investigation of the Modelling-Animation Relationship in Computer Graphics

Stephen C. Maddock

Abstract

In computer graphics over the last 20 years or so, the main body of research work has concentrated on the rendering process. Animation, particularly shape change animation, has been a relatively poor cousin. In recent years this has changed. Full-length computer animated films have appeared employing sophisticated animation techniques, and there has been a surge of interest in virtual reality and the use of three-dimensional animation techniques in computer games. Computer animation now commands as much research attention as the emphasis on rendering techniques did in previous years.

This thesis will offer a review of the modelling-animation relationship and the reasons for the general lack of success of computer animation (except for the case of rigid body animation) in the light of this relationship. A detailed examination of the important case of facial animation will be offered as an example of state-of-the-art shape animation. This is much studied and arguably the most difficult application of shape change animation, mainly because of our high expectations. We are acutely attuned to the subtlety of facial expression and thus the shortcomings of computer facial animation are perhaps more apparent that other shape changing applications. The thesis will also explore the failure of modelling techniques (particularly in animation) to represent surface detail at expected photorealistic quality.

The main thrust of the thesis will concentrate on shape change, using facial animation as an important example. The following results will be included as part of the thesis discussion: a localised deformation control method for quilts of bicubic Bezier patches, the combined use of free-form deformations [Sede86] and muscle-based control [Wate87] for facial animation, a technique for dealing with polygon model silhouette edges in two dimensions, a fast, hybrid technique for cloth animation in computer games and the use of texturing techniques in modelling detailed imperfections such as scratches and dents over time.
Acknowledgements

It was a chance encounter with Jim McGregor that led me into computer graphics as a career – my thanks go to him and to Alan Watt for allowing me to work with them on one of their early computer graphics books. This eventually led to my decision to do computer graphics research, and Alan Watt became my supervisor. Now, as my colleague and friend, I wish to thank him for all the support he has given to me in the years leading to the production of this thesis. His insights into many research areas have been invaluable and he has generously shared his time and ideas.

Over the years, there have been many people in the Department of Computer Science whom I have enjoyed fruitful research conversations with. I especially thank the members of the Computer Graphics Group during that time. Thanks also to my colleague, Colin Smythe, who, as Head of Department, supported my application for more time to complete this thesis. Thanks also to the friends, family and colleagues who offered support and urged me on. Thanks to my parents, who, throughout my whole life, have always supported and encouraged me in my educational endeavours.
Contents

1. Introduction ................................. 1
   1.1 Modelling and animation: an intimate relationship ......................... 2
   1.2 Overview of thesis .............................................................. 3
   1.3 Summary of contributions of thesis ....................................... 4

2. Modelling and Animation .................. 6
   2.1 Representation ................................................................. 7
   2.2 Modelling techniques ....................................................... 8
      2.2.1 Constructive approaches ............................................... 9
      2.2.1.1 ‘Hard’ primitives with blends .................................. 10
      2.2.1.2 Soft primitives ...................................................... 12
      2.2.1.3 Swept primitives .................................................. 16
      2.2.1.4 Layering .............................................................. 20
      2.2.1.5 Procedures and primitives ..................................... 21
      2.2.2 Refinement approaches .............................................. 23
      2.2.2.1 Direct surface manipulation .................................. 24
      2.2.2.2 Form features ..................................................... 33
      2.2.2.3 Parameterised surface alteration ............................... 36
      2.2.2.4 Space manipulation – generic deformations ............... 37
   2.3 Animation techniques ..................... 44
      2.3.1 Low-level techniques .................................................. 46
      2.3.2 Medium-level techniques .......................................... 49
      2.3.3 High-level techniques ................................................ 53
   2.4 Summary and discussion .................. 54

3. Facial Animation ............................ 58
   3.1 Impressions of a face ....................................................... 59
3.2 Facial form ................................................................................................................................. 60
  3.2.1 Producing a model of an existing face .................................................................................. 60
  3.2.2 Conformation and the creation of a new face ....................................................................... 63
  3.2.2.1 Blending ........................................................................................................................ 64
  3.2.2.2 Parameterisation .......................................................................................................... 65
  3.2.2.3 Construction from parts ............................................................................................... 68
  3.2.2.4 Deformation or surface refinement .............................................................................. 68
3.3 Facial expression .......................................................................................................................... 76
  3.3.1 Key-pose interpolation for producing expression ................................................................. 77
  3.3.2 Direct parameterisation for control of expression ................................................................. 80
  3.3.3 Pseudomuscle-based control of expression ....................................................................... 82
  3.3.4 Muscle-based control of expression .................................................................................. 84
3.4 Lip control - Visible speech ......................................................................................................... 95
3.5 Summary and discussion ............................................................................................................ 97

4. Detail in Modelling and Animation .......................................................................................... 100
  4.1 Polygons and their problems in interactive computer games ................................................. 102
    4.1.1 Level of detail .................................................................................................................. 105
    4.1.2 Polygons and silhouettes ............................................................................................... 108
    4.1.3 Non-planar primitives for computer games ..................................................................... 114
  4.2 Using polygons in shape change animation - a case study: cloth ........................................ 119
  4.3 Modelling detail to imitate imperfections .............................................................................. 126
  4.4 Summary and discussion ........................................................................................................ 134

5 Conclusions .................................................................................................................................. 136

Bibliography ................................................................................................................................... 140
# List of Figures

2.1 The relationship between the constructive and refinement approaches ................................................. 9  
2.2 A CSG tree reflecting the construction of a simple object from three primitives ................................. 11  
2.3 An object modelled using implicit primitives. The object is sliced in half to reveal intermediate isosurface layers ........................................................................................................... 13  
2.4 An object modelled using implicit primitives. Light grey primitives are additive and dark grey primitives are subtractive. (Courtesy of Agata Opalach) ................................................................. 13  
2.5 An object modelled using implicit primitives. The top pair of images show unwanted blending as the generators move. The bottom pair of images show unwanted separation as the generators move. (Courtesy of Agata Opalach) ................. 14  
2.6 Superquadrics ........................................................................................................................................ 16  
2.7 Using sine waves to modulate the sweep of a torus ............................................................................. 17  
2.8 A wine glass as a surface of revolution ............................................................................................... 17  
2.9 Sweeping a varying circular cross-section along a space curve ............................................................ 18  
2.10 Objects produced using a simple shape grammar .............................................................................. 22  
2.11 A truncated pyramid and a cuboid are stochastically distributed on the surface of a sphere to produce a more complex object (produced using a 3D Studio Max plug-in)........................................... 23  
2.12 Progressive movement of a control point for a bicubic Bezier patch .................................................. 24  
2.13 The Utah teapot: (a) lines of constant u and v; (b) exploded wireframe of control points; (c) wireframe of the patch edges .................................................................................................................. 25  
2.14 The Utah teapot drawn as a wireframe by converting the patches to polygons using uniform subdivision (32, 128, 512 and 2048 polygons respectively) ................................................................................. 26  
2.15 Four adjoining bicubic Bezier patches. The group of nine control points centred on the meeting point of the four patches can be moved as a unit whilst maintaining continuity across the joins ........................................................................................................... 29  
2.16 Local surface deformation of a cube defined using Bezier patches .................................................... 30  
2.17 Local surface deformation of a B-spline surface (used with permission from [Watt92]) .................. 31  
2.18 Gindy’s form feature taxonomy (after [Parr93]) .................................................................................. 34  
2.19 Classification of pre-defined design features (after [Viei96]) .............................................................. 35  
2.20 Parameterised object alteration of a teapot: (a) global scale; (b) the lid is treated as a separate object .................................................................................................................................................. 37  
2.21 A shell-like object formed using twisting and tapering operations ..................................................... 38  
2.22 Barr’s transformations (see [Barr84]) applied to a teapot and a cube .............................................. 39  
2.23 A rendered version of the tapered teapot in Figure 2.22 .................................................................... 40  
2.24 Unwanted undulations on the surface of the cube caused by manipulation units .......................... 40
The problem of using manipulation units illustrated using curves ........................................40
Using a widget to control simple axial deformations (after [Watt98]) ........................................41
FFD example: A single hyperpatch is used to deform a sphere (used with permission from [Watt92]) ..........................................................................................................................42
Animation techniques ..........................................................................................................45
Inbetweening using multisection Bezier curves ......................................................................47
Motion capture data for a short dance sequence .......................................................................48
Increasing the amount of twist and bend applied to a teapot modelled as parametric patches .................................................................................................................................49
Using travelling sinusoidal waves for water: (a) adding parallel waves; (b) radial waves ........51
Hierarchy for a human figure ..................................................................................................51
A physically-based model of a human leg driven by muscle contraction values.
(Courtesy of Lee Cooper.) ......................................................................................................52
Animation as a hierarchy of knowledge control (after [Calv94]) ............................................55
A simple diagrammatic notation for relating representation, modelling and animation. ......56
Comparing particle systems with flocking ................................................................................57
Documenting (a) the dancing figure with physically-based Disney effects from
[Opal95b] and (b) the dancing spoon from [Watt92] ..............................................................57

3.1 The vertex topology of (a) Waters’s model, and (b) Parke’s model (see [Park96] for
a description of both models) .................................................................................................62
3.2 The vertex topology in FaceWorks stretched over three different faces .............................62
3.3 Vertex positions are used to blend two faces .....................................................................64
3.4 A variety of faces produced using Parke’s conformation parameters ................................65
3.5 Interpolation between two faces: From left to right, faces 1, 3, 5, 7 and 9 are given
and faces 2, 4, 6 and 8 are interpolated ................................................................................66
3.6 Extrapolating a blended face .............................................................................................67
3.7 Forensic facial reconstruction: (a) laser-scanned data for a skull (40x40 polygon
mesh VRML 2.0 file); (b) reconstructed ‘emaciated’ face; (c) reconstructed ‘obese’
face. (Data courtesy of Martin Evison, Dept. of Forensic Pathology, Univ.
Sheffield.) .................................................................................................................................70
3.8 A global deformation used to stretch the face to one side: (a) using a simple linear
deformation; (b) using a tricubic Bezier hyperpatch ............................................................72
3.9 A local deformation to produce an elongated nose: (a) using a simple linear
deformation; (b) using a tricubic Bezier hyperpatch ............................................................72
3.10 Two further noses produced by moving the control points of a tricubic Bezier
hyperpatch: (a) undeformed nose; (b) a turned-up nose; (c) a hooked nose .........................73
3.11 Pinching the face inwards towards the chin ....................................................................73
3.12 The problem of FFD overlap ..........................................................................................74
3.13 FFDs applied to Parke’s model .........................................................................................76
3.14 Interpolation of vertex positions between the key poses of fear and disgust ................78
3.15 Bilinear interpolation of vertex positions between four key poses .................................79
3.16 Extrapolation of vertex positions for a smile........................................................................79
3.17 A variety of facial expressions produced by varying Parke's expression parameters ..........80
3.18 Kalra's RFFD actions (after [Kalr93])..................................................................................83
3.19 The geometry of a linear muscle .........................................................................................85
3.20 Changing the linear muscle contraction factor: (a) the muscle overlaid on the mesh contracts and distorts the mesh; (b) increasing the contraction; (c) two muscles; (d) two overlapping muscles ..................................................................................86
3.21 (a) The geometry of a sphincter muscle and (b) its action on a simple mesh ...............87
3.22 (a) The geometry of a sheet muscle and (b) its behaviour ..............................................87
3.23 The six primary facial expressions of happiness, anger, surprise, sadness, fear, and disgust/contempt .................................................................89
3.24 Vertices and muscles of the face .......................................................................................90
3.25 Surprise, incorporating jaw movement .............................................................................90
3.26 An FFD is used to stretch the side of the face and also move the end points of muscles, before the six primary expressions are created using the same muscle contraction values as in Figure 3.23 ..................................................91
3.27 An FFD is used to pinch in the chin and also move the end points of muscles, before the six primary expressions are created using the same muscle contraction values as in Figure 3.23 ..................................................92
3.28 Interpolating key pose vertex values versus muscle parameters ...................................94
3.29 Four example visemes from a table of mouth shapes based on observation of real lips (after [Wate93]) .................................................................95

4.1 Frame rate versus image quality versus cost (after [Helm94]) ..............................................102
4.2 Scenes from Actua Soccer 2 .................................................................................................104
4.3 Hoppe's edge collapse transformation (after [Hopp96]) ....................................................107
4.4 Using a simple metric for edge collapse. From top to bottom, the teapot mesh contains 1074, 488 and 292 polygons respectively ........................................108
4.5 Polygon counts for different rendered subdivisions of a teapot (a) 128 (b) 512 (c) 2048 (d) 8192 ......................................................................................109
4.6 The final image quality can be improved by using fewer polygons for internal areas and more on the silhouette edge ........................................................................109
4.7 Cutting corners (after [Phon75b]) .......................................................................................110
4.8 Smoothing the silhouette edge of a poor polygonal approximation to an elliptical shape such as, the upper arm of a character .........................................................................112
4.9 Fitting silhouette curves to a polygonal approximation of a teapot .................................113
4.10 A Bezier curve is fitted through silhouette vertices and used to calculate new vertex intensities for subdivided polygons ......................................................114
4.11 Tears produced by non-uniform subdivision of patches .................................................115
4.12 The character hierarchy with attached skirt ....................................................................121
4.13 The skirt model ...............................................................................................................122
4.14 The lower layer skirt vertices are calculated from the position of the legs ..................122
4.15 The skirt in the run cycle .................................................................................................123
4.16 The skirt in the dance sequence ......................................................................................124
4.17 Skirts in Actua Tennis: texture mapping is used to enhance a low geometric resolution polygon model .......................................................... 125
4.18 A scene from Actua Soccer 2 .................................................................. 126
4.19 A chewed pencil rendered using bump mapping ........................................ 128
4.20 A close-up of the chewed pencil rendered using bump mapping .................. 128
4.21 A mug and a chip created using CSG techniques ........................................ 129
4.22 A tumour, created using CSG techniques, applied to an eye for Waters’s face model (see chapter 3) .............................................................. 129
4.23a Stage 1 in the evolving desktop scene: no imperfections ............................... 130
4.23b Stage 2: slight imperfections – warped paper and one chewed pencil .......... 131
4.23c Stage 3: Mug stain on the desk and curling edges on the calendar .............. 131
4.23d Stage 4: More marks on the desk, coffee stains on the piece of paper where the mug was and one pencil is now very short and sharpened at both ends ...................... 132
4.23e Stage 5: The mug has been knocked over causing a spill on the table, the table has doodles, scratches and stains on it, and the ruler is broken .......................... 132
4.23f Overhead viewpoint for stage 5 .............................................................. 133
Chapter 1.

Introduction

The study of shape change and deformable objects has a long history: A sequence of cave paintings at Lascaux in France show a horse starting from rest and breaking into a run. More recently, D’Arcy Thompson has expounded on the difference between form and function for natural objects, including work on deforming one natural object into another [Thom61], and the Disney studios have famously elucidated a set of rules [Thom81] for producing expressive animation of deformable objects for the entertainment industry, which have been duly documented for the computer graphics world by Lasseter [Lass87]. Many of these ideas find their apotheosis in the recent Disney/Pixar film “Toy Story”, which also features much work on facial animation, an area in which modelling and animation are intimately linked and where our familiarity imposes strict judgement criteria. In recent years, three-dimensional deformable objects have also become commonplace in real-time interactive environments, with computer games featuring simulations of team-based sports, such as football, taxing available hardware to the limit and thus needing novel modelling solutions. This thesis will investigate the relationship between modelling and animation that is inherent in all these examples.

The thesis will be set in the context of two general areas of investigation. The first area will investigate modelling techniques in relation to animation control in three dimensions and will be explored in the context of human facial animation. Here, the design mantra of form follows function can be considered in relation to an animation context. The second area to be considered is that of shape change in relation to ‘detail’, where detail can be interpreted as real three-dimensional geometric detail or apparent detail created using, say, computer graphics texturing effects. This will include work in relation to computer games, where one of the main problems to be addressed is how to provide adequate visual realism in a real-time interactive environment with its inherent frame rate constraints. The problem of dealing with complexity is a significant aspect of each of these applications. For animation the complexity is in the control of multiple degrees of freedom, the modelling of surface detail deals with trade-offs in complexity and the real-time environment of computer games concerns the production of apparent complexity for least cost. Thus the thesis could be said to also address the issues of growth in complexity of solutions to the problem of creating realism effects in modelling and animation.
1.1 Modelling and animation: an intimate relationship

For deformable objects the interplay between modelling and animation is paramount. In deciding what animation capabilities are required, we must consider the capacity of the modelling technique since it may unduly affect the animation capabilities. For instance, polygons can be used to represent virtually any object type. Indeed they seem to have become a de facto base modelling primitive to which other complex modelling primitives may at some point be converted (probably because of the research effort expended on producing polygon-based rendering architectures). But whilst polygons are simple primitives, easily transformed and rendered, they require complex control methods to manipulate them when they are used to model deformable objects. An abstraction needs to be formed above polygon level, otherwise the number of points to be controlled would be inordinately large. Scripting every point on a polygon model of a human face would be infeasible. Instead an abstraction such as a pseudomuscle model [Wate87] can be used to control collections of points.

Whilst polygons can be used to model organic objects, large numbers are inevitably needed to capture the smoothness of the initial surface and any changes it may undergo. This is difficult to predict in advance and can mean using excessive numbers of polygons so that smoothness is retained whatever deformation occurs. This might indicate that a curved primitive such as a parametric patch would prove more suitable for modelling deformable objects since more detail could be represented with a single primitive. The problem then is maintaining continuity between patches and offering both local and global surface control. A good example of this is the animation of the baby’s face in the Pixar animation “Tin Toy”. The face is modelled using parametric patches and the subsequent animation is marred by “buckling” effects around the mouth area [Reev90].

Whichever modelling primitive is chosen, the main problem in computer animation is the number of degrees of freedom that have to be controlled. This is especially true when deformable objects are considered. Here the number of degrees of freedom can saturate the interface with parameters to be controlled. Consider the human face. Parameters must be supplied to control the motion of the eyes, the mouth and the tongue, as well as such things as the colour of the skin for individuality and for reactions such as blushing. It is inevitable with such complex models that there is a need to consider higher levels of control with mappings to the low-level control parameters. Even so, this still presents the problem of defining the appropriate mappings and a suitable amount of expressive control at the higher level.

One way of providing high-level control is to use physically-based modelling and behavioural models. The physically-based modelling ensures that a deformable object behaves accurately in its environment, for instance squashing into immovable surfaces, and does not undergo impossible changes. The laws of such a model can then be altered to produce exaggerated effects [Opal95b]. But physically-based models are compute-intensive and can become unstable, for instance mass-spring-damper systems can undergo catastrophic failure under extreme conditions [Coop98]. Also it can be difficult to assert the required control over an object if it is at the same time trying to obey in-built physical laws. The same is true of
CHAPTER 1. INTRODUCTION

behavioural models. Here, rules can be used to instil some form of personality into, say, a human model, so that it behaves accordingly in particular circumstances. For instance a human might nod his head in agreement, look in a particular direction when thinking or blush if embarrassed. But again it can be difficult to assert ‘required’ control.

1.2 Overview of thesis

Chapter 2 of the thesis will explore the relationship between modelling and animation in more detail, setting forth a taxonomy of modelling techniques and relating these to animation techniques. The issue of form and function will also be discussed. We will see that what creates a structure needs to be addressed so that control can be automated and not made more difficult through poor approximation to the functional aspects of a model. Chapter 2 will also present a new technique for local control of nets of bicubic Bezier patches and a new diagrammatic technique to show the relationships between modelling and animation techniques in applications.

Chapter 3 will build on the discussion of chapter 2 by investigating the area of computer facial animation [Park96]. There has been much work in the area of facial animation, from Parke’s work in the early seventies [Park72] through to more recent work incorporating physically-based modelling [Terz90] and behavioural models [Cass94]. Nonetheless, the expressive control of realistic 3D facial animation remains an elusive goal. Instead facial animation has achieved most success in two quite separate areas: model-based coding for video conferencing and stylistic facial animation for cartoon characters. The first area is outside the study of this thesis, although brief comments will be made on it in chapter 3 and in the conclusions. The second area can be briefly summarised using the output of one company – Pixar.

In Pixar’s early computer-generated short film ‘Red’s Dream’ (1987), a few simple facial expressions, e.g. raising of eyebrows and smiling, were implemented for a clown riding a unicycle. The facial animation is quite successful, probably because of the nature of the cartoon-like face and the simple set of expressions used. This was followed by a more elaborate system for the facial expressions of a baby in ‘Tin Toy’ (1988) (the first computer animated film ever to win an Academy Award). Here, the facial animation is clearly not successful and the baby has a somewhat gruesome look at times. The model was based on parametric patches, for which there were continuity problems across patch boundaries thus causing creasing in the rendering of the baby’s face [Reev90]. After this less than auspicious effort, Pixar went back to using toys for ‘Knickknack’ (1989). The facial animation was quite limited and the fact that the characters were made to look computer generated using shiny surfaces meant that the simple facial animation was acceptable and not judged harshly. No doubt lessons had been learnt. Imitating real human faces is difficult.

In 1995, Pixar produced ‘Toy Story’, a landmark full-length computer-generated film. Interestingly this again featured toys as the lead characters, for which the facial animation was effective. In contrast, the facial animation of the human beings in the film was not so good and was kept to a few short sequences. It seems that we are so acutely attuned to the nuances of the
human face that the closer the computer facial animation gets to real faces, the stricter the
judgement criteria become, whereas for cartoon characters we are prepared to suspend such
criteria in favour of something more simple, such as ‘is the character expressive?’ On the basis
of this we might argue that three-dimensional computer facial animation has been a failure for
photorealistic imagery. Despite over 25 years of research, the easy control of successful 3D
character animation is still a distant goal. There is no agreed way of deciding on a set of
parameters to control a relatively simple object. Indeed, this is compounded by the fact that
there appears to be little or no work on what is the best modelling technique to use for a
particular object in a particular application. In chapter 3, a technique for using free-form
deformations (FFDs) [Sede86] for facial conformation control will be presented. This will then
be combined with a pseudomuscle model technique [Wate87] for expression control to show
how a hybrid technique can be used to unite expression and conformation in an intuitive and
effective way.

Another aspect of the modelling-animation relationship is the concept of the detail in a model.
For static models, built just for rendering purposes, intense geometric and graphical detail can
be used to create photorealistic quality imagery. For animation applications, detail needs to be
dealt with more carefully, especially with regards interactive computer animation. Chapter 4
will address two aspects of detail: the use of polygons in interactive computer animation and
modelling to mimic surface imperfection. The particular interactive computer animation
application will be computer games, where polygons are the de facto modelling primitive and
realism is constrained by frame generation time, which thus limits the number of polygons that
can be used to represent objects. A technique for smoothing silhouette edges will be presented
along with a case study of the animation of shape change for cloth. For surface imperfections,
such as stains, dents and scratches, very small scale shape change over time will be considered
in a discussion of apparent realism, or how 2D techniques are used to mimic 3D.

1.3 Summary of contributions of thesis

As part of a general investigation of the modelling-animation relationship, the thesis presents a
comprehensive taxonomy of modelling techniques for animation. Together with a number of
standard modelling techniques, which were implemented to facilitate the production of this
taxonomy, a new technique that uses manipulation units for local control of nets of bicubic
Bezier patches is introduced. The categorisation of techniques is documented in a diagrammatic
technique that can be used to make plain comparisons between different topic areas in
modelling and animation.

A hybrid control technique for facial animation is presented, which uses FFDs for local and
global conformation of the face, and simulated muscles for expression control. The hybrid
technique permits the deformation of both the facial model and the simulated muscles used in
the expression control technique. Thus, one face or parts of one face can be easily deformed
and the muscle parameter values that were used on the undeformed face can be used on the
deformed face to give similar results.
CHAPTER 1. INTRODUCTION

The thesis also investigates the use of detail in modelling and animation. In considering the uses and problems of using polygons to represent modelling detail in interactive computer animation, techniques for enhancing visual appearance for least cost are covered. A new technique for smoothing piecewise silhouette edges in two dimensions is described, and a case study of the use of polygons for shape change in computer games is given by implementing a hybrid cloth model. The work on polygons includes work done whilst on a six month sabbatical at Gremlin Interactive, a computer games company based in Sheffield. Finally some work on surface imperfections, such as stains, scratches and dents, together with changes in surface imperfection over time, is included as an example of apparent realism in modelling detail.
Chapter 2.

Modelling and Animation

Introduction

Modelling and animation are often considered as separate fields. For instance in the computer-aided design world the modelling and subsequent analysis of the modelled object is paramount. Animation is often considered as a separate next step when a range of animation techniques can be considered for effecting the movement or deformation of the object. A similar scenario can be construed in the animation world. A technique is developed for controlling the animation of an object, for example the use of parametric curves, and this is then considered in relation to different modelling representations. Another example might be the use of free-form deformations [Sede86] and their subsequent animation control [Coqu91] when applied to a particular modelling representation. However, this separation between modelling and animation breaks down when we consider organic objects or natural phenomena. Here modelling and animation are intimately linked. A modelling representation may determine the animation potential and the actions required of an object may determine how the model must be constructed. Form and function must be considered together. This is especially true for living organisms, as "Structure without function is a corpse and function without structure is a ghost" ([Wain88] quoting Vogel and Wainwright, 1969). This link between form and function will be explored in relation to facial animation in the next chapter. In this chapter we will consider it in a more general study of the mixed bag of modelling and animation techniques produced by the computer graphics community.

The chapter will investigate the transition from treating an object as a static design object to a moving (natural) deformable object. Gomes provides a formal system [Gome93] in which to study modelling in computer graphics. This can be used to help address the confusion of terminology that often affects the computer graphics community. With this in mind, a range of modelling techniques will be examined such as sweeping and free-form editing. The categorisation of animation techniques tends to focus on the mechanics of particular techniques with the usual categorisation considering them on a simple scale of high to low-level [Watt93]. A categorisation by the amount of input required by an animator and the amount of control this imparts will be presented and considered in relation to a range of animation techniques.
CHAPTER 2. MODELLING AND ANIMATION

2.1 Representation

Modelling and representation are often used as an umbrella phrase when considering the creation of a 3D computer graphics model or the data structure used for the model or the manipulation of the model; terminology is not the strong point of the computer graphics community. The data structure used for a model is not the same as the representation of the model. The modelling representation is not the same as the modelling technique. Gomes presents a formal framework [Gome93] that can be used to address this confusion of terms. It is based on Requicha's universes paradigm [Requ80] which provides a mathematical foundation for geometric and solid modelling. In that field the motivation is the need for robust implementations that permit analysis and simulations of objects, for instance computing physical attributes such as mass and forces or computing geometric attributes such as area and volume. For general modelling in computer graphics, Gomes lists the following universes:

- physical;
- mathematical;
- representation;
- implementation.

The physical universe is the real world of tangible objects. The mathematical universe is the abstract description of objects from the physical universe. The representation universe is concrete, offering finite descriptions of mathematical models. And the implementation universe is the actual computer implementation that separates the finite representation from the particularities of data structures and languages. In the mathematical universe manifolds, space decompositions and piecewise algebraic schemes hold sway. In the representation universe the use of pure primitives, the constructive use of primitives as in constructive solid geometry and the decomposition approach using space-based schemes such as B-rep or object-spaced schemes involving spatial occupancy are considered. Gomes regards exact topology and good approximation to the object geometry as important in the representation universe. (The different representations will be presented in more detail in the relevant subsections of the following section on modelling techniques.) In the implementation universe spatial data structures such as graphs, trees and lists are employed, for example a binary tree could be used for a constructive representation or a winged-edge structure could be used for a polygon mesh. Samet [Same89] provides a detailed investigation of such spatial data structures.

The use of the universes paradigm allows the analysis of uniqueness and of completeness of abstractions and representations to be set in a more formal context. For instance the mathematical abstraction of a curve could be represented as a sequence of line segments but this is not unique as any sequence of line segments could serve. This raises the issue of what is the best representation. Whilst the fact that there are so many representation schemes means great flexibility it is also, as the curve example indicates, the origin of many problems. This is compounded by the fact that hybrid schemes maintain multiple representations simultaneously in order to use the advantages of each representation. This raises issues such as how to convert between representations and whether or not there is a universal representation.
CHAPTER 2. MODELLING AND ANIMATION

The type of representation used affects the process of deforming an object. Consider a polygonal representation – the most common representation in computer graphics – where ‘enough’ polygons are used to ensure a smooth appearance for a surface. Depending on the severity of a deformation that stretches and twists the surface, the polygonal nature of the surface may be revealed. This would not happen with a parametric representation, which would retain its smooth appearance under severe distortions. The comparison between polygons and patches suggests a way of structuring the discussion of modelling techniques by initially distinguishing between discrete and continuous representations. However, a split between discrete and continuous description is perhaps more suited to a mathematical categorisation of form [Lor84].

Snyder suggests the following criteria for evaluating a shape representation [Sny92]:

- Ease of specification – how easy is it for a designer to enter shapes into the computer?
- Renderability – how quickly and realistically can images of the shapes be generated?
- Analyzability – what analytical operations are allowed on shapes? How fast and robust are these operations?

Snyder adapts Requicha’s work [Requ80] to further break down ease of specification into naturalness, compactness, completeness, controllability, editability, validity, accuracy and closure. Borrowing from this and incorporating ideas from Badler and Bajcsy [Bad87], from Cobb [Cobb84], from Watt [Wat93] and from Watt and Policarp [Wat98], we can consider the following categories for evaluating representations for deformable models:

- Naturalness – does the representation correspond to the way designers think about shape? What is the nature of the primitive elements?
- Compactness, Cost and Complexity – how much information is required to specify the shape? There is inevitably a trade-off between compactness and representation complexity here. There are also issues of representation accuracy – is the representation only an approximation of the physical object or the designer’s intention?
- Editability – can the designer modify the shapes easily? What operations on the representation are natural?
- Controllability – can the designer predict what shape will result from a given input?

These will be considered in relation to the modelling techniques discussed in the next section.

2.2 Modelling techniques

An oft-used categorisation of modelling techniques is based on a surface versus solid distinction, perhaps because of the historically independent evolution of free-form surface and solid modelling systems [Requ80]. Nowadays, surface and solid modelling functionality have been combined in systems (see early work in [Van84, Mil86, Brun90]) and modelling techniques that are dissociated from the underlying geometric model (e.g. [Barr84, Sed86, Coq90, Bor91, Chan94, Laza94, MacC96]) have become commonplace. A different categorisation is needed, especially if animation rather than object representation is the goal.
In considering geometric aspects of form, Lord and Wilson [Lord84] make a distinction between shape and form by considering internal structure. Form implies that internal structure is under consideration whereas shape is merely the external appearance of an object. Since we are interested in techniques to deform a representation, a categorisation that considers the structural aspects of an object seems appropriate. This relates to Gomes representation universe [Gome93] where a distinction is made between constructive and decomposition approaches to representation. In fact, in considering modelling techniques, Gomes suggests that many are closely connected to the scheme used to represent the constructed object. This suggests that a categorisation can be made by whether the modelling technique makes constructive use of primitives or whether it tends more to a decomposition approach starting from a whole and changing or refining parts of this. This is not strictly the usual meaning for a decomposition approach to modelling where, say, a spatial subdivision scheme would discretise the space of an object. Instead, the term is used more loosely. Figure 2.1 illustrates the relationship between the construction and refinement approaches. Primitives are combined in a constructive approach to produce a ‘whole’ model. A whole model is refined to produce another whole model which itself could be a primitive.

![Constructive and Refinement Diagram](image)

**Figure 2.1: The relationship between the constructive and refinement approaches**

### 2.2.1 Constructive approaches

In general, constructive approaches involve building an object from pieces or primitives or using layers. The constructive approach means that the design of an object is based on initially being able to break an imagined object into a number of pieces. High-level primitives can then be used to create a rough model of the object and a global vision of the object is then retained as the pieces are subsequently refined or replaced with other, possibly multiple, pieces.

A similar approach is found in certain computer vision work where high-level primitives can be used to provide initial estimates to objects in a scene, thus facilitating the construction of hypotheses of what the scene represents. Badler and Bajcsy [Badl78] explain this by stating that the “basic tendency in human vision is to prefer regularity, for both its transformation invariance and its data reduction capabilities”. Thus simple objects are substituted in describing more complex objects, e.g. a cylinder for an elongated object. Many vision researchers have built on this idea, choosing superquadrics as their primitives (e.g. [Pent86, Huxo89, Terz91a]). Pentland [Pent86] likens them to “lump[s] of clay” which correspond to the notion of “part of
an object”. Individual superquadric objects are used as prototypes that are then deformed to generate new forms and combined using Boolean operators to match objects in the scene under study. The formative history of an object is also recorded which, as Huxor and Elliott [Huxo89] state, is a particularly good approach in aiding conceptual design. Terzopoulos and Metaxas [Terz91a] make similar use of deformable superquadrics to “bridge the gap between shape reconstruction and shape recognition”. First gross shape features are captured from visual data (recognition) – this is similar to using simple volumetric primitives such as spheres to segment a shape into a small set of parameters – and then local deformation parameters reconstruct complex detail (reconstruction) – this is similar to using spline models that can deform locally subject to geometric discontinuity constraints. Instead of superquadrics, Muraki [Mura91] advocates the use of Blinn’s “blobby models”. However Muraki’s work can be regarded as a decomposition or refinement approach. He starts from a single ‘blob’ primitive and a better shape approximation is evolved using an energy minimisation process. This avoids the segmentation problem inherent in trying to initially decide on the gross details of an object. This can also be considered a disadvantage in that the ‘right’ gross details of the object do not necessarily emerge for subsequent control.

In essence, the constructive approach is about discovering “the logic of a shape”, a phrase used by Kajiya in the foreword to Snyder’s book on generative modelling [Snyd92]. He goes on to say that “because of this, shapes are no longer inscrutable lumps, but a series of puzzles”. Since these puzzles involve identifying and piecing together parts, we structure the discussion of constructive approaches by considering the following types of primitives, although there is some overlap amongst them:

- hard primitives with blends
- soft primitives
- swept primitives
- layering
- procedures and primitives

The classic technique of constructive solid geometry is covered by hard primitives. Soft primitives focuses on soft or blobby objects [Blin82, Nish85, Wyvi86] although the use of superquadrics as deformed primitives [Terz91a] are also briefly commented on. Swept primitives covers techniques that involve pinching and welding pieces together [Gasc89] or combining them with other operators [Snyd92]. Layered models [Chad89, Over97] are regarded as a constructive approach although they will be looked at in more detail in the discussion on animation. Finally, procedural primitives are discussed as a separate section although the other sections incorporate primitives that can themselves be parameterised and treated as procedural primitives.

2.2.1.1 ‘Hard’ primitives with blends

Constructive solid geometry (CSG) is the classic constructive approach to modelling solid objects. A number of simple ‘hard’ geometric primitives are provided – “the traditional CSG
primitives are block, sphere, cylinder, cone and torus" [Hoff96] – and these are combined using the Boolean set operators union, intersection and subtraction. Rigid body transformations are used to position primitives before the set operators are applied. The resulting object is stored as a tree, with the leaves being the primitives and the nodes being the operators or transformations. Figure 2.2 shows such a tree. The power of the process is apparent with the possibility of combining more complex intermediate objects to create ever more complex objects.

The tree gives a record of the user's interactions to produce the final object and thus it can be said that the modelling technique is the representation. The direct partnership between the modelling technique and the representation is perfectly suited to the application area that CSG is usually applied in, that of creating machine parts. Indeed it is easy to see the natural relationship between the machining processes used on simple blocks of metal and the Boolean operators. For instance in Figure 2.2, two simple blocks of metal are combined and a hole is then 'drilled' in one of them by subtracting a cylinder. Figure 2.2 can also be used to illustrate one of the problems of the technique in that the intermediate two-box assembly could have been produced by subtracting a small cube from a larger cube or the cylinder could have been subtracted from one box before producing the two-box assembly. The representation is not unique and in practice an object can be produced in a number of ways. This could have knock on effects for subsequent animation of parts of an object in that the 'main' parts of the object are not unique – what may be a satisfactory CSG tree for machining an object may be less satisfactory for animation purposes. Also for assemblies of parts, relationships must be expressed between mating parts. Such relationships then have to be maintained which reduces controllability.

![Figure 2.2: A CSG tree reflecting the construction of a simple object from three primitives](image)

CHAPTER 2. MODELLING AND ANIMATION

11
The joins between primitives can be smoothed by using a range of operators. This might be
done for manufacturing, aesthetic or structural reasons. For instance a fillet between two parts
can be produced by the sweep of a ball of appropriate radius rolling along the join [Cobb84].
However such modifications to an object’s surface can affect the mathematical integrity of the
CSG approach. As with any local modification of one of the primitives, the local modification
would have to be maintained in any subsequent Boolean operations on the whole object. Thus
the standard CSG modeller is less suited for applications like character animation or the
simulation of biological objects. Its very strength for precise machine part modelling becomes
its undoing when complex deformable models need to be deformed.

It is not necessarily the constructive approach that is the problem, more the need to maintain
mathematical integrity in line with the specialised application area of standard CSG modellers.
As we have noted, the natural application area of CSG is the design of machine parts or man-
made objects. And, as Badler and Bajcsy [Badl78] state: “the limitations on modelling man-
made objects are primarily physical (storage and processing time)”. For natural objects, such as
the human body, skull, brain, heart and blood vessels, the limits are “conceptual: what is a
reasonable representation of significant shape?”. The next section looks at a modelling
technique that captures that ‘essence of shape’ for natural objects.

2.2.1.2 Soft primitives

‘Blobby’ objects or soft objects, introduced to the computer graphics community by Blinn
[Blin82], Nishimura et al [Nish85] and Wyvill et al [Wyvi86], are the most common form of
implicit surface used in computer graphics modelling. Implicitly defined components (the
‘blobs’, so called because spherical components are the norm) are used as primitives which are
combined by summing the field effects of the equations defining the primitives. Thus, each
primitive or point-based field generator has an area of influence, usually a sphere, and a
‘potential’ function returns a scalar value for any point P within the sphere. An example
potential function is:

\[
 f(P) = \left(1 - \frac{r^2}{R^2}\right)^2 \quad \text{for } r \leq R
\]

where \( r \) is the distance of a point \( P \) from the primitive ‘centre’ and \( R \) is the radius of influence.
A scalar field \( F(P) \), at a particular value of \( r \), is produced from the combined effect of the
potential functions for each point-based generator. The simplest combination is addition. So for
a point \( P \), each contributing potential function is evaluated and summed. This results in an
isosurface or implicit surface which serves as the object representation. Figures 2.3 and 2.4
illustrate the process. In Figure 2.3, the implicit surface is cut open to reveal five point
primitives and isosurfaces at different values of \( r \). In Figure 2.4, the primitives or point
generators are shown on the left and the resulting implicit surface is shown on the right. The
radius of each sphere is proportional to the radius of influence and spheres are coded to show
whether their field functions should be summed (light grey) or subtracted (dark grey).
Figure 2.3: An object modelled using implicit primitives. The object is sliced in half to reveal intermediate isosurface layers.

Figure 2.4: An object modelled using implicit primitives. Light grey primitives are additive and dark grey primitives are subtractive. (Courtesy of Agata Opalch.)
For an organic or ‘clay-like’ object such as the one in figure 2.4 this representation seems quite natural. A few individual primitives initially capture the essence of the model which is then refined by adding more primitives – 64 primitives are used in the final image. It would be difficult to capture the fluid appearance with as few primitives using another modelling technique. The shape is easily editable at the detailed level with new primitives having a localised effect on the final shape. A simple interface can be defined which varies the radius of influence of a primitive and positions it relative to the others in the evolving model. Depending on the relative size of a new primitive to the whole object, only local adjacency relationships need to be examined in updating the evolving model. Opalach and Maddock [Opal95a] make use of this idea in an animation system where a three-dimensional grid discretises space and keeps track of primitives and their area of influence in lists indexed by the grid. The same secondary data structure could be used in an editing system for fast conversion to a polygon model.

There are a number of well-documented problems with implicit surfaces, namely unwanted blending and separation of primitives as they are moved [Opal93]. Figure 2.5 illustrates these problems. Also, unsatisfactory bulging effects can occur where implicit primitives blend at branching points [Bloo95]. The usual solution to unwanted blending and separation of primitives is to add some form of extra ‘support’ to the model. To prevent separation of
primitives, constraints can be defined which limit the movement of one primitive with respect to others. This is particularly suitable for modelling homogenous fluid-like behaviour. Terzopoulos et al use physically-based constraints between primitives to simulate heating and melting deformable models [Terz89]. Gascuel [Gasc93] and Desbrun and Gascuel [Desb94] build on this idea by adding collision detection and approximate volume preservation in modelling inelastic objects using implicit surfaces. Similar constraints can be defined in a modeller, but this then impacts on controllability with the effect of an edit being less predictable.

The use of a skeleton or general graph can be used to prevent unwanted blending [Opal93]. For instance, graph link and joint properties can be defined such that primitives on a link and a joint can blend, whilst primitives on adjoining links cannot. But this limits the very fluidity of the modelling technique unless the graph can be dynamically altered. The skeleton approach can be quite successful for certain types of model though. Fluid-like effects can be added to an articulated skeleton by attaching primitives as the ‘flesh’ around the skeleton. Choreographing the skeleton and simulating physically-based movement of the primitives results in cartoon-like effects such as squashing and stretching [Opal95b].

An alternative to using a secondary data structure to group a collection of simple primitives is to use fewer but more complex primitives. In computer vision, as mentioned above, superquadrics (e.g. [Pent86]) and hyperquadrics [Hans88] are used. These attempt to capture the essence of physical objects as a collection of modifiable primitives composed using a shape grammar. Figure 2.6 shows a set of superquadrics formed by varying the parameters e and n between 0.05 and 4.0 in the following implicit form of the superquadric formulation:

\[
f(r) = f(x, y, z) = (x^{2e} + y^{2e})^{1/n} + z^{2e} - 1
\]

Perhaps more familiar in the computer graphics community are Bloomenthal and Shoemake’s convolution surfaces [Bloo91]. With convolution surfaces, instead of point-based primitives, arbitrary curves and polygons are used as generators of potential fields. For certain objects, these can make better ‘skeletons’ than point-based primitives. For example, to approximate a flat surface, point generators must be closely packed to avoid bumps. For a convolution surface, a polygon can be used as the field generator. Convolution surfaces meet the criteria of being easily editable and edits are predictable with the resulting surface being suggested by the shape of the skeleton. Smooth surfaces are also guaranteed. However, the convolution technique is computationally expensive [Over97]. In a conceptually similar approach Decaudin and Gagalowicz [Deca94] use a fusion technique which smoothly blends interpenetrating shapes, where the shapes can be polygonally defined. A constraint is that the fusion shape has the same volume as the original shapes. Their technique allows hard edged shapes to be blended with smoother objects. For implicit surfaces, hard-edged shapes can be incorporated by adding implicit primitives into the standard CSG approach. The advantages of the two approaches are thus combined for creating models. Whether simple or more complex primitives are used for detailed models, for example a human face that can subsequently be animated, the primitives must inevitably be structured in some way for subsequent controllability.
2.2.1.3 Swept primitives

The use of sweeping as a representation has a long history, mainly in Computer-Aided Design (CAD) systems (see for example [Coll74, Goss76, Faux78]). The general idea is to define a cross-section in two dimensions (the generator) and sweep this along a curve defined in three dimensions. An example would be a torus that could be constructed by sweeping a circular cross-section around a circle. Figure 2.7 shows examples of this, with a sine wave being used to modulate either the cross-section or the sweep circle. A simple example, alternatively called an
extrusion, is to sweep a cross-section along a straight line, in effect adding a depth component – three-dimensional fonts can be easily created using this approach. Extending this by using multiple cross-sections or contours is known as ‘lofting’ (see, for example, [Wu77]). A surface or solid of revolution is a special case of the sweeping approach where a curve is rotated about a central ‘axis of revolution’. As an example, Blinn uses surfaces of revolution to construct the body and lid of a teapot [Blin87]. Figure 2.8 shows the results of sweeping a curve to produce a wine glass.

![Figure 2.7: Using sine waves to modulate the sweep of a torus](image)

![Figure 2.8: A wine glass as a surface of revolution](image)

As indicated in the previous paragraph, the sweeping approach can be generalised by allowing the cross-section to vary along the length of the sweep curve. These are sometimes referred to as ducted solids, generalised cylinders or tubular surfaces. The cross-section can be varied by a profile curve or a finite set of cross-sections can be interpolated or the two can be combined [Coqu87]. Figure 2.9 shows an example of a variable circular cross-section swept along a parametric curve. The circular cross-sections are instanced along the curve and are oriented using Frenet frames (see [Bloo90] for a discussion of such reference frames). Bezier curves are used for smooth interpolation between cross-sections. Polygons or parametric patches can be easily generated as a representation for visualisation purposes (see [Goss76, Coqu87, Watt92] for examples).
CHAPTER 2. MODELLING AND ANIMATION

![Figure 2.9: Sweeping a varying circular cross-section along a space curve](image)

Sweeping can be generally defined as [Sour96]:

$$ S = \bigcup_{t \in [t_1, t_2]} G(t) $$

where $S$ is the swept solid, $G(t)$ is a moving solid or generator, $\cup$ is union operation of all instances of $G(t)$ for sweep parameter $t$ in the interval $[t_1, t_2]$. Note that this formulation includes solid objects as generators. This has important application in many areas, e.g. NC machining verification, motion planning of manipulators in robotics to avoid collisions, temporal anti-aliasing in computer animation. Pickover [Pick90] uses spheres instanced along space curves to produce organic computer art. Sourin and Pasko [Sour96] concentrate on the problems of incorporating sweeping into CSG modellers, one problem of which is defining set operations on sweeps.

As Snyder [Snyd92] says “One of the advantages of sweeps is their naturalness, compactness, and controllability in representing a large class of man-made objects.” Two sweeps with varying cross-sections, together with instancing, can easily describe a wooden stool, and two surfaces of revolution in conjunction with two sweeps can generate a teapot [Blin87]. The description of the relevant 2D generating curves is compact and easily editable and objects generated by surfaces of revolution are easily machined using a lathe. However, for a general sweep along a three-dimensional curve, problems can occur. For instance, tight curves can cause the swept object to intersect itself. Also translation in the plane of the generating curve can create a degenerate object.

The roots of Snyder’s ‘generative modelling’ approach [Snyd92] lie in sweeps and in considering the logic of shape or how to construct an object from a set of swept pieces. In his work sweeps are specified procedurally and combined using a shape description language (in a similar way to shade trees [Cook84]). Primitive operators are combined to produce high-level shapes and shape-building operators as in a CSG approach. Snyder quotes an “airplane wing” as an example: an airfoil cross section is translated from the root to the tip of the wing and, at the same time, its thickness is modified, it is twisted, swept back, and translated vertically according to other curves. The technique’s roots in sweeping shine out. What Snyder manages to do though is add a controlling system to the sweeping approach so that models can be specified in a general, powerful and controllable way. Related to Snyder’s work, Gascuel
CHAPTER 2. MODELLING AND ANIMATION

[Gasc89] uses sweeping to produce primitive parts and then uses combination operators to produce more complex objects. Her work concentrates on smoothly joining pieces defined using splines by using welding and pinching operators.

A modelling concept that sweeping promotes but which receives little explicit attention, usually being taken for granted or mentioned in passing, is the notion of symmetry. Many objects can be described by their properties of symmetry. Indeed surfaces of revolution explicitly require an axis of symmetry. Tanaka et al [Tana89] make use of “generalized symmetry” to produce swept objects from 2D line drawings. They define generalized symmetry as “a property of paired three-dimensional smooth curves, where their symmetry axis is also a three-dimensional smooth curve. The pair of three-dimensional curves twist around their generalized symmetry axis”. Extra information is specified – contours, auxiliary vectors and their inclination angles – so that the projection of the drawing into 3D can be obtained. This representation is compact and the 2D interface is intuitive for objects with a simple varying cross-section along a sweep axis, e.g. the long stem of a pipe or the seat of a moulded chair or the arching leaves of a plant. Tanaka et al note that their method only produces an approximation of the line drawing, so the method could not be used to produce accurate engineering models. The technique does however suggest that symmetry as a modelling technique could be made more explicit within a modelling system (as will be illustrated in section 2.2.1.5). The work also suggests that 2D sketches might be used as a successful technique in building up models of 3D objects, something that has received attention recently [ZeLe96].

Besides man-made objects, sweeps are suitable for modelling many natural objects. As Badler and Bacjsy [Badl78] state “many biological structures have axes and are acceptably modelled by small numbers of generalised cylinders”. Wainwright [Wain88] also argues that many plants and animals have a cylindrical axis and “a cylindrical shape of the body, roots, branches, appendages and internal conduits”. Bloomenthal demonstrates this in modelling a Maple tree [Bloo85]. It is apparent that sweeps satisfy the naturalness criteria for both man-made objects and organic objects. They also produce a compact description at low cost. For an articulated human body, Badler and Bajcsy state that 80 cylinders would be satisfactory (although 60 of these would be just for fingers and toes). Shapes can be easily edited – editing of cross-sectional curves occurs in 2D – and the impacts of edits can be easily predicted. In contrast, Pentland [Pent86] argues that the need for a description of the sweeping function makes the use of generalised cylinders “neither succinct nor intuitively attractive” in comparison to using deformed superquadrics as the building pieces.

As indicated in the previous paragraphs, much of the interface work for sweeps is done in 2D. Perceptually the user must make a link between the 2D descriptions and the final 3D object. A collaboration between Brown graphics group and the university of Utah has been working on the use of 3D widgets for interactive control of three-dimensional deformations [Conn92, Sni892]. Generalised sweeps are part of this work. In effect the 2D sweep description, e.g. the cross-sections or the profile curve, is displayed in conjunction with the 3D object using transparency. The 2D descriptions are parameterised and attached handles can be interactively
manipulated in 3D to alter the swept object. Thus the parameterisation is explicitly exposed and permits intuitive exploration of the family of possible swept objects.

Questions still remain over the process of joining pieces together or surface closure. Although Gascuel’s work can be used to produce smooth joins, it is not clear how this is affected when the pieces either side of a join are articulated. Interestingly this is a current problem in the computer games industry where it is known as ‘skinning’. A skeleton is driven by motion capture data and a surface skin must be fitted. This can be done by sweeping individual limbs and using a Z-buffer to composite the parts, but if smooth joins are required between limbs a skinning technique must be used. Sets of points on the separate limbs can be joined in a plug-and-socket approach, although this doesn’t produce a smooth join, or a continuous skin must be fitted over the entire skeleton. In either case problems occur when limbs bend with respect to each other. For instance, the cross section can distort unpredictably – a flattening cross section can produce an effect similar to early cartoon animation in which the limbs appear to be made out of rubber hose.

2.2.1.4 Layering

A layered model can be described as a hierarchy with a simple ‘skeleton’ representation or structure as a starting point, on top of which more detailed levels or layers are built. Such layered models have become commonplace in computer animation, especially with regards to articulated structures (e.g. [Chad89, Terz91h, Opal95b, Turn95, Gasc97]). Articulated structures are naturally built using a layered model. A skeleton is used to choreograph gross movement and different layers of tissue can be used to give other effects. A distinction is usually drawn between geometric models of flesh and physically-based ones. Whilst geometric models can be very expressive, they can require too much user intervention to achieve realistic-looking effects. Physically-based models achieve more realistic behaviour, but are compute intensive. They are also difficult to control in order to give a desired behaviour. We will only consider a few examples of layering here, as the technique will be discussed more in relation to animation later. We are interested in layering as an approach to dealing with complexity.

We have already mentioned that for implicit surfaces a graph or rigid internal skeleton is used to structure the primitives for subsequent animation control. This layered model is used successfully to structure a range of implicit objects in the animation piece “Simply Implicit” [Fur93, Gasc97] and to structure simple dancing characters in Opalach’s work [Opal96]. This abstraction or, in essence, hierarchical or constructive model, provides a natural control method for a complex object. The use of a layered model helps the designer or animator focus on gross movement of character parts before detailed layered movement. This is a natural way to think of a complex object. Whilst a layered or hierarchical model increases storage cost, this is more than offset by offering a controllable description of a complex object. Given inputs result in predictable changes but this control is bound to the internal structure, whereas in the section on decomposition approaches we will see that an ‘external’ structure is applied in order to impart control.
Adding a further layer to an articulated figure, Ng and Grimsdale [Ng95] present a layered model for controlling cloth fold formation. A cylindrical cloth layer is defined around a cylindrical flesh layer covering a skeleton. No expensive collision detection tests are performed and the cloth is not physically modelled. Instead a simple geometrical measurement of distance between perturbed cloth layer grid points and associated flesh layer grid points determines where \( \sin^2(x)/e^{2x} \) fold functions are added to the cloth layer. As Ng and Grimsdale state “fold lines occur along the contours of maximum slackness” in the cloth. However, it is not made clear in the paper how the cloth layer would be scripted in relation to a moving arm or leg or how joints would be handled, although the authors indicate that a rule base could be added to associate folds with underlying geometrical features and a spring-mass-dashpot model could be added between the cloth and flesh layer to generate large-scale deformation automatically.

Van Overveld and Wyvill [Over97] present a polygon inflation algorithm as a layered approach for designing 3D objects. A simple skeleton of polygons is used as a starting point to construct a more detailed object. Individual skeleton polygons are extruded to produce a smooth surface. This combines the naturalness of a simple, coarse model with an easily understood technique – extrusion – to create a more detailed smooth surface. Also the model is compact and changes in the skeleton give predictable changes in the final surface. The authors compare the approach with implicit surfaces in that for implicit surfaces an underlying structure is animated and an implicit surface is created for each frame. However, their process doesn’t suffer from the unwanted blending and bulging problems of implicit surfaces. Also less polygons result in the visualisation stage than in an equivalent model produced using implicit surfaces. The final representation is thus more compact.

### 2.2.1.5 Procedures and primitives

Here, we distinguish between a single procedural or parameterised object for which a set of parameters are defined and procedural organisation of a collection of simple primitive objects. Procedural models are a form of constructive modelling technique in that parameters for a model, e.g. relationships between parts, have to identified and this is most easily done as part of the construction process. Thus each of the previous approaches, e.g. CSG models and Snyder’s meta-shapes [Snyd92], can be included here since they compose an object of parts and parameters can be defined to vary or alter that composition. A spiral staircase illustrates the idea of a model constructed from pieces and composed using sweeping. The model pieces and sweeping process are both amenable to parameterisation.

For procedural organisation we can employ a shape grammar, e.g. L-systems [Pnus90]. Most computer graphics work in this area has been oriented towards producing models of trees and plants. However, we can also use the ideas to structure a general collection of primitives. Combining this with the use of symmetry and using a simple set of primitives, as in CSG, we can compose simple relationships such as square, corner, triangle, centre, circle, which would produce a square with a triangle at each corner and a circle in the centre of each triangle. Parameters can be defined for both the primitives and the symmetry relationships. Figure 2.10
CHAPTER 2. MODELLING AND ANIMATION

shows examples of objects produced using this approach. In effect, we can say that the grammar organises a pattern of primitives into an object.

Figure 2.10: Objects produced using a simple shape grammar

Once the parameterised object is produced, families of objects can be created by varying the parameters. We can thus consider the approach as a decomposition or refinement approach since once the parameters are defined the object is treated as a whole and is refined by altering the parameters. This is similar to a features-based design technique (see section 2.2.2.2) where features of an object must be identified. The features can either be identified in order to construct the model or a whole part can be altered by subtracting or imprinting features (see for example [Cave92]). Such features can be imprinted on another object procedurally, as shown in Figure 2.11 where a truncated pyramid and a cuboid are used to modify the surface of a sphere. Finally, we can also include the ‘cloud’ of objects produced using particle systems [Reev83]
(see section 2.3.2) as an example of a procedural object. The inherent process of database amplification produces a final pattern, albeit an amorphous one.

Figure 2.11: A truncated pyramid and a cuboid are stochastically distributed on the surface of a sphere to produce a more complex object (produced using a 3D Studio Max plug-in)

2.2.2 Refinement approaches

The refinement approach to modelling starts from a ‘whole’ object or surface and changes or refines this. The ‘whole’ object may be composed of parts and the technique may operate on only one part but the object is regarded as a whole since it is the refinement process that is concentrated on. We identify four kinds of refinement:

- direct surface manipulation
- form features
- parameterised surface alteration
- space manipulation – generic deformations

In the section on directly manipulating the points or control points of a surface (e.g. [Fors88, Alla89, Galy91, Bor94, Qin96]), we also discuss the use of ‘tools’ to carry out the manipulation. An extensive survey of feature-based design is not undertaken (see [Parr93] for such a review). Rather the use of form features, e.g. pockets, for surface alteration is considered. This also includes the production of impressions for creating car body panels from flat surfaces (e.g. [Cave92, Cave95]). Parameterised surface alteration is taken to mean to mean those techniques that parameterise the surface in some way and then alter the parameters to achieve a change. An example would be a set of parameters describing relationships between surface features, e.g. parts of a face [Park74, Park96]. Space manipulation captures the range of techniques that deform the space an object is defined in (e.g. [Barr84, Parr86, Sede86, Coqu90, Bor91, Snib92, Laza94, MacC96]) irrespective of the representation used. This section focuses
on the popular free-form deformations [Parr86, Sede86] and axial deformations [Barr84, Laz94].

2.2.2.1 Direct surface manipulation

Although the polygon is the de facto standard in computer graphics and surface manipulation
techniques have been defined for working with polygons (e.g. [Pare77, Alla89]), the majority
of surface deformation work considers parametric patches. This section concentrates on these
although the work on polygons will be discussed where relevant.

One of the most commonly used patch representations is the Bezier patch, named after the
pioneering work of Pierre Bezier on a system called UNISURF [Beze74] in the 1960s. A
cubic Bezier patch can be defined as:

\[ Q(u, v) = \sum_{i=0}^{3} \sum_{j=0}^{3} P_{ij} B_i(u) B_j(v) \]

where \( P_{ij} \) are the control points, \((u, v) \in [0, 1] \) and \( B_i(u) \) and \( B_j(v) \) are Bernstein basis functions:

\[ B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i \]

where \( t \in [0, 1] \) and \( n=3 \) when used with a cubic patch. This is illustrated in Figure 2.12 where
a single control point is progressively moved. The well-documented properties of the Bezier
patch (see for instance [Watt93]) mean that the resulting surface distortion is predictable – it
mimics the shape of the control hull formed by the control points.

![Figure 2.12: Progressive movement of a control point for a bicubic Bezier patch](image)

A single cubic Bezier patch can capture a range of useful shapes but for more complex shapes
an alternative approach is required. Higher degree Bezier patches are not commonly used as
they can suffer from unwanted ‘wiggles’. Instead an alternative parametric representation can
be employed (see later) or a quilt of Bezier patches is used. Such a quilt can be obtained from
the results of any one of a number of input sources, for example the sampling of an existing 3D
object with a laser scanner or the output of a conversion process on one of the aforementioned
CHAPTER 2. MODELLING AND ANIMATION

constructive modelling techniques. Bezier states that “less than 30 major patches” can be used to define half a car [Beze74] although over 50 patches are then added for details such as “handle recesses, gougings, windshield housing, etc… although they do not cover more than 5% of the total surface”.

Figure 2.13: The Utah teapot; (a) lines of constant \( u \) and \( v \); (b) exploded wireframe of control points; (c) wireframe of the patch edges

Figure 2.13 shows the (famous) Utah teapot, defined using 32 bicubic Bezier patches. The teapot illustrates one of the main advantages of Bezier patches over a polygon representation. Besides the fact that it is an exact analytical representation, it is a more economical representation. 32 patches can be stored as approximately 288 three-dimensional values (since most patches share 12 control points). A ‘reasonable’ polygon mesh representation for the teapot would require, say, 2048 four-sided polygons (see Figure 2.14) which is 2048 three-dimensional values. The inexact polygon representation uses more than seven times as much memory. In practice, the application would determine the exact ‘cost’ of comparing the two representations. Whilst “patches might be storage efficient for an accurate human body” [Badl78], their use in an interactive computer animation application involving many moving human figures may prove less cost effective than using polygon models, especially as current graphics hardware is tailored for fast display of polygons.

One of the main problems with using a net of Bezier patches is that continuity constraints must be maintained between patches as control points are moved. Traditionally parametric derivatives are matched at the joins to give \( C^n \) continuity. Many researchers, especially for computer graphics work as opposed to CAD, have relaxed this and specified weaker constraints using geometry continuity, or \( G^n \) continuity. For instance a \( G^1 \) continuous surface would possess a continuous tangent plane in addition to a continuous position (i.e. \( G^0 \)). Relaxing the continuity constraints allows for extra control parameters to be used on the surface. For
instance, Schmitt and Du define a set of shape control parameters called tension, bias and continuity that can be used to adjust the behaviour of patches at joins [Schm87]. An alternative approach is to use a different parametric representation where continuity constraints are less of a problem.

![Figure 2.14: The Utah teapot drawn as a wireframe by converting the patches to polygons using uniform subdivision (32, 128, 512 and 2048 polygons respectively).](image)

For the B-spline patch representation more control points can be defined for a single patch without raising the degree of the patch. Predictable surface shape control is maintained and more complex shapes can be described (see [Cobb84] for examples) since more control points are available. This contrasts with Bezier patches where more control points for a single patch can only be achieved by raising the degree of the patch and introducing unwanted undulations and ‘wiggles’ into the surface. If the B-spline patches are required to interpolate initial surface data, then the usual process is to insert multiple knots and, in effect, split the representation into separate patches (equivalent to a net of Bezier patches) and thus issues of continuity between patches returns. Non-Uniform rational B-Spline surfaces (NURBS) [Pie91] offer more parameters to control the shape of an object, however the control of knots and weights associated with the control points is arguably less predictable on surface shape than movement of control points [Blan95, Geor92]. Whichever representation is adopted there is still the issue of controlling the deformation of groups of points since moving single points in a large net of points is a time-consuming and error-prone process. A technique is needed to move groups of control points in a controlled way.

The metaphor of sculpting can be used to describe how a group of control points can be moved as a unit. Tools such as repelling tools can be used to create depressions and attraction tools can be used to create protrusions. Such operations can be defined as procedures with parameters.
CHAPTER 2. MODELLING AND ANIMATION

that distort regions of a surface in a controlled way. In effect the operation becomes a black box with a series of dials to adjust the parameters. The idea is to provide the designer with a perceptual connection between the tool used to effect deformation and the resulting shape change. Parent’s work [Pare77] likened the process to moulding a piece of clay. In this section we focus on surface operations where the metaphor of a rubber sheet is more appropriate. General space sculpting is the focus of a later section.

We can identify the following general processes:

- identifying the point or group of points to be directly moved by the designer
- identifying a group of points to be influenced by such movement – this involves some distance criteria
- isolating points from being affected by any subsequent movement – adding constraints
- localising the deformation effects

As already mentioned the simplest operation that can be performed is to move a single vertex or control point. For a polygon surface, moving a single vertex with no effect on surrounding vertices is of limited use. In contrast the properties of the parametric patch formulation means that moving a single control point smoothly affects an area of the surface. In effect, a ‘bump’ is added to the surface. The size of this bump is dependent on the numbers of control points defining the surface. More control points within an area means more local control. This basic point-based bump is the starting point for further work.

Cobb [Cobb84] uses a 2D polygon and a 2D polyline, e.g. a letter Z, to identify a collection of points to directly move. Both are easily defined in 2D, making them intuitive to a designer. Cobb uses the polygon as a region warp and the polyline as a skeletal warp on a B-spline surface. The polygon is aligned with a projection of the control points based on the current view to determine which set of control points to move. The polyline is used in a similar manner. Having identified a set of points to move, and more complex sets could be identified, perhaps at the loss of intuitiveness, the region of influence of the subsequent surface warp must be defined. Cobb (quoting earlier work by Carlson [Carl82]) lists a range of distance measures for identifying which surface points move in a warp:

- simple Euclidean distance between two points
- counting the number of arcs traversed on the path from one point to the other
- an accumulated Euclidean distance [which] is the sum of the lengths of all the arcs traversed on the path from one point to the other (used by Carlson)

Allan et al [Alla89] use simple Euclidean distance. A current vertex on a polygon mesh is initially selected and a circular radius of influence can be altered to determine the vertices to be affected by the movement. The movement of the points is controlled by a decay function – cone bell, cusp, random decay and sine wave are defined – centred on the current vertex. A cone function gives an effect not unlike moving a single point on a parametric patch. Parent [Pare77] also works with polygon meshes and achieves a similar effect – called a group warp - by using an interpolation routine that pulls neighbouring points proportionally to the current vertex. The weighting function can be interactively adjusted to give different bump profiles within an area.
CHAPTER 2. MODELLING AND ANIMATION

of influence. Cobb quotes Carlson as arguing that simple Euclidean distance is not necessarily the best measurement as two points that are close in Euclidean distance are not necessarily close topologically. Thus either of the path measurements is better. To ensure greatest flexibility, Cobb supplies a range of measures for the designer to choose from.

In designing an object the designer will also want to restrict the movement of certain points so as to introduce such things as hard edges into the surface. This can be done by anchoring vertices in a polygon mesh [Alla87, Pare77] or isolating a region of control points in a B-spline mesh [Cobb84] or by increasing the multiplicity of knots in the surface definition [Cobb84, Blan94]. Blanc and Guitton [Blan94] hide the complexity from the designer by supplying a crease tool which can be applied to their NURBS surface. Other such tools can be defined which manipulate a set of points in a particular way thus making the manipulation process more intuitive for a designer. For instance, both Cobb [Cobb84] and Ostby [Ostb86] define flattening operations that make one or more control points lie in the same plane. Ostby also alludes to “non-interactive programs which distort regions of patches in a controlled way” but alas gives no details. The only comment is that such programs are needed for very complex models, as it becomes too difficult to identify and move individual control points.

Constraint-based work has been done by Borrel and Rappoport [Borr94] and Celniker and Gossard [Celn91]. The work of Borrel and Rapport is perhaps more akin to a space deformation technique, since local B-spline basis functions are centred at constraint points and used to deform the space below within a user-controllable radius. Celniker and Gossard satisfy geometric constraints and load forces on a finite element representation by minimising an energy functional. The main advantages of the geometric constraint-based techniques are that arbitrary surface points can be manipulated and the designer is not concerned about the properties of the underlying surface representation. However, as Borrel and Rapport point out, “the type of deformation is not related to the fact that it is defined by constraints” [Borr94]. The same shape could have been computed more directly using control point manipulation. In general, the physically-based work is too slow for interactive design and the representation must be one that supports such simulations and therefore cannot be arbitrary [Borr94]. This problem is however addressed by the work of Terzopoulos and Qin [Terz94, Qin96] on D-NURBS, a physically-based generalisation of standard NURBS surfaces. Shape control is effected by the usual geometric control point or knot manipulation or by satisfying applied forces and constraints. The physically-modelled surface net causes points to respond to the movement of others within a region. This is similar to the geometric warps discussed above, although it is perhaps less clear how to control the parameters of the physical model to achieve the exact localised effect.

For detailed, local modifications to a surface, the surface resolution must be increased so that there are enough surface defining points within an area to realise the desired effect. For a polygon mesh, individual polygons can be separately subdivided without affecting neighbouring polygons. There are no continuity problems to concern. Allan et al simply subdivide a polygon into many polygons without taking implied surface curvature into account. Recent work by van Overveld and Wyvill [Over98], which uses vertex normals to fit patches.
and then subdivide the surface, could be used to address this although the surface produced can only be a guess at what the true surface was. For a quilt of patches, increasing the number of control points for one patch has knock on effects for neighbouring patches. For a net of spline patches, rows and columns of control points must be inserted. These span across the lines of neighbouring patches in the north, south, east and west directions from the initial patch [Watt92].

The preceding discussion points can be illustrated using a quilt of bicubic Bezier patches. We can address the main problem of using nets of bicubic Bezier patches by defining manipulation units that span across multiple patches. Figure 2.15 shows a manipulation unit for four adjoining bicubic Bezier patches. This manipulation unit, which consists of nine control points, must be scripted as a single unit so as to ensure continuity between the adjoining patches. The use of manipulation units means that local deformation effects can be applied. The relevant Bezier patches can be subdivided uniformly and manipulation units at finer and finer levels of subdivision can be manipulated.

Figure 2.15: Four adjoining bicubic Bezier patches. The group of nine control points centred on the meeting point of the four patches can be moved as a unit whilst maintaining continuity across the joins.

Figures 2.16a-e show the use of manipulation units to deform a cube, where each face is defined using multiple bicubic Bezier patches. In Figures 2.16a and b, a column of manipulation units is moved, which is similar to Cobb’s skeletal warp. In Figures 2.16c and d, a single manipulation unit is moved in the centre of the cube face which corresponds to the basic bump warp. In Figures 2.16b and d, the cube face patches are uniformly subdivided to give more local control over the deformation. In Figure 2.16e, multiple manipulation units are moved on each face to produce a simple die. In each case the surface is visualised as a wireframe of polygons by using uniform de Casteljau subdivision.
**Figure 2.16: Local surface deformation of a cube defined using Bezier patches**

The main problem with this approach is that as the surface is subdivided to get more and more local control, it becomes difficult to then apply a more global deformation. The problems discussed previously emerge again and larger groups of control points must be identified and manipulated as some form of unit. This suggests using some form of hierarchical editing, where changes can be made at a level of detail desired by the user. Forsey and Bartels [Fors88] introduced an approach based on hierarchical B-splines in order to address this. Instead of subdividing the whole surface, the surface is only subdivided in the area of interest (using the Oslo algorithm [Ries81]). Thus a ‘minimal surface’ is identified around the control point to be moved. This minimal surface is the area affected by the movement of the control point and regardless of the movement of the control point the boundaries of the minimal surface will join smoothly with the original surface. The minimal surface can be further subdivided so that a smaller minimal surface is defined around the control point to be moved, thus further localising the effect of the movement (see Figure 2.17). The minimal surfaces are stored as a hierarchy of overlays. Deformation can be performed at any level in the hierarchy by refining the appropriate minimal surface overlay. Each overlay is stored as offsets from the previous hierarchy level. Thus control point movement at a coarse level affects subsequent finer overlays and fine level adjustment is overlaid or carried on top of coarser levels.

Allan et al [All89] points out that a similar process can be done with a polygon surface. Individual vertices can be moved, with a radius of influence and a warp function determining the effect of movement on neighbouring vertices, and individual polygons can be subdivided to give appropriate representation quality to perform the deformation. Allan et al demonstrate the effectiveness of this by producing a figure of an animal’s head similar to the one produced by
Forsey and Bartels to illustrate their work. Allan et al do not store and use a hierarchy of overlays, but there seems to be no reason why such a scheme couldn't be implemented for polygon models using relative coordinate positions for each overlay. The main difference with the B-spline-based approach would be a lack of smooth blending between levels, although a suitable blending process could be defined for polygon levels using subdivision of adjoining polygons.

Figure 2.17: Local surface deformation of a B-spline surface (used with permission from [Watt92])

The techniques used by Allan et al and by Forsey and Bartels are similar. It is the modelling representation used that is the main difference. The B-spline surface (and generally any parametric representation) is an exact mathematical representation that guarantees smooth surfaces. Its very nature is well suited as a Computer-Aided Design (CAD) representation, where mathematical reasoning about smoothness criteria is important. The polygon in contrast relies on the deformation technique being used with a mathematical function to ensure some form of 'continuous spread' of any deformation. It is not an exact representation and for an acceptable smooth representation requires an order of magnitude increase in amount of data to represent the same surface as a B-spline representation. But the differences between the two are less important for computer graphics than for CAD. In computer graphics, both would serve adequately for modelling smooth objects, each having relative advantages. As we have seen the modelling techniques used upon them are similar. The number of control points or vertices can
be handled and continuity issues can be controlled. Perhaps the main issue, in computer graphics work, is how much data can be handled at interactive rates when the deformable model is being animated. More will be said about this in Chapter 4, which will consider using multiple levels of detail for an object in a real-time environment.

In the previous discussion, little has been said about interaction techniques. A common theme that emerges is the recourse to 2D to edit in 3D. Many CAD systems avoid true 3D editing by restricting the movement of an editing cursor to two dimensions. For instance, successive or multiple 2D projections are used to accomplish a 3D movement. Allan et al define a simple interface that involves selecting one of the polygons of the current vertex and then moving the resulting plane normal to the polygon and moving the current vertex within the plane. Earlier versions of the industry standard AutoCAD used a similar approach calling the ‘working plane’ the ‘User Coordinate System’. Cobb appears to specify most of the information needed for the interface in 2D. For instance the regional warp is specified by defining a polygon in 2D and projecting the B-spline control points onto this based on the current view. Jeng and Xiang [Jeng96] describe a technique involving a user-controlled cursor plane intersecting a fully rendered object. A parallel projection is used and the cursor plane is parallel to the display surface. The depth of the cursor cross can be controlled and the intersection of the cursor plane with the object is indicated by a surface contour. Colour coding indicates proximity to the cursor cross and individual vertices can be manipulated with the cursor cross. Switching from this editing mode to a view mode allows the object to be positioned relative to the cursor plane. By using the cursor plane to segment the model into two halves, the positioning of vertices is made clearer as they move forwards or backwards relative to the plane. Even many of the space deformation techniques use 2D to edit in 3D as will be shown in a later section.

Instead of directly manipulating the surface, Vickers et al [Vick78] interactively adjust marine propellers using a superpositioned ‘influence surface’. The influence surface is an “imaginary thin, flexible elastic plate”, subjected to point displacements with the resulting deformation calculated using relevant equations. The advantage over 3D editing is that the influence surface can be adjusted in a more spatially constrained interface. A similar spatial constraint is adopted by Pentland et al [Pent90]. They present a surface refinement technique that produces detail by painting onto a ‘flattened’ 2D version of a surface. Brighter shades of grey are used to produce peaks and ridges and darker greys produce valleys and pits.

In contrast to the surface techniques, Galyean and Hughes [Galy91] truly make use of the sculpting metaphor in three-dimensions. A 3D input device is used to modify a 3D volume or voxmap. An analogy is drawn with a 2D paint system, where a pixel is coloured to indicate the presence of a certain amount of ink. In the same way the vertices of a 3D grid are ‘painted’ with values and a variation on the marching cubes algorithm [Lore87] is used to extract a polygon surface. Tools include an additive tool (analogous to squeezing out toothpaste) which leaves a trail of material, a heat gun to melt away material (the opposite of building up material using a paint spray in 2D) and sandpaper to smooth areas (similar to low-pass filters in 2D paint systems). A further addition to such a system would be some form of force feedback although it
is not clear how a force feedback mechanism should present the surface shape to the user [Hiro95].

We can make a comparison with the use of implicit surfaces for modelling and the Galvanean and Hughes voxmaps. A low resolution voxmap can be used to produce a coarse version of a model, just as a few ‘soft object’ primitives can be used to produce a coarse description of a model. The low resolution voxmap can then be subdivided to produce a high resolution 3D grid for finer localised sculpting operations. For the soft object model, more primitives can be added to refine localised areas. Yet one is using a refinement approach to modelling and the other a constructive approach. For the voxmap approach a whole block of material is ‘edited down’, whereas for the soft object approach, the model is ‘built up’. The voxmap results in a surface with no internal structure. The use of soft objects, as with the other constructive approaches, results in a global structure and a surface.

2.2.2.2 Form features

The word ‘feature’ in relation to design conjures up many possible meanings. As Vieira et al. state there is “no common precise definition of what a feature is” [Vie96]. Parry-Barwick and Bowyer [Parr93] define three ways of looking at features:

- Functional feature – this specifies the (mechanical) operation of the feature without the constraints associated with topological and geometric information;
- Design feature – expression is in geometric terms; function is implicit; design features can also be further considered in terms of form or in terms of material, tolerance and surface finishes;
- Manufacturing feature – method of production is expressed explicitly; function is implicit.

These different views highlight the fact that the intent of the feature-based approach is to unite the different stages of a product’s life cycle, e.g. process planning and manufacture, linking CAD and CAM. From this brief introduction we might initially consider that the use of features seems somewhat specialised to CAD. However, further elaboration will show that there are ideas applicable to general computer graphics.

We are most interested in design-by-features and, in particular, in the use of form features. Cavendish and Marin [Cave92] describe these as “local geometric configurations” such as “pockets, channels, beads and rims”. Parry-Barwick and Bowyer present a more formal taxonomy adapted from Gindy’s work [Gind89], as shown in Figure 2.18. This taxonomy again indicates the CAD/CAM linkage. The general form of a feature, e.g. a depression, is manufactured from a number of external access directions using a milling machine. As an example, the ‘Through Slot’ can be milled from three directions, the boundary is open in contrast to a closed boundary such as a pocket and there are no blind spots for the milling tool such as there would be with a notch. Gindy’s taxonomy can also be viewed as an elaboration of general form features to pre-defined design features that are related in a strong way to the definition and classification of machining features. Such pre-defined design features (see
Figure 2.19) might include cylindrical holes, rectangular pockets and simple slots accompanied by parameters to alter their dimensions.

Parry-Barwick and Bowyer further categorise the design-by-features approach into:

- Synthesis by features – parameterised features are combined using a set theoretic modeller
- Destructive modelling with features – subtract features from a ‘blank’
- Design without the use of a solid modeller – the use of a language such as LISP

This suggests that we could classify the use of features as a constructive approach to modelling. However, we argue that, in general, the process can be regarded as more of a refinement process. The model is not build up using global features. It is refined with features added to or subtracted locally from a starting model.

Figure 2.18: Gindy’s form feature taxonomy (after [Par93])
Cavendish and Marin describe their feature-based design process as one of surface assembly with “secondary component surfaces” (the features) being combined with the primary surface (the initial panel). This fits the definition of a refinement approach with a whole (the primary surface) being refined using features. The features are blended with the primary surface to give a smooth join. They illustrate their work with a typical car that “contains more than 200 stamped sheet metal parts”. About 10 of these are free-form outer panels and the rest are multi-featured, irregular, functional inner panels. (The resulting panels are then used in a constructive approach to complete the whole car.)

The main limitation of the design-by-features approach is that the features are limited in scope to those that can be manufactured with machine tools, e.g. slots and pockets. Designers can be allowed to design their own features, but such features may prove difficult to manufacture, thus compromising the main intent of the features-based approach to provide a CAD/CAM linkage. Also there are issues of consistency to be considered [Vie96]. Incorporation of the feature-based technique into a set theoretic modeller is relatively easy, but incorporation into a boundary modeller is less so. Features added into a boundary representation are in effect lost – it is a destructive approach – unless a secondary data structure is used to record the process.
It seems that the features-based approach, like CSG is best suited to modelling man-made or machinable parts. After all, that is its raison d'être. But the approach can be used to model natural objects, such as a human face, in an intuitive manner by blending a feature, such as a nose, into an underlying surface. Indeed it might be said that a sculptor thinks in this way. For instance, he might say that he is working on the nose when asked what part of an object he is working on, rather than saying a particular set of control points. Thus the control points of the original surface are not manipulated to make a nose, but a local area is identified and a feature is made for this area. The feature can be defined as a separate overlay surface and blended into the initial boundary surface. Wyvill et al [Wyvi97] describe such a feature-based approach where features, such as hooks, can be defined as a separate surface and blended into a base surface. Inspiration for this approach can be found in Forsey and Bartels' work with hierarchical B-splines [Fors88].

The features-based approach can thus be regarded as intuitive for refining a gross approximation to a model. It is easy to see how a coarse model of a face could be constructed from a sphere when it is seen as a collection of protrusions and depressions. A designer would work feature by feature in refining areas of the face. As an example, Wyvill et al produce a model of a head using 11 features. It is however only a coarse model and would require much more refinement to create a detailed model. Interaction between features is less clear which has implications for animation. The features approach can produce a compact representation with features being parameterised and repeated. The features can also be separately edited, either as surfaces (in which previous surface techniques become applicable) or through parameters, thus focussing on local and less complex areas of a full model. In effect the features become primitives and the distinction between a refinement approach and a constructive approach is slightly blurred.

2.2.2.3 Parameterised surface alteration

For parameterised surface alteration, the general idea is that rather than directly manipulate the surface, say, using control points, other parameters are provided to alter surface shape. A designer can thus explore the range of shapes produced by altering a few numbers that, hopefully, give intuitive and expressive control. To a certain degree, this technique has been covered in section 2.2.2.1, where NURBS knots and weights were given as an example, although as previously stated the control of knots and weights associated with the control points is arguably less predictable on surface shape than movement of control points [Blan95; Geor92].

Another example is work related to the constraint-based ideas discussed in section 2.2.2.1. Rappoport et al [Rapp94] introduce a design technique for spline surfaces based on what they call soft constraints. In other words the constraint is not rigid and does not have to be met exactly. A difference between this and the techniques for anchoring points or introducing creases discussed in section 2.2.2.1 is that any point on the surface can act as a control point. Both global and local parameters are available. A global parameter determines the global amount of deviation of the new surface from the existing one. The local parameters are defined
at the constraints and determine how ‘soft’ the constraint is. This local soft parameter is also nonisotropic. The designer specifies constraints at certain points and can then interactively explore the family of shapes that result from satisfying the constraints by adjusting the relevant local and global parameters.

Parameters can also be defined for the relationship between form features. A specific example is Parke’s work [Park74, Park96] on defining relationships between facial features for facial animation work. Example parameters include chin-to-mouth scaling, eyebrow separation and jaw width. Parke’s work will be discussed in greater detail in the next chapter.

Figure 2.20 gives an example of the use of parameterised surface alteration, or in this case parameterised object alteration. A teapot is stretched into a coffee pot. Figure 2.20a shows the result of the whole teapot being scaled upwards from the base. In Figure 2.20b, the lid has been treated as a separate object and is translated, rather than being scaled. In effect, the parts of the teapot are separately parameterised. The distinction between this and the use of procedures and parameters in section 2.2.1.5 is that the parameter is dynamically attached to a predefined object for the purposes of manipulation rather than being part of the original specification of the object.

![Figure 2.20](image.png)

Figure 2.20: Parameterised object alteration of a teapot: (a) global scale; (b) the lid is treated as a separate object

### 2.2.2.4 Space manipulation – generic deformations

This section focuses on the range of techniques that deform the space an object is defined in irrespective of the representation used. The aim is to treat an object as if it were made of a type of “topological putty or clay which may be bent, twisted, tapered, compressed, expanded and otherwise transformed repeatedly into a final shape” [Barr84]. We can split the techniques into two categories: axial-based deformation and general space deformation. For general space deformations a ‘container’ is used to typically deform an object from ‘outside’. The container
surrounds the whole object or part of the object – there appears to be no work that embeds such a container within an object to deform the surrounding space. For axial-based deformations, the typical approach is to use a ‘skeleton’ axis to deform a surface from ‘within’, although many researchers also show that using the axis externally can produce useful surface deformation effects.

Barr [Barr84] can be credited with introducing the idea of an axial deformation, with his introduction of twisting, bending and tapering operations for geometric objects. For a point \((x,y,z)\), the deformed point \((X,Y,Z)\) can be given for Tapering and Twisting, using the \(z\) axis as the axis for deformations, as follows:

Tapering: \( r = f(z); \ X = rx; \ Y = ry; \ Z = z; \) where \(f(z)\) is a function that decreases as \(z\) increases;

Twisting: \( \theta = f(z); \ X = xC_\theta - yS_\theta; \ Y = xS_\theta + yC_\theta; \ Z = z; \) where \(C_\theta = \cos(\theta)\) and \(S_\theta = \sin(\theta)\), and \(f(z)\) specifies the rate of twist per unit length along the \(z\) axis.

Figure 2.21 illustrates these two operations. The shell-like object is formed by initially sweeping a corrugated circle (a circle with added sine wave) along a straight line and then tapering and twisting.

![Figure 2.21: A shell-like object formed using twisting and tapering operations](image)

Bending is a slightly more complex operation since the region of bend must be defined, together with continuity behaviour at the joins of the bending region to the unbent region. Barr presents a solution that causes a jump in the derivative of the bend angle at the joins, although suggests that a continuous function could be defined. Fournier and Wesley [Four83] give a more comprehensive presentation of the use of bending processes for manufacturing parts in the CAD/CAM field. Three regions are identified in their work: the fixed region, the bending region and the relocation region, i.e. the part that is moved, not bent. Also in their work, a bending axis can be placed outside an object thus a straight staircase can easily be turned into a spiral staircase.
Figure 2.22: Barr's transformations (see [Barr84]) applied to a teapot and a cube

Figure 2.22 shows the use of Barr's transformations on a cube and teapot, both defined using bicubic Bezier patches. Figure 2.23 shows a rendered version of the taper operation. As described earlier (see figure 2.15), the control points are grouped around patch corners into manipulation units and it is the manipulation units that are transformed. The technique appears to work quite well. However there is a problem when the cube face is subdivided into more patches, which in turn produces more manipulation units. The problem is highlighted in Figure 2.24 and shows unwanted undulations in the cube surface. These are not revealed in the teapot surface because, for instance, the Barr taper operation is perhaps interpreted as looking 'correct' and showing a squashing effect before a jump. Figure 2.25 illustrates why the problem with manipulation units occurs by simplifying the explanation using cubic Bezier curves and groups of three control points centred on the joins. The sequence of curve segments is tapered from left to right. The groups of three control points are simply translated and 'platforms' appear in the piecewise curve, with more platforms appearing when the curve is subdivided. Whilst simple rotations would solve this problem, the solution in 3D is less clear.
CHAPTER 2. MODELLING AND ANIMATION

Figure 2.23: A rendered version of the tapered teapot in Figure 2.22

Figure 2.24: Unwanted undulations on the surface of the cube caused by manipulation units

Figure 2.25: The problem of using manipulation units illustrated using curves

In Barr’s work the axis is a straight line. Lazanu et al [Laza94] extend the use of axial deformations to include a curved bend axis defined using splines. Each point of an object to be
CHAPTER 2. MODELLING AND ANIMATION

deformed is associated with one point on the curve using a local coordinate system defined at
the respective point on the curve. Thus as the curve deforms, the set of local coordinate systems
moves and in turn each object point moves with its local coordinate system. Two generalised
cylindrical zones of influence, one contained in the other are defined around the axis in order to
localise the deformation.

Singh and Fiume [Sing98] make use of two curves to define an axial deformation; a wire curve
and a congruent copy called the reference curve. Points in an object are associated with wire
curves and deformation is relative to the movement of wire curves with respect to reference
curves. As with Lazarus et al’s work, a field of influence around each wire means that local
surface deformations can be performed. In addition wires can be used to join separate surfaces
together (facilitating a form of constructive modelling technique) and can be joined to form
lattices that can then deform a whole object in a similar way to a general space or container
deformation, e.g. FFD (see later).

Bar’s simple axial deformations introduce intuitive ‘global’ modelling operations for a
designer, namely bend, twist and taper. The subsequent general axis deformation work (e.g.
[Lazi94]) adds local control at the expense of some extra complexity. For surface effects, the
general axis deformation is not unlike Cobb’s skeletal warp [Cobb84]. One concern is how to
position the deformation axis and associate the parameters with their position relative to the
axis and their effect on the object. Snibbe et al [Snib92, Conn92] use 3D widgets to do this for
the simple axial deformations, although the technique could be extended for general axial
deformations. Figure 2.26 shows how a widget – a ‘rack’ with ‘handles’ – can be used to
provide design control for tapering and twisting operations. The metaphor is thus of a real tool
embedded within the object and direct, intuitive 3D manipulation of an object is effected and
simplified.

Figure 2.26: Using a widget to control simple axial deformations (after [Watt98])
CHAPTER 2. MODELLING AND ANIMATION

A similar control approach is offered by Chang and Rockwood [Chan94]. They use a mixture of a Bezier curve skeleton and local axes defined on each Bezier control polygon segment. The local axes act like handles to control tapering and twisting along the curve. The curve itself can be altered to produce stretching and bending operations. A generalised de Casteljau algorithm is used to warp an object along the Bezier curve.

The second category of space deformation technique is the free-form deformation (FFD) technique introduced by Sederberg and Parry [Sede86; Parr86], although Bezier had used a similar tripertametric volume in his UNISURF system to distort the shape of objects [Bezi78]. Sederberg and Parry suggest the analogy of a parallelepiped of clear, flexible plastic in which the object is embedded. The simpler plastic container is distorted and the embedded object distorts with it. The major advantage of the FFD technique is that it is independent of object representation. For instance, the object could be represented as a solid model or as a polygon net.

The FFD approach involves a mapping from \( \mathbb{R}^3 \) to \( \mathbb{R}^3 \) through a trivariate tensor product Bernstein polynomial. A single tricubic Bezier hyperpatch is defined as:

\[
Q(u, v, w) = \sum_{i=0}^{3} \sum_{j=0}^{3} \sum_{k=0}^{3} p_{ijk} B_i(u) B_j(v) B_k(w)
\]

where \( B_i(u) \), \( B_j(v) \) and \( B_k(w) \) are Bernstein polynomials of degree 3 and \( p_{ijk} \) are a three dimensional grid of 64 control points defining a volume of space with \( u, v, w \in [0, 1] \). For each object vertex, the world coordinates \((x,y,z)\) are defined relative to the enclosing parametric space to give \((u,v,w)\), the control points of the parametric space \( p_{ijk} \) are moved to \( p'_{ijk} \), and the \((u,v,w)\) are substituted into \( Q'(u,v,w) \) to give the deformed \((x,y,z)'\). Figure 2.27 shows the use of a single hyperpatch to deform a sphere.

![Image of FFD example: A single hyperpatch is used to deform a sphere (used with permission from [Watt92])](img)

Figure 2.27: FFD example: A single hyperpatch is used to deform a sphere (used with permission from [Watt92])
CHAPTER 2. MODELLING AND ANIMATION

Hyperpatches can be connected together to give more local control over the enclosed object although continuity must then be maintained between patches, or the hyperpatch can be used to deform just part of an object. For instance Chadwick uses FFDs to deform the muscles on a character’s arm [Chad89] and Watt and Watt [Watt92] use them to deform the neck, head and tail of a horse.

In general the hyperpatch is useful for global deformations. Cavendish [Cave95] gives a good example of this. FFDs are used to apply global bending operations to panels initially designed using a feature-based approach. Cavendish describes how stamped sheet panels have a tendency to ‘spring back’ to a shape that is slightly different from the shape of the die face used for stamping. This can be compensated for by using FFDs in the design of the die face. FFDs are less successful for complex, subtle, local deformations where the number of FFD blocks required would produce a clutter of control points than could be more easily replaced by using direct surface manipulation for certain representations. Having said this, Kalra [Karl92] uses FFDs to simulate facial muscles. The difference here is that the FFD is itself parameterised and it is the extra level of parameterisation that is actually manipulated not the control points of the FFD. More examples of the use of FFDs for facial animation work will be given in Chapter 3. In particular the use of FFDs for facial conformation work, that is the shape and proportions of the face, will be presented.

A number of extensions have been made to the basic FFD technique. Greissmair and Purgathofer [Grei89] use trivariate B-splines although they claim no benefits for this. Instead their work concentrates on sufficiently locally subdividing the surface to be deformed so that piecewise irregularities are not revealed. The intrinsic parallelepiped FFD lattice shape prohibits arbitrary shape deformations. For example, defining a circular bump on a surface is not possible using FFDs. Coquillart [Coqu90] defeats this problem by introducing EFFDs that allow non-parallelepipedical lattices. Instead of general space deformations, she concentrates on their use for surface deformation, extending the warp ideas introduced in Cobb’s work [Cobb84]. By welding lattices together Coquillart creates general EFFD blocks that can be used to create a library of deformation tools, like moulds in a stamping process. One problem with her technique is that it relies on numerical techniques to calculate the parameterisation of a point within the initial trivariate volume. More recently, MacCraken and Joy [MacC96] have introduced FFD lattices of arbitrary topology. Other researchers have focussed on adding physical properties to the FFD mesh. Rappoport et al [Rapp96] address the problem of preserving volume when using FFDs, something of obvious importance in industrial design, and Faloutsos et al [Falo97] present hierarchical dynamic FFDs for use in animation.

One problem with the general FFD approach is that the underlying object only mimics the deformation applied to the FFD lattice. Exact control of particular parts of a surface is not possible. In addition, the use of piecewise FFD lattices can confront the designer with the complexity of many control points to move to achieve a desired effect. Also the screen can become cluttered when many control points are used and some may even be occluded by the object being deformed. Hsu et al [Hsu92] address these problems by allowing the designer to directly manipulate the surface of the object. A (virtual) point (or multiple points) on the object
CHAPTER 2. MODELLING AND ANIMATION

surface is directly moved and the necessary alteration to the control points of the FFD mesh that will induce this change is automatically computed. The fact that surface points are being manipulated seems to obviate the need for an FFD mesh. But there are two reasons why it is useful. The first is that the surface point being moved could be a virtual point on a solid model in the sense that the user picks a point and indicates where it should go. The point is not given directly as part of the representation and so could not normally be manipulated. Secondly, the resulting FFD mesh could be reapplied to a higher detailed model or a model at a different scale and still produce the same deformation effect.

We can summarise this section on modelling techniques with two suggestions drawn from Pentland et al’s work [Pent90]. First, those modelling techniques that use an analogy to physical processes that people are familiar with have a better chance of being accepted. For instance the notion of sculpting with clay is suggestive of a fast medium to work with and has intuitive behaviour. Pulling a rubber sheet around works in a similar way, although it is perhaps less intuitive when it comes to localising effects. Second, modelling is an iterative process that starts with roughing in and proceeds to detailing. Thus a good modelling system must allow the level of detail to be adjusted during the modelling process. In following these two suggestions, Pentland et al produce a modeller that works at two levels. First, a constructive technique based on defoming and joining lumps of clay is used in the roughing out phase. Second, a refinement technique is used to add surface detail. It seems that the best modeller, especially for natural objects, will offer the designer a range of techniques that can be applied at different stages of the design process. But there remains a problem. A detailed model that respects only final surface appearance (or form) is not necessarily going to be easy to control for subsequent use and possible shape change in an animation sequence.

2.3 Animation techniques

Traditional animation, as used in, say, Disney films, involved creating the illusion of movement by photographing a series of individual drawings on successive frames of film. Computer animation replaces photographing, drawings and successive frames with the display on screen of a series of individual states of a dynamic scene [Thal94]. The essential idea is that each state of a scene can be represented as a set of numbers and changing these numbers over time gives animation. This simple definition could encompass the morphing of 2D images and the use of complex, physically-based models in 3D worlds. In the 2D case the numbers are the pixel values and the parameters of the morphing process. In the 3D case the numbers describe the 3D model and the parameters to control it. We might then say that for animating 3D objects, animation techniques differ in the parameters they offer and the control that those parameters give over the resulting motion or deformation.

Wilhelms [Wil87] offers an early classification of animation techniques based on the level of abstraction that specifies the motion. Low-level control methods specify each parameter of a model and higher levels offer more automatic methods of motion control, for instance stimulus-response behavioural control where environmental interactions are taken into account during motion generation. Thus an animator can specify general movement in more abstract terms.
Watt and Watt [Watt92] offer a similar classification although they label many of the techniques as medium level control since they are specialised for a particular domain of applications. An example would be a stochastic animation technique such as the use of particle systems [Reev83]. Watt and Watt distinguish high-level control in systems that orchestrate a collection of animation techniques where high-level commands are compiled down to low-level actions and higher levels respond to feedback from lower levels. Zeltzer [Zelt85] uses guiding, animator-level and task-level to describe similar levels.

Instead of focussing explicitly on low and high level control methods, Hegron and Amaldi [Hegr92] classify motion and deformation control methods into three general families: descriptive or phenomenological models, generative or physically-based models, and behavioural models. The differences are based on considering cause and effect when producing change. Descriptive models cover direct manipulation of object parameters where an effect is produced without knowledge of its cause. Generative models include the dynamics of articulated rigid objects and deformable models and thus describe the cause that produces the effects. Behavioural models simulate the actions of living things that respond to internal and external stimulation. Thalmann and Thalmann [Thal94] use the words geometric, physical and behavioural in a similar classification.

Figure 2.28: Animation techniques

Figure 2.28 illustrates Hegron and Amaldi’s classification using different sized arrows to indicate the quantity of parameters that need to be controlled. A large arrow means that many parameters must be input by the animator. Two arrows are used for the behavioural family to indicate that some behavioural techniques need quite a difference in input. We can use the quantity of parameters to relate the three general ways of looking at animation techniques described above. Descriptive techniques are, in general, low level and involve controlling many parameters. Full control of each object or scene attribute is available although at the expense
and tedious of much data to specify. Generative techniques involve a reduction in parameters through some algorithmic approach to specifying motion. An animator must then work within the constraints of the algorithm, with a subsequent loss of direct control over an object. Behavioural techniques imbue an object with rules that specify how to behave in response to certain external or internal stimulation. This means even less direct control for an animator. An object produces its own choreography based on the stimuli provided, which in the case of a human character might be in the form of a task such as stand up and walk to the door. Other high level controls might allow the animator to specify the kind of walk, in effect creating a director-oriented system [Thal94] in an analogy with film actors and directors.

In essence, the different classifications presented above are similar – a simple comparison identifies they each define three types of category. They each agree on what a low-level control system offers, namely explicit, descriptive control over every parameter. They disagree about how to constitute the middle level of control abstraction. For instance, Watt and Watt describe particle systems as a medium level technique, whereas Hegron and Arnaldi classify particle systems as behavioural, which perhaps suggests a high-level technique. And, in general, they agree that a task-driven approach is high-level, although the definition of a task may be rather general. In the following descriptions we will attempt to combine the two classification approaches by distinguishing between low, medium and high level by quantity of parameters.

Form and function, similar to Hegron and Arnaldi’s effect and cause, will also be used as demarcation labels.

2.3.1 Low-level techniques

There is general agreement about the kinds of technique that belong in this category. This is the first level of abstraction in the task of motion specification. A set of parameters are identified for an object and these parameters are explicitly changed over time. Keyframing is the simplest technique and draws its inspiration from the hierarchical techniques of traditional hand-drawn animation as developed by Disney. Skilled artists design a sequence by drawing the objects at key instances, thus establishing the feel of the animation, and less-skilled animators fill in the ‘in-between’ frames. For computer animation, the animator manually sets the parameters for each key frame and interpolation techniques replace the in-between artist. The process can be generalised by allowing a range of interpolation technique, e.g. use of spline curves, and by allowing any object parameter to be in-betweened. The term ‘key parameter’ can be used [Watt92] to distinguish these more general approaches from traditional keyframing.

Parameterisation is the most important concern of keyframing. Whatever parameters are chosen, they need to be intuitive to an animator and the manipulation has to produce the desired effect. Whilst for low-level methods the animator has direct access to the parameters, those parameters might not be the right parameters. An oft-quoted example is that of a rotating line, e.g. the arm of a character. Linear interpolation of the endpoints will result in a line that shrinks in length and then grows again. Using angles as the parameterisation gives better results, although using and interpolating Euler angles for general 3D orientations can give rise to other problems due to the order dependence of applying rotations about mutually orthogonal axes.
CHAPTER 2. MODELLING AND ANIMATION

The inbetweening process for shape changing two objects can lead to unpredictable results. Figure 2.29 shows an ‘EZ’ (easy) shape change between two letters, each modelled with the same number of piecewise Bezier curves. The inbetween process interpolates the control points so as to preserve $G^1$ continuity. The inadequacy of simple interpolation between keyframes is revealed in the messy inbetweens. Although individual segments of the piecewise Bezier curve do not cross each other, they rotate and cause the curve to loop back on itself causing an apparent discontinuity. In general, using more keyframes so that the difference between keyframes is less can naively solve the problem. But this means specifying more data. Alternatively, extra constraints must be placed on the inbetween process [Sede93].

![Figure 2.29: Inbetweening using multisegment Bezier curves](image)

The amount of data that must be specified is the biggest problem for the keyframing approach. One short-cut to solve the data input problem is to use motion capture data. This is regularly used in the computer games industry for animating human figures. Libraries of short capture sequences are joined together to create more complex sequences. Figure 2.30 shows frames from a short dance capture. (This will be reused in Chapter 4.) It also shows one of the main problems with the technique. The hands do not join in the clap above the head. This is because the capture data was gathered for a figure with different arm and shoulder proportions. Such problems of adapting data captured for one figure to another differently proportioned figure and joining short capture pieces together is an active area of research [Brud95, Witk95, Rose96].
As noted above we can use the term key parameter to identify a particular parameter to be inbetweened. There is no reason for a complete key frame to exist. Instead individual object parameters can be independently interpolated or controlled over time using spline curves. Such curves can then be edited in real-time to change the animation effect. Separate curves can also be used to control velocity characteristics in a double interpolant method [Stek85]. Kochanek [Koch84] describes a class of splines (based on the Hermite basis) that are particularly appropriate, interpolating specific points and providing parameters to alter tension, bias and continuity. Such parameters give intuitive animation controls. For instance, the tension parameter can tighten or loosen the curve at a particular key position and bias provides the animator with the ability to ‘follow-through’ an action and ‘overshoot’ a key position.

As aforementioned, the use of key parameter values and interpolation techniques is the first level of abstraction in controlling motion. The closeness of the technique to the world numbers or parameters implies a generality of use. All that needs to be done is to choose the right parameters to control. Any of the modelling technique parameters described in section 2.2 can be controlled using these low-level techniques and impressive effects can be achieved. For instance the FFD modelling technique can be controlled by interpolating particular control point configurations. Instead of controlling the many vertices of a complex object represented as polygons, which would require masses of data to control an effective general shape change, spline curves can be used to choreograph the FFD control points in deforming the more complex object. Watt and Watt extend this idea by using factor curves [Watt92], which modify the parameters of a transformation before it is applied to the object. The transformation is modified as a function of the position at which it is applied (based on Barr’s work [Barr84])
and also as a function of the time it is applied. In effect this just means that the amount of
deformation is changed over time. Figure 2.31 illustrates this by increasing the amount of twist
and bend applied to a teapot. Thus the low-level techniques become very powerful when
combined with the correct object control ‘parameterisation’. This can even be extended further
by independently animating the object parameters and the FFD control points [Coqu91].

![Figure 2.31: Increasing the amount of twist and bend applied to a teapot modelled as
parametric patches](image)

2.3.2 Medium-level techniques

As described at the end of the last section, the combination of a low-level animation technique
with an expressive modelling technique, could be considered as more than a low-level control
technique. The fact that a simple curve or even a set of keyframes is still the essential control
method though means that it stays in the low-level control methods. This section considers the
use of algorithmic or procedural techniques to control relevant parameters. A generative
process then converts a few parameters into many parameters in a process of database
amplification. Also included in this section is the use of a hierarchical data structure to enable a
cascading of information from a single input value. An example would be a hierarchical data
structure for a human figure where a single rotation value for a thigh also causes the lower leg
and foot to move.

Particle systems [Reev83] are an example of a medium-level control system. A cloud of points
is created using a stochastic generation process and parameters are available to control such
things as the lifetime of individual particles, production rate for each frame, starting velocity of
each particle and rules for updating particle positions from frame to frame. Variance
parameters make the process stochastic. It is important to note that each particle is independent of all others. No collision detection process is invoked between particles, although one may be used for collisions with the environment. It is the amorphous cloud of points that produces the relevant effect. Example uses included fire [Reev83], spray and foam on wind-driven waves [Peac86, Four86] and waterfalls [Sims90]. The simple particle system can also be extended to increase the domain of applications. A simple cloud or smoke effect can be realised by blending ellipses centred on particle positions. A fast version of this would use 2D ellipses oriented on planes normal to the line of sight. Fields of grass [Reev85] can be generated by recording the path of each particle as a set of connected lines or cylinders.

The animator must work within the constraints of the particle system. This is at a more abstract level that a low-level control technique. The position of individual particles cannot be scripted. Instead only the starting values and the frame-to-frame update rules can be changed. Nonetheless particle systems can produce impressive effects that would be difficult to realise using any low-level control technique. Also of note is that particle systems produce their own representation, albeit one made of individual primitives. In effect they are a modelling technique in the class of procedural primitives.

Cloth animation [Wei86], wind waves on deep water [Watt92] and waves approaching a beach [Peac86; Four86] can be included as medium-level techniques. Each has a highly limited domain of application and a few parameters are used as input to control a generative animation process. For cloth animation constraint points are animated and an underlying analytical model provides a solution to the way in which the cloth will hang from the constraint points. For waves on deep water, sinusoidal waves are added to a surface represented as a mesh of polygons, with parameters to control the propagation direction of the wave, its amplitude, its wavelength and its velocity. Parallel waves can be animated using:

\[ z(x, y, t) = A \cos \left( k \left( x - ct \right) \right) \]

where

A is the amplitude

\( k \) is the wave vector \((u,v,0)\)

c is a velocity vector that lies in the direction of \( k \) and is given by \( c = ck \)

\( c = (g/k)^{1/2} \)

g is acceleration due to gravity

\( k \) is the wave number given by \( k = 2\pi/L \)

\( L \) is the wavelength

\( x \) is the vector \((x,y,0)\)

Figure 2.32a shows the addition of two parallel waves travelling in different directions. A simple flag waving in the wind can also be animated using the same technique. Figure 2.32b shows radial waves which can be defined in a similar way to parallel waves. The main problem with adding sinusoidal waves is that the geometric resolution of the surface representation must be high in order to produce a smooth profile.
Articulated figure animation is included in this section by the fact that a hierarchical data structure is used to intuitively organise the input from the animator and enable cascading of information. Forward kinematics can be used to produce joint angles that drive end effectors (a robotics term meaning the free end in an open chain of links) such as hands and feet. Inverse kinematics can be used to produce joint angles given an end effector position.

Figure 2.33 shows a simple hierarchy for a human figure. Joint angles determine the relative positions of two links. Spline curves can be used to control the variation of angles over time [Watt92] or motion capture data can be applied (as shown in figure 2.30). Classical animation techniques [Thom81] can be easily transferred to a forward kinematics system to produce expressive animation [Lass87] and the hierarchical structure can act as a skeleton for a layered model involving deformable tissue [Chad89, Opal95b, Turn95].
In this section we can also include the use of physically-based approaches to animation, incorporating dynamics into articulated structures (see [Badl91]) or into a linked mesh of deformable elements [Terz87]. As with kinematic approaches to articulated structures, both forward and inverse dynamics can be used. An animator now controls forces, e.g. muscle forces on a skeleton, and the resulting geometric motion is generated, as shown in the example of Figure 2.34. Whilst the motion of, say, bouncing against other objects can be automatically calculated it is less clear how to produce controlled motion using forces [Wil87]. By accurately modelling the physics of an environment, the resulting motion looks natural. But not all animation should be natural. For instance, expressive cartoon animation relies on unnatural motions, such as objects squashing and stretching [Thom81, Lass87] in unbelievable ways. Here a combined kinematic and physically-based approach can be fruitful. For example, Opalach and Maddock [Opal95b] model ‘Disney-effects’, such as squash-and-stretch, using a combination of a kinematically driven articulated skeleton (using dance notation) and attached flesh modelled using soft object primitives. Lennard-Jones forces [Terz89, Heg92] control the interactions of the flesh primitives and parameters can be changed to make the forces weaker and the flesh ‘sloppier’ or stronger and the flesh ‘tauter’. Thus the advantages of automatic flesh effects are combined with direct kinematic control of the skeleton movement. Using a mixed model can also alleviate one of the main problems with physically-based approaches in that they are expensive on CPU time and as a result can tend to be slow. An animator needs interactive feedback when designing a movement. They can also suffer from numerical inaccuracies and limit the possibilities for animator intervention to drive the animation in a particular direction.

Figure 2.34: A physically-based model of a human leg driven by muscle contraction values. (Courtesy of Lee Cooper.)
2.3.3 High-level techniques

In all of the examples of the previous section, a generative process was used to convert a few parameters into many parameters (as shown in Figure 2.28). In general, the animator was still in control of what happened in the world and feedback played little or no part in the resulting motion. The exception was the physically-based approaches where environment forces could result in an object shape change, e.g. a collision process. But such forces were as a result of further specification of the environment rather than just the motion of the object. This brings them closer to the techniques used in this section where the control process is characterised by continual feedback in the process of animation control. In fact, direct animation control becomes even more limited in this section. An object internally transforms itself under the control of a set of rules mimicking a biological process or acts and reacts to an external stimulus. The external stimulus may be biological or could be a high-level task set by an animator such as “walk to the window and look at the garden”. Such a system would be called a task-level animation system [Zelt82].

Plant models expressed by L-systems [Prus90] are an example of an internal transformation process. Such L-systems are based on botanical knowledge, with control provided by a set of production rules. Early work concentrated on the form of the final plant model. This was taken further using developmental models [Prus88, Prus93], simulations of the functions of the tissues of plant buds [Reff88] and the effect of the environment on the growth process [Gree89, Prus94, Mech96]. Other work has looked at the eroding of terrain [Musg89]. In these last two cases the internal transformation process is reacting to external influence. Reynolds work on flocking [Reyn87] is similar. The following rules are used to organise individuals with respect to neighbours, i.e. the ‘internal’ behaviour of the flock:

- Collision avoidance: avoid collisions with nearby flockmates
- Velocity matching: attempt to match velocity with nearby flockmates
- Flock centring: attempt to stay close to nearby flockmates

A migratory point ahead of the flock is animated (possibly using a low-level technique) and the flock behaviour emerges as a result of following the rules and attempting to reach the migratory point. The same ‘self-animating’ technique can be used for animation of other multitudes. Disney used a similar approach in a stampede sequence in the Lion King (1994). Tu and Terzopoulos [Tu94] build on the idea for a simulation of the behaviour of a school of fish, using individual fish controlled using a muscle model [Terz90] and introducing motivational behaviours such as the search for food and avoidance of predators. Individual fish are given a vision sensor and temperature sensor to help govern their behaviour. Boulic et al [Bouil94] and Thalmann et al [Thal96] have also advocated such sensory perception for driving animation.

Tu and Terzopolous’s model reflects the complexity of high-level control methods. In essence, most are task-driven involving governing of lower-level processes and, subsequently, control techniques. A task is set in terms of goals and constraints, goals are split into actions that accomplish something over a period of time and elementary actions make objects move. Such task planning is common in robotics for piloting a mobile robot or controlling a grasping hand.
CHAPTER 2. MODELLING AND ANIMATION

A final example can be used to illustrate the complexity that characterises high-level animation systems. Rijpkema and Girard [Rijp91] describe a knowledge-based system for a grasping human hand. Geometrical, physical and mechanical, topological and dynamic characteristics of the target object, the hand and the environment are identified and a three-phase process is carried out to achieve a goal. In the task initialisation phase an object to grasp is identified and information about the environmental constraints for any motion path are gathered. In the target approach phase the possible set of solutions to the goal is constrained so that a particular grasp can be chosen. For instance, the thumb almost always takes part in a grasping operation, so the thumb can be placed on one face of an object and the fingers on the opposite face. This phase also consists of determining contact surfaces, the grasp position, grasp mode and hand structure, preshaping of the fingers and path generation to the object. Inverse kinematics can be used in the path generation process. In the target approach phase a simple 'primitive' object is used. In the final grasp execution phase the fingers move from their position on the primitive object to the final touching positions on the actual object. The final grasp execution stage is a nice example of making the process more general. A similar approach is used in Motivate (Motion Factory's commercial animation system) where a set of primitive 'handles' are used for a grasping process and a set of primitive shapes are used in a sitting process. For example, a cylinder can be used as a 'handle' within the grasping process and the real object is enclosed within the cylindrical handle.

The general summary we can make is that low-level motion control is tedious, especially if the object has many degrees of freedom. This can be alleviated by using a modelling technique that abstracts in some way, e.g. by surrounding an object with a simpler control mesh as in FFD. But working at a low-level gives an animator explicit artistic control over every world attribute. High-level control makes animation easier for the animator in the sense that it is done at a task level. But this results in less artistic freedom. An analogy with film actors and directors can be made with the director attempting to get the best from an actor of limited skills. At some stages the director may wish to give explicit direction to get an exact effect. At other times actors can be given general directions and allowed to improvise within their limits. Perhaps this is the lesson to be learned for animation systems. They must offer the possibility to interact at each control level.

2.4 Summary and discussion

We have identified three aspects to modelling and animating shape change: representation, modelling techniques and animation techniques. We might consider a model as an implementation of a representation of an abstraction of a physical or imaginary object. There are many studies that focus on the representation techniques for models or on the final form of a model. Rather than present a modelling 'star-system', the techniques used for creating a model are structured into general categories that attempt to include functional aspects of an object. The two general categories of constructive and refinement modelling techniques reflect the way an object is perceived and how it is subsequently animated. In writing about living creatures, Wainwright states that "Even though structure and function may be treated one at a
CHAPTER 2. MODELLING AND ANIMATION

55
time for analytical clarity, an understanding of the whole organism can arise only when structure and function are combined" [Wain88]. We can equally apply this to man-made objects undergoing controlled shape change. The richer and more complex objects inevitably contain a mix of construction and refinement. For instance a constructive articulated structure dressed in polygon primitives may be deformed using free-form deformation to refine the polygon vertices.

Figure 2.35: Animation as a hierarchy of knowledge control (after [Calv94])

In considering animation techniques, we have adopted a looser categorisation based on the amount of parameters an animator controls, with a simple high/medium/low-level hierarchy. Calvert et al [Calv94] offer a pertinent view of this animation-as-a-hierarchy process in terms of knowledge control. As Figure 2.35 illustrates, the three types of knowledge control are explicit, procedural and declarative. Whilst, in general, animation means movement or deformation or both simultaneously, we are instead focussing on the animation of shape change and excluding rigid body animation, although the animation techniques discussed could equally be used for this. Animation of articulated structures are also included, since for a living object, e.g. an animal or human, the articulated structure and the deformation change are not separate. We note that current techniques in computer animation carry a trade-off which results in a contradiction: low-level motion specification can be inordinately tedious (depending on the number of degrees of freedom in the model) but offers full expressive control, whereas high level specification is easier for an artist to understand and use, but generally results in less artistic freedom.

The main motivation of this chapter was to bring together the mixed bag of modelling and motion control techniques produced by the computer graphics community, with the main goal being to lay better foundations for addressing the control of shape change animation. The taxonomy of modelling techniques presented and related to animation techniques in the thesis
gives a general foundation for relating the structural composition of an object to the refinement
techniques used to mould it and alter its shape. Whilst the general efficient and intuitive control
of complex models could be said to remain elusive, the exposition of form and function
inherent in the categorisation of this chapter suggests that the functional aspects of a model in
relation to its structure need to be attended to more. Where this is done, as in the case of
hierarchical structures, complexity is brought under better control.

Figure 2.36 summarises the techniques presented in the chapter. This simple diagram can be
used to make plain comparisons between different animations or to document particular
animations. For instance, Figure 2.37 shows that what little difference there is between flocking
and particle systems is in the level of animation technique used. Figure 2.38a documents the
techniques used to animate a dancing figure built using implicit primitives [Opal95b] and
Figure 2.38b documents the techniques used to create a dancing spoon [Watt92].

Figure 2.36: A simple diagrammatic notation for relating representation, modelling and
animation

In this chapter we have created a basis for considering the general modelling-animation
relationship. In any complex application area, there is an overlap of techniques used to create a
model and animate it. For instance, eyes and ears can be added to a facial model using
construction with primitives, the vertices of the skin representation can be controlled using
refinement of a surface, facial features and their relationships can be parameterised, for
instance blinking eyes, layering can be used to add skin to bone or hair to a head and low level keyframing can be used to animate the shape of a mouth. The next chapter will focus on the specific application area of facial animation.

Figure 2.37: Comparing particle systems with flocking

Figure 2.38: Documenting (a) the dancing figure with physically-based Disney effects from [Opa95b] and (b) the dancing spoon from [Watt92]
Chapter 3.

Facial Animation

Introduction

Computer facial animation is arguably the most difficult application of shape change animation, mainly because of our high expectations. We are acutely attuned to the subtlety of facial expression. For example, consider the difference between a genuine smile and a forced smile. The shortcomings of computer facial animation are perhaps more apparent than other shape changing applications. This chapter will explore the dependency between modelling and animation in computer facial animation.

The typical approach to facial animation is divided into a number of stages. A model of the face is constructed and then varied using conformation parameters. These alter the proportions and relationships between parts of the face. Then a control technique for animating expression is chosen and finally, detailed effects are layered on such as lip animation and realistic skin rendering. Of course the detail of the model and the effects required are characterised by the application area. For example, medical applications need detailed models that support surgical simulation [Larr86, Saga94, Koch96], teleconferencing work needs compression algorithms such as model-based coding [Kane91, Huan95] to cope with bandwidth restrictions, speech applications need lip-synchronisation to sound [Broo92], police forces need to be able to reconstruct individual faces from sketchy descriptions [Eave90], avatars or social agents in interfaces [Cass94, Walk94, Take95] need to be able to converse which requires linguistic abilities [Pela91a, Pela91b] and speech synthesis, the games industry requires real-time performance on limited platform capability, which often means making use of prerendered sequences for talking characters (e.g. Fragile Allegiance – a recent computer game from Gremlin Interactive) and stylistic, talking toys (as in Toy Story [Jaco97]) are needed in the entertainment and advertising industry.

We can divide the discussion of shape change into three sub-disciplines: facial conformation, facial expression and lip motion for speech synthesis. As a USA National Science Foundation report on a Standards for Facial Animation workshop [Pela94] states: “while some control strategies have been developed within each sub-discipline, a coherent framework encompassing all these has not yet evolved.” More will be said about this with regard to the categorisation of
Chapter 3. Facial Animation

Chapter 2 in the discussion at the end of this chapter. First, we will present the three sub-disciplines in the order they are usually addressed in the facial animation community.

Section 3.2, on conformation, discusses techniques for both building a computer model of an existing, real face and for producing a completely new face, such as a computer-generated ‘extra’ or a caricature of a real actor or a face for a talking animal. Conformation control would also cover the ability to add, say, a bump to the forehead of a computer-generated cartoon character. A technique for using FFDs [Sede86] for conformation control will be presented. The section on expressions will focus on the general techniques for controlling facial expression, ignoring the exact application area. The techniques could be used to construct the prerendered sequences for a game or to construct an animated film in off-line mode or, with optimisation, could be adapted for real-time avatars in a social user interface.

As with modelling techniques for more general modelling work, an aim is to make the facial expression techniques independent of the facial representation. However, as with general modelling, the fact that the model is deformed means that to a certain degree the ability to produce expressions is not independent of the model or representation. The nature of the model may limit the amount of deformation and thus the amount of expression control. This will be discussed in more detail in section 3.3.

The section on lip motion will consider the shape changes that characterise movements of the mouth. The most common form of visual speech animation makes use of static visemes and interpolation techniques. This use of a static representation will be questioned. The rendering of detail, such as skin effects and eye construction, will be left until the next chapter. Before discussing each of the three forms of shape change control in face animation, it is worth considering what can be seen in a close-up of a moving face, so as to appreciate the complexity that is still beyond facial animation researchers.

3.1 Impressions of a face

Individuals can be recognised by their face, illness can be portrayed and subtle expressions can convey mood, as well as alter the meaning of spoken words. The face is a marvel of detail and movement. Consider the following observations that were made whilst watching an extreme close-up interview of Harold Pinter on a TV programme called ‘Face to Face’:

‘...lots of flesh under the chin area that wobbles when he’s talking or wrinkles when he moves his chin towards his neck. Most movement occurs in the mouth area. An occasional smile reveals the teeth. The eyes continually blink and move. The head angles around. As the eyebrows raise the forehead wrinkles. He occasionally adjusts his glasses or scratches the side of his head or touches his chin. There are large crease marks from the base edge of the nose to the sides of the mouth, where other creases emanate downwards. His hair does nothing – it is too short to move around. He is balding at the front. His thickish glasses distort the view

---

1 Interview of Harold Pinter by Jeremy Isaacs on ‘Face to Face’, BBC 2, 21 January 1997
through to the side of his face. They also cause some caustics on his skin. The skin seems thick and slightly glowing in places, no doubt due to the heat of the studio lights. There is a lot of flesh on his neck. There is an occasional movement of the bottom of the ear where it is attached to the wobbly flesh of the neck…’

This close-up observation indicates the level of detail that any computer facial animation would have to achieve before convincing a viewer of realism. We are a long way from achieving that. No other shape change animation work has to contend with such intense observational judgement. As an example, consider the facial animation in the recent Pixar/Disney film ‘Toy Story’ (1995). The film features the set of toys owned by a ‘good’ child called Andy, many of which have human characteristics. There are also some human characters in the film, notably the ‘evil’ toy-destroying child called Sid who lives next door to Andy and his family. The facial animation of the toys could be considered to ‘work’ whereas that of the human characters doesn’t. The plastic look of the faces of the toys and the expression of emotion is accepted since it is the impact that is being judged not any physical realism. For the human characters, the faces look plastic and the movement does not appear natural and free-flowing. Our judgement criteria are harsher. Interestingly the same observations apply for the bipedal movement of the humanoid toys in comparison to the humans.

3.2 Facial form

We can consider shape change in two aspects of creating a face: the initial construction of a face and the variation or conformation of an existing model. In the previous chapter we identified a number of modelling techniques that could be used to produce shape change. A number of these techniques have been used in creating and conforming faces, mostly using polygon models.

3.2.1 Producing a model of an existing face

A common approach to producing a polygon model of a face is to use some form of digitisation process [Park72, Berg86, Wate88, Magn90, Pate92]. The most labour intensive approach is to do this manually using a three-dimensional locator device or using stereophotogrammetry. The main advantage of such techniques is that the intelligence of the human digitiser is employed to make sure points are put where they are needed. More recently, laser scanners have begun to automate the process of gathering polygon models of objects (for example, a laser scanner was used to produce the facial models used in the film ‘The Abyss’ [Ande90]), enabling libraries of polygon models to be produced and sold. As with the stereophotogrammetry techniques, the laser scanner is non-invasive (meaning that the object being digitised is not physically touched). This is an advantage over the invasive three-dimensional digitiser where it is difficult to accurately capture a living face as the face may deform during the digitisation process. Whilst laser scanners can produce accurate models of complex objects in a non-invasive way, the amount of data produced is inordinately large for animation purposes. The same is true of techniques which use stereo matching to produce elevation images that are then converted to
CHAPTER 3. FACIAL ANIMATION

polygon models (such as Siebert and Urquhart’s C3D system [Sieb95]). Although excessive
data can be reduced using polygon simplification algorithms [Turk92, Schr92, Hopp93], these
will not necessarily produce the right amount of data in the right areas. For instance, an area
represented by few polygons may subsequently be subject to excessive deformation thus
revealing piecewise silhouettes. Another problem with the laser scanner approaches is that the
resultant data often needs ‘cleaning up’ as the scanning process fails to accurately capture
concavities in the surface. Williams likens this to image processing [Will90] and uses filters to,
for example, smooth over holes in the data.

Of each of the three aforementioned techniques, perhaps the most successful is
stereophotogrammetry, which has been used by a number of researchers to produce facial
models [Park74 and Park96, Wate88, Akim93, Ip96]. Two orthogonal photographs of a human
head are used, one from the front and one from the side giving a profile view. Parke [Park72]
drew a grid onto the real face and photographed this from each viewpoint. Matching and
measuring vertices in each viewpoint then gave x, y, z positions. As noted above an advantage
of using a manual approach is that points can be placed where they are needed. Waters
[Wate88] established some basic guidelines for this process:

• More facets are required where the curvature is high
• Extraneous distortions in the facet structure can be avoided by following general
  latitude/longitude lines and orbital features such as the areas around the mouth and eyes
• A dividing line of vertices between the corner of the mouth and the ear can be used to
  establish vertices affected by jaw rotation
• Crease lines can be used to identify facet divisions – this is especially important in flexing
  of the face
• The approximate symmetry of the face means it is only necessary to model half of the face
• Excellent results can be obtained with only a limited number of polygons. Excluding the
eyes and teeth, Waters states that a respectable half face can be constructed from less than
300 vertices and 500 polygons. Figure 3.1a shows the face vertices and polygons used in a
recent version of Waters’s model2 and Figure 3.1b shows the vertices used in Parke’s
model3. The two models are similar, although in Parke’s model there are few polygons on
the forehead, which would limit wrinkling effects.

An alternative to drawing a grid on the face is to draw the vertices onto two orthogonal
photographs of a face or to adjust the vertices of a predefined topology to match the
photographs. The front view of the face gives two dimensions of the vertex position, the side or
profile view also gives two dimensions and the two sets of measurements taken together give

2 Waters’s model can be downloaded from
http://www.crl.research.digital.com/publications/books/waters/waters_book.html and is described in
[Park96].

3 Parke’s model can be downloaded from
http://www.crl.research.digital.com/publications/books/waters/waters_book.html and is described in
[Park96].
three dimensions. Figure 3.2 shows the topology used in FaceWorks\textsuperscript{4}, which, in essence, can be used to create talking photographs for multimedia applications.

![Figure 3.1: The vertex topology of (a) Waters’s model, and (b) Parke’s model (see [Park96] for a description of both models)](image)

![Figure 3.2: The vertex topology in FaceWorks stretched over three different faces](image)

\textsuperscript{4} FaceWorks is beta facial animation software from DEC’s WWW site: http://www.crl.research.digital.com
CHAPTER 3. FACIAL ANIMATION

A number of researchers have attempted to automate the process of fitting a polygon topology to the results of the digitisation process. Akimoto et al [Akim93] and Ip and Yin [Ip96] address automation of the stereophotogrammetry approach. Lee et al [Lee95] describe how to fit a canonical facial polygon network to data produced from a laser scanner. Akimoto et al and Ip and Yin use similar techniques. First, individual features, such as the chin tip, the mouth, the nose tip and nose bridge are identified in the photographs using template matching. A generic 3D model of a head is then adjusted to match the position and shape of the features. In all these cases, the ease of data gathering is combined with a topology that fits Waters’s guidelines. The main drawback with the digitisation approaches however is that the data is specific to a particular face. Other techniques must be used to construct a completely new face.

3.2.2 Conformation and the creation of a new face

The creation of a new or artificial face, as opposed to digitising an existing face, can be considered in relation to the modelling techniques discussed in chapter 2. A face can be constructed from catalogues of parts, as in the Electronic-Fit or E-Fit approach used by police forces, or it can be produced as a result of refining an existing model using direct surface manipulation or using space deformation techniques. In Chapter 2, the locality of a deformation effect was important for refinement techniques. In facial animation work, the surface manipulation technique is not wholly concerned with locality of effect. If global relationships between features are not maintained whilst a local deformation is performed, then the designer must manually correct them, which could be a tedious process. Thus, in using either constructive or refinement modelling techniques, both the features of the face and their global interrelationships need to be considered.

Many researchers from a range of disciplines have attempted to identify facial shape or conformation guidelines. Patel [Pate92] gives a good review of many of these areas of study, covering the psychology of representation, criminology and identification, facial reconstruction and anthropology, medicine and anatomy and portraiture, sculpture and proportion. Parke and Waters [Park96] list 40 guidelines in sections covering overall shape and proportion, the eyes and eyelids, the nose, the mouth and lips, the ears, male versus female and children. Evidently, there are many perspectives on how to describe a face. Despite this, it is not fully understood how we distinguish the individuality of a face. Perhaps this is one of the reasons why most facial animation work still looks unnatural in comparison to a real face. We can’t accurately describe what makes a face natural.

A number of approaches to producing new faces have been offered by the computer graphics community. We will consider the use of blending, parameterisation, construction from parts and deformation techniques.
3.2.2.1 Blending

Perhaps the simplest technique for creating a new face is to blend a set of existing faces using interpolation. Figure 3.3\(^5\) shows a blend of two faces created by interpolating between vertex positions. The vertex/polygon topology of each face is the same. Magnenat et al [Magn89] describe a technique that can be used to interpolate faces if their topology is different. The problem with the blending process is that it desaturates all features. More control could be offered by allowing the designer to specify different blend amounts for different features of each face. For instance, face x may have been chosen because it had a big nose, and face y because it had big eyes, and any new face should retain both features.

Blending only offers limited control of the final results. It may be able to give a gross approximation quickly (although the final effects are, of course, limited by the number of originals available), but fine control of facial shape is not workable. Instead parameters are needed to give fine local control.

---

\(^5\) Figure 3.3 was produced by extending Parke’s program code, which is described in [Par96], and is available from http://www.crl.research.digital.com/publications/books/waters/waters_book.html. Other similar figures in this chapter were produced by further extending this program.
3.2.2.2 Parameterisation

Parke’s pioneering work in facial animation used direct parameterisation control for both facial conformation and facial expression [Park74, Park96]. Here, we concentrate on the conformation parameters. Expression will be discussed in section 3.3. Table 3.1 lists the conformation parameters used by Parke. As implied in the table, control is imparted by combining rotation, scaling, offsetting and interpolation on these parameters. Figure 3.4 gives examples of faces produced using these parameters.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value Range</th>
<th>Parameters</th>
<th>Value Range</th>
<th>Parameters</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpolation</td>
<td></td>
<td>Scaling</td>
<td></td>
<td>Translation</td>
<td></td>
</tr>
<tr>
<td>Forehead</td>
<td>0.0 - 1.0</td>
<td>Head X Scale</td>
<td>0.5 - 1.5</td>
<td>Chin X Offset</td>
<td>-50.0 - 50.0</td>
</tr>
<tr>
<td>Cheekbone</td>
<td></td>
<td>Head Y Scale</td>
<td></td>
<td>Chin Z Offset</td>
<td></td>
</tr>
<tr>
<td>Cheek Hollow</td>
<td></td>
<td>Head Z Scale</td>
<td></td>
<td>End-of Nose X Offset</td>
<td></td>
</tr>
<tr>
<td>Chin Shape</td>
<td></td>
<td>Chin-to-Mouth</td>
<td></td>
<td>End-of Nose Z Offset</td>
<td></td>
</tr>
<tr>
<td>Neck Shape</td>
<td></td>
<td>Chin-to-Eye</td>
<td></td>
<td>Eyebrow Z Offset</td>
<td></td>
</tr>
<tr>
<td>Eye-to-Forehead</td>
<td></td>
<td>Eye-to-Eye</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eyelid X Scale</td>
<td></td>
<td>Eyelid Z Scale</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jaw Width</td>
<td></td>
<td>Cheek Width</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nose Bridge Width</td>
<td></td>
<td>Nose Nostril Width</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1 Parke’s conformation parameters (after [Park96])

Figure 3.4: A variety of faces produced using Parke’s conformation parameters
CHAPTER 3. FACIAL ANIMATION

A particular problem with Parke’s approach is that the conformation parameters (and the expression parameters) are connected directly to the vertex topology. Particular vertex numbers are identified for each control parameter. It would be a time consuming process to connect the parameters to a different facial vertex topology and the same parameter values may not then achieve the same effects.

Interpolation and extrapolation can be used to produce new faces from existing faces. If a sufficiently large population of faces is blended, e.g. the average of each parameter is calculated and recorded, then the face can be considered as a norm or neutral face. It is a point in an n-dimensional coordinate system or ‘face space’, where n is the number of parameters used to describe the face. A line can be drawn from the neutral face to any other face in face space and interpolation can be used to produce inbetween faces. Figure 3.5 shows the interpolation process, whilst Figure 3.6 shows an extrapolation process applied to the parameters of Parke’s model. The blend face is made up of the 8 surrounding faces, the parameters of which are then extrapolated from the average using a multiplication factor of 2.0.

![Figure 3.5: Interpolation between two faces: From left to right, faces 1, 3, 5, 7 and 9 are given and faces 2, 4, 6 and 8 are interpolated](image)

We might consider the extrapolated faces as caricatures. However, this simple approach does not produce effective caricatures for a number of reasons. First, and foremost, all the parameters are extrapolated equally. Thus all the features are exaggerated. Only those features that are sufficiently different from the average should be exaggerated. Second, because all parameters are extrapolated, the parameters for expression control are also changed, so, for instance, the pupil dilates or the jaw rotates. The extrapolation process would be better if applied to a ‘face space’ focusing on appropriate facial features, e.g. nose and eye size. Dewdney reports on Brennan’s work using just such a feature-based description [Dewd86]. Brennan used 2D line drawings and outlined 39 facial features using 189 key points. For instance, 6 points were used for the left eyebrow. Interestingly, Brennan reported that caricatures were found to be better as a recognisable representation of a face than an ordinary line drawing.

A variation on the theme of using a blend or norm is given by DeCarlo et al [DeCa98]. Anthropometric statistics, based on landmark points on the face, are used to generate the likely or average face measurements in a population. A randomised version of this data is then used to create constraints for fitting a smooth B-spline surface using variational modelling techniques. The approach offers the opportunity to quickly produce the faces of a cast of computer-
generated actors (or ‘extras’) that fit anthropometric statistics. It doesn’t offer a way to produce a face with specific features.

Figure 3.6: Extrapolating a blended face
3.2.2.3 Construction from parts

Two-dimensional Electronic-Fit or E-Fit systems, based on the earlier manual Identikit and Photofit systems, (see [Eave90] and [Wu94] for examples of systems) are based on the idea of construction using parts or features. The idea is to use a catalogue of features, such as different shaped noses, to construct a whole face. In the earlier Photofit system strips of cardboard containing individual features are put together in a jigsaw style. In the computer based E-Fit system the features are part of a database and the resultant on-screen image can be easily edited. Whilst such systems can be used to focus on the size or shape of individual features, they seem to lose sight of the ‘whole’. A description of the form “he had an evil-looking face” is not really captured by a system of parts. Instead such systems appear to need an experienced artist/user who interprets a witness’s initial description before using E-Fit to construct a starting face [Bard94]. This is then edited in a session involving both the artist/user and the witness. Subsequent variation involves changing or altering individual pieces, which are blended together with the other pieces to form a whole face. No relationships between major parts are identified though and it is possible that one part could overlap another.

An interesting angle on the E-Fit process is described by Webb and Maddock [Webb95]. Parameters used to describe individual pieces and the relationships between them are recorded as a ‘gene sequence’. A genetic algorithm is then used to produce new faces. Peng and Hayes [Peng95] use a similar approach to produce facial expressions. In Webb and Maddock’s work, the witness is presented with six faces at each stage of the iterative process. The witness picks the face that is ‘closer’ to the one they are thinking of and this then steers subsequent evolution. The problem with this technique is that the steering process can lock the evolution process into moving in the wrong direction. Also, it doesn’t replace the coaxing from and interaction with an experienced police artist, who builds up experience about facial feature relationships. It may however be possible to use the work of DeCarlo et al [DeCa98] to simulate some of this experience and build it into an automated system.

A final point on the E-Fit approach is that a 3D version does not seem to have been developed. Such a system would allow the lighting conditions that the face was seen in to be simulated, although perhaps this would be too close to reality for a witness. Instead of building up a whole face, the construction from parts approach appears to be reserved for adding details such as eyes, ears and hair to a head modelled using other techniques.

3.2.2.4 Deformation or surface refinement

As noted in Chapter 2, refinement techniques in general start from a whole and refine this. This can be done by direct surface manipulation, by identifying and parameterising features, or by general space deformation. We have seen how direct surface manipulation techniques are used in stereophotogrammetry. However, without the aid of a reference image the movement of individual surface points to effect a new face is tedious and prone to production of local surface curvature defects. Instead groups of points need to be moved. This is the same issue discussed
in chapter 2. Techniques are needed to select and transform groups of points. Magnenat-Thalmann et al [Magn89] identify 5 ways of selecting a region:

- Selecting specific vertices;
- Selecting vertices within a box;
- Selecting vertices within a pie slice of a cylinder;
- Selecting all vertices with a given hue value;
- Using set operations on regions selected using the previous methods.

They also identify 4 methods of transforming the points:

- Percentage of a vertex – each vertex is moved towards a reference vertex according to a specified percentage of the distance between both vertices;
- Scale according to a plane – this is similar to the previous method, except a reference plane is used. If the group of points were translated to lie on the plane this would produce a flattening operation [Cobb84];
- Guided translation – a group of points are translated as a whole based on the direction between two vertices and a percentage of the distance between them. The discontinuities introduced between the region and the surrounding area make this of questionable use when applied to a face;
- Variable translation – this is similar to the previous method except that points move by varying amounts depending on their distance to the centre of the region. It is similar to a point based warp using a decay function [Cobb84, Alla89].

The use of a variable translation, or point-based warp function, appears to be the most useful function, since continuity between a region and its surroundings can be attended to by choosing an appropriate decay function. As an example of its use, the tip of the nose can be pulled and the surrounding area is warped accordingly. DiPaola [DiPa91] provides another example of a point-based warp. The difference is that this is applied in conjunction with a surrounding elliptical volume. He used 12 warp functions at a time to produce local effects such as muscle bulges, brow furrows and double chins as well as more comical effects such as enormous chins and craniums.

In both cases, the technique could be improved by using either a skeleton warp or a region warp [Cobb84] instead of a point-based warp. For instance, a line can be drawn along the length of the nose and used to warp the whole nose, or the nose could be outlined and lifted away from the face as a whole with the surrounding region being appropriately warped. The use of such feature-based warping is common in 2D morphing [Betl92]. In 3D, the technique generalises to an axis-based deformation, as discussed in Chapter 2. Singh and Fiume [Sing98] present such a technique based on what they call wires. A wire placed on the face, together with a surrounding warp function, can be used to locally deform areas of the face. Thus wires can surround features and alter their size. A set or parallel wires can be used to produce creases or wrinkles on the forehead of a face. Wires can also be used to outline features that shouldn’t be affected by any deformation. In effect these are constraints that prevent deformation and are common in general surface modelling techniques, but less so in general facial animation. For instance in
Waters’s muscle model [Wate87], it is easy to stretch the eye opening to unfeasible limits (see section 3.3).

In the techniques discussed so far, a single face level is implicitly assumed for deformation control. In effect the ‘skin’ of the face is the model. There is no underlying bone or fat structure. Patel [Pate92] explicitly offers three levels of control – global, regional and local – over both the skull and skin layer. At the global level, control is offered over the general proportions of both the skull and skin layer using 3D scaling. This gives variation in skin thickness, albeit a rather ad hoc variation. Skin thickness obviously has an important role to play in accurately modelling the movements of the face. In forensic pathology, its role is important in building up facial models from skull remains [Tyr97]. Figure 3.7 gives an example. However, few researchers have looked at its role in facial animation (with Lee et al [Lee95] being the notable exception).

![Figure 3.7: Forensic facial reconstruction: (a) laser-scanned data for a skull (40x40 polygon mesh VRML 2.0 file); (b) reconstructed ‘emaciated’ face; (c) reconstructed ‘obese’ face. (Data courtesy of Martin Evison, Dept. of Forensic Pathology, Univ. Sheffield.)](image)

In Patel’s work, regional control is achieved by dividing both the skull and the skin layer into three regions. For the skull this consists of the upper skull, the lower skull and the middle skull. These are delineated by horizontal bars which can be interactively adjusted to stretch the proportions in each region. Similar parameters are defined in Parke’s work [Park74, Park96], but Parke directly connects the parameters to particular vertices, whereas Patel’s approach is more general being applicable to any facial topology. Waters [Wate88] also experiments with the use of horizontal divisions, using horizontal divisions at the top of the head, the base of the
CHAPTER 3. FACIAL ANIMATION

hair line, the top of the eye, the base of the eye, the top of the nose flange, the base of the nose, the midline of the lip, the base of the lip and the base of the chin. In Patel’s work, vertical divisions can be combined with the horizontal divisions to provide control over head width and profile. For local control of the skull, Patel provides the ability to scale, translate and rotate particular bones. For the face, features such as the eyebrows, eyes, nose, lips and neck are altered in a similar way to the bones. It is not clear, however, if the local process merely serves to alter specific vertex numbers as a group and thus is connected to a particular topology in a similar way to Parke’s direct parameterisation approach.

In both DiPaola’s work and Patel’s work, general space deformation has been used rather than direct surface manipulation. DiPaola suffers from the limitation of elliptical warps, whereas Patel appears to resort to direct connection to particular vertices for local feature control. Whilst Singh and Fiume’s wires can be used to focus on warping specific features, they require a lot of low-level manipulation of wire control points and do not give full feature containment. FFDs [Sede86] offer an alternative approach. They provide the general deformation of DiPaola’s skeletal warps, they can be used at a global and local level independently of representation, they can be easily parameterised and they offer an intuitive visualisation of containment. All these were identified in Chapter 2 as important attributes for a modelling technique.

Figure 3.8b<sup>6</sup> shows the use of a tricubic Bezier hyperpatch (FFD mesh – see formulation in Chapter 2) in a global deformation of the face. The control points to the right hand side are moved as a unit away from the face, which is intuitively stretched in the same direction. Figure 3.8a shows the same movement in a simple linear deformation (similar to that used in the work of both Patel and Waters), where a vertex’s new position p’ is given by:

\[ p' = p_0 + u(p_1 - p_0) \]

where \( u = (p-p_0)/(p_1-p_0) \), p is the original position, \( p_0 \) is the x coordinate of the cube face on the left, \( p_1 \) is the x coordinate of the cube face on the right and \( p_1' \) is the new x coordinate of the cube face on the right. Figure 3.9 compares the two again for a local deformation of the nose. The simple comparison between the two serves to illustrate the expected extra controllability of the FFD mesh.

Figure 3.10 shows a tricubic Bezier hyperpatch being used to produce two further nose shapes. In Figure 3.10b the nose is turned up a little at the front and in Figure 3.10c it is moulded into a hooked shape. Figure 3.11 shows the use of an FFD mesh to pinch the face inwards towards the chin area.

---

<sup>6</sup> Figure 3.8 was produced by extending Waters’s program code, which is described in [Park96], and is available from http://www.crl.research.digital.com/publications/books/waters/waters_book.html. Other similar figures in this chapter were produced by further extending this program.
Figure 3.8: A global deformation used to stretch the face to one side: (a) using a simple linear deformation; (b) using a tricubic Bezier hyperpatch

Figure 3.9: A local deformation to produce an elongated nose: (a) using a simple linear deformation; (b) using a tricubic Bezier hyperpatch
Figure 3.10: Two further noses produced by moving the control points of a tricubic Bezier hyperpatch: (a) undeformed nose; (b) a turned-up nose; (c) a hooked nose.

Figure 3.11: Pinching the face inwards towards the chin
Figure 3.12: The problem of FFD overlap

The use of FFDs to contain and deform a whole object is clear-cut. The contained object intuitively mimics the movement of the FFD mesh control vertices. Their use for local deformation of parts of an object is not as straightforward. Figure 3.12 illustrates this in 2D. In Figure 3.12a, the FFD block partly covers the surface, which will deform intuitively in response to moving a control point away from the surface, as shown in the noses of Figure 3.10. In Figure 3.12b, the whole FFD block covers the surface. If a control point is moved, the contained area deforms accordingly, but the external area does not move. This will introduce a discontinuity in the surface and, with a large movement of the control points, will cause the contained area to overlay the external area. A solution is to use a Rational FFD (RFFD) [Kalr92, Kalr93]. These extend the general FFD technique by adding weights for each control point of the (tricubic) FFD parallelepiped grid. By using RFFDs, the general space
deformation can be controlled by moving the control points or by adjusting the weights at control points. With weights of unity the formulation is the same as standard FFDs. Using the weights, the feature can only be adjusted within the size of the RFFD block. Discontinuities will emerge at the boundaries of the RFFD block, but since the control points have not moved no overlap of surrounding features will occur. Scaling the RFFD to allow large modifications would mean that other structures would be enclosed and deformed. Another solution is to use multiple connected FFD blocks, only allowing the control points in the centre block to be adjusted. Continuity can then be maintained by adjusting relevant control points across FFD block boundaries. However this could lead to visual clutter in editing, although this could be solved by showing the inner FFD block and setting the surrounding blocks to be invisible. Figure 3.12c suggests an alternative solution. A decay function, not unlike that used with implicit surfaces, could be used to move external points by an amount dependent on their distance from the original FFD block and on the direction of movement of the control point. Thus as long as the control point is not moved beyond the limits of the decay function, no overlapping will occur.

Figures 3.8 to 3.11 were produced by changing individual control points. As with any other manual approach this is time consuming. In the example noses in Figure 3.10, the deformations could have been accomplished more easily by combining an axis-based deformation (see Chapter 2) with the FFD control mesh. An axis bend operation (e.g. [Barr84]) could be used to deform the FFD mesh which would in turn deform the contained surface points. Individual control points could then be adjusted for finer control. An axis bend could be used on its own, without an FFD, but there is no clear visualisation of containment of the area surrounding the axis as there is when the FFD is used. Another improvement that could also be made is to use an irregular FFD grid [MacC96] so as to more closely fit the initial facial feature, although this may have the effect of adding extra control point complexity.

The main advantage of using FFDs is their independence from the surface representation. This is shown in Figure 3.13, by applying FFD meshes to Parke’s face model to produce the same nose deformations as shown in Figures 3.9 and 3.10, which are based on Waters’s face model. The same effects cannot be produced using Parke’s parameters. As a further example of the independence from model representation, we note that Waters’s model supports a muscle-based approach to expression control which is achieved by giving end points for muscle vectors in relation to the face (see next section). The FFD mesh is deforming space, so the end points of the relevant muscles can be deformed also. Thus the same muscle control system can be used on the newly deformed face (as shown in the next section).

As with Forsey and Bartels’ work [Fors88] on hierarchical B-spline refinement, the FFD meshes could be used in a hierarchical fashion. An FFD mesh surrounding the entire face can deform the face and then a feature-based FFD mesh can be altered to give local control. Also, the contained FFD mesh could itself be deformed by the FFD mesh at a higher level in the hierarchy. Finally, the FFD mesh could be parameterised so that groups of control points can be moved as a whole. For instance, parameters might be defined for the width of opposite planes.
of the block. This would support a pinching operation. The advantage of this form of parameterisation is that it is still independent of surface representation.

![Figure 3.13: FFDs applied to Parke’s model](image)

3.3 Facial expression

There are five fundamental approaches to control of facial expression: keyframe interpolation, direct parameterisation, pseudomuscle-based, muscle-based and performance-driven animation [Park91a, Park91b]. In essence, all the techniques are similar. They all involve some form of parameterisation and animation is achieved by controlling the parameterisation. For keyframe or key-pose interpolation, perhaps the widest used technique, the parameters define the facial geometry. It is a low-level animation control technique and is very labour intensive as, in general, many parameters must be specified per key pose. The direct parameterisation approach [Park74, Park96] uses parameters to define relationships between parts of the face. These serve both conformation and expression control purposes. Spline curves can be used to vary the parameters over time, but the technique is still low-level.

The pseudomuscle-based techniques use low-level animation control techniques in conjunction with a geometric modelling technique to simulate the results of the action of a muscle. The parameters are the pseudomuscle values and the regions of the face acted upon. Magnenat et al [Magn88, Magn90] present abstract muscle action (AMA) procedures, not dissimilar to the direct parameterisation approach, which are hard-wired to particular regions of the face or particular features of the face, e.g. the eyes. Kalra [Kalr93] uses minimum perceptible actions (MPAs) in a similar way, but uses rational free-form deformations (RFFDs) to deform the space containing a group of points in a general space deformation technique. Both are low-level animation control techniques. They also implement a layer above to control expressions (groups of pseudomuscles). At this level they can be regarded as medium level control techniques because they proceduralise the results of the action of muscles through the idea of expression control.
CHAPTER 3. FACIAL ANIMATION

The muscle-based approach involves simulating the muscle itself. Animation is achieved by varying the simulated muscle ‘contraction’ values which cause regions of the skin to deform. Waters’s early work [Wate87] involved a geometric simulation of a muscle and is equivalent to a surface manipulation technique with certain surface properties simulated in the deformation, e.g. elasticity of the surface. His subsequent work with Terzopoulos [Terz91] used a physically-based model to simulate properties of the fat and skin tissue. The parameters in both cases are the tissue simulation values and the contraction values for the muscles. Platt and Badler [Plat81] also used a physically-based simulation with the skin modelled as a ‘tension net’ and deformed by applying forces through ‘muscle fibres’. All these techniques are medium level techniques as secondary shape change actions result from the proceduralisation of the muscle action and the skin action.

Performance-driven animation [Will90] is an example of using a high-level input device to control whatever parameterisation of the simulated face is chosen. The input device could be a physical device such as a DataGlove – a device worn on the hand that picks up the movement of individual fingers turning them into numerical input – or could involve video-based motion tracking of the face [Essa95].

All the approaches focus on the algorithms and techniques to drive the face model. Relatively little work has been done on establishing what is a necessary set of controls or what is the right parameterisation for achieving control and for giving an intuitive interface. This reflects the general shape change problems outlined in the last chapter. Different applications produce different ad hoc solutions. The main common theme that has emerged in more recent work is the separation between conformance and expression control. It should be possible to apply a control technique to a multitude of different shaped faces. Metrics for measuring the success of facial animation techniques will be discussed in a later section. First, the techniques will be described in more detail, together with examples.

3.3.1 Key-Pose Interpolation for Producing Expression

Key-Pose interpolation is based on traditional keyframe interpolation as used in the cartoon animation industry. Parke [Park72] was the first to use this technique for facial expression animation. Geometric data (such as the vertex positions of a polygon representation) is stored for the face in two poses and intermediate poses are then generated by interpolation of the data (vertex positions). The single control parameter is the interpolation coefficient which may be varied linearly or using some other function such as a cosine curve or a general spline curve to introduce acceleration and deceleration (ease-out and ease-in) effects. The technique is fast, involving simple interpolation calculations, and a single facial topology can capture a wide range of faces and expressions. For instance, Parke proved that a single topology of 300 polygons defined by about 400 vertices could allow representation of many expressions and reasonable inbetween transitions [Park74, Park96].

To produce a smoothly changing animation sequence, multiple key poses can be interpolated using the spline curve techniques described in chapter 2. Figure 3.14 shows interpolation
between the vertex positions in two key poses. Also, instead of just interpolating between two key poses, four key poses can be blended by using bilinear expression interpolation (see Figure 3.15). More key poses can be blended using n-dimensional expression interpolation but the usefulness of this for expression control is questionable as it is not particularly intuitive [Park96]. Figure 3.16 shows how extrapolation can be used to produce an exaggerated key pose by ‘over interpolating’ a key pose from a neutral start pose. The smile is ‘overshot’ or exaggerated using a final interpolation value of \( t = 3 \) in the linear interpolation \( (1-t) \mathbf{p}_1 + t \mathbf{p}_2 \) where \( \mathbf{p}_1 \) and \( \mathbf{p}_2 \) are the vertex positions of frames 1 (the neutral position) and 3 (the stored ‘smile’ expression) in the sequence. Using spline curve parameters such as tension and bias to control interpolation of a sequence of poses can produce cartoon-like effects in the animation sequence, e.g. exaggeration of a pose by overshooting using a bias parameter. An example is the short film ‘Tony de Peltrie’ [Berg86] where Kochanek’s spline curve [Koch84] incorporating tension, continuity and bias parameters is used.

![Figure 3.14: Interpolation of vertex positions between the key poses of fear and disgust](image)

The main limitation of the key pose approach is that the range of expression control is directly related to the number of key expressions and their disparity. Generating a large number of key poses is a time-consuming process, although once they are stored in a library the poses can then be reused and easily added to over time. But there appear to be no limits on the range of expressions that might be needed for a character. Bergeron et al [Berg86] found that 20 expressions satisfied their requirements but the control of a caricature cannot be equated to the control of a believable, convincing human actor. Although the key pose approach has limitations in expression control, it is nevertheless commonly used in mouth shape animation for speech. Here a limited number of visemes are used to produce all the relevant mouth shapes (see section 3.4). Another limitation of the key pose technique would appear to be that the topologies of the key poses need to be the same. Magnenat-Thalmann et al [Magn89] show how to overcome this problem by converting two arbitrary topologies into m-by-n rectangular grid topologies for subsequent interpolation.
Figure 3.15: Bilinear interpolation of vertex positions between four key poses

Figure 3.16: Extrapolation of vertex positions for a smile
3.3.2 Direct parameterisation for control of expression

Parke [Park74, Park82, Park96] developed the direct parameterisation approach in response to the problems associated with key-pose interpolation. A fairly small set of parameters are used to control both facial expression and facial conformation. Examples for expression include eyelid opening, eyebrow arch, eyebrow separation, jaw rotation, mouth width, mouth expression, upper-lip position, mouth corner position and eye gaze. As with conformation (described in section 3.2.2.2 above) control is imparted by combining rotation, scaling, offsetting and interpolation on these parameters. Figure 3.17 shows examples from Parke’s model.

Figure 3.17: A variety of facial expressions produced by varying Parke’s expression parameters

The general problems with the direct parameterisation approach are the same as that of all shape control: what is a good set of parameters to give a wide range of effects? how is that set of parameters controlled to give the required effects? are the parameters intuitive? The idea of confining control to individual primitive operations such as eyebrow arch initially seems quite intuitive. The shape of a part of the model is controlled in direct analogy to the real world counterpart. It seems natural to say arch the eyebrow by a certain amount. This means that only a local area of the model is changed and thus less data needs to be specified than in the key-
CHAPTER 3. FACIAL ANIMATION

pose approach where the data for all vertices in the key pose is needed. But such actions are not local. Raising the eyebrow may be accompanied by a wrinkling of the forehead, or a lifting of the cheek area, or an opening of the pupil. Also the reason for raising the eyebrow may be in response to a look of surprise. The parameter for one part of the model is linked to parameters for other parts of the model. This means that parameters need to be interrelated and subsequently coordinated.

It is clear how to control a single parameter. It is not clear how to control a group of parameters and what interference is caused by manipulating parameters in combination. Parke’s work requires the parameters to be applied in a particular order [Park96]. If the same effect can be produced by different combinations of parameters then the set of parameters is not complete. This concern does not seem well addressed and most parameters seem to be chosen by observation rather than by any formalised process. In contrast, the parameter sets developed in the muscle simulation approaches base their parameter sets on anatomy. In practice the problem is usually simplified to one of how to generate a good set of parameters that will allow the required control for the particular application.

A further problem with the direct parameterisation approach is that the parameters are directly related to the vertex topology of the facial model. In Parke’s model, particular vertex numbers are identified for each control parameter. It would be a time consuming process to connect the parameters to a different facial vertex topology and the same parameter values may not then achieve the same effects.

Parke’s model provides a high proportion of parameters for the eye and mouth areas since these are arguably the primary areas of importance in conveying expression. This is particularly the case in cartoon animation where the eyes are deemed particularly important [Thom81]. The particular reason for the high proportion of parameters in the mouth area is that one of the goals of Parke’s model was to support speech animation. Many researchers have subsequently built on this work [Pear86, Cohe90, Pela91a, Cohe93]. The direct parameterisation technique has also been successfully used in a production situation by DiPaola [DiPa89] who has produced a model incorporating approximately 80 parameters for controlling the face, the head and the facial hair. Whilst 80 parameters gives control over many features, the possibilities for interaction between parameters and achieving the same effect with different sets of parameters increases.

The main criticism that can be targeted at the direct parameterisation approach is that it fails to take into account the causes of a shape change in the face. Instead, the parameters are based on observation of movement, e.g. the eyebrow raises so a parameter is needed to control eyebrow arch, etc. But, as already noted, this local movement is accompanied by secondary actions or caused by other actions. In concentrating on local parameter change and localness of operators, we lose sight of the global interrelated movement of the face. Building a hierarchy upon the local operators can in some ways give control of interrelated parameters, but it is not clear what those interrelationships should be. This bottom-up approach does not necessarily capture the right parameters to provide the right interrelationships at higher levels. The next two
approaches attempt to address these problems by looking at the main active cause of facial movement, namely the contractions of muscles.

3.3.3 Pseudomuscle-based control of expression

The movement of the face and subsequent shape changes are caused by the complex interaction of forces acting on the different tissue layers of the face. Fat wobbles as the head moves, skin is stretched and creased over bone, muscles and fat layers, and muscles are contracted in mastication, in talking and in creating expressions. The aim of pseudomuscle-based approaches is not to exactly simulate these complex interactions. Instead areas of the face are controlled by simulating the actions of muscles. Magnenat-Thalmann et al [Magn88] present abstract muscle action (AMA) procedures and Kalra [Kalra92, Kalra93] presents rational free-form deformations (RFFDs) to simulate muscle actions.

Magenenat-Thalmann et al use AMA procedures to deform the surface shape of the face. An AMA exists because of the actions it performs. It has no volume. The face is purely a surface and has no underlying layers. The AMA procedures also assume that the face is symmetric [Magn90]. The most important AMA procedures are listed in table 3.2.

As noted, each AMA procedure parameter is related to a particular area of the face. For example, the procedure moving the eyes is controlled by parameters 23.26. The procedure encodes the eye using a centre position and a maximum allowable vertical and horizontal angle. A command such as MOVE_RIGHT_EYE_VERTICAL 0.4 moves the eye up by 0.4*maximum_vertical_angle. Individual ‘actors’ can have different maximum vertical angles for eye movement. Thus an ‘actor’ is characterised by unique face, facial regions and parameter

<table>
<thead>
<tr>
<th>Number</th>
<th>AMA procedure</th>
<th>Range for the corresponding facial parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VERTICAL_JAW</td>
<td>0 .. 1</td>
</tr>
<tr>
<td>2</td>
<td>CLOSE_UPPER_LIP</td>
<td>0 .. 1</td>
</tr>
<tr>
<td>3</td>
<td>CLOSE_LOWER_LIP</td>
<td>0 .. 1</td>
</tr>
<tr>
<td>4</td>
<td>COMPRESSED_LIP</td>
<td>0 .. 1</td>
</tr>
<tr>
<td>6</td>
<td>MOUTH_BEAR</td>
<td>0 .. 1</td>
</tr>
<tr>
<td>7</td>
<td>RIGHT_EYELID</td>
<td>-1 .. 1</td>
</tr>
<tr>
<td>8</td>
<td>LEFT_EYELID</td>
<td>-1 .. 1</td>
</tr>
<tr>
<td>9</td>
<td>LEFT_LIP_RAISER</td>
<td>0 .. 1</td>
</tr>
<tr>
<td>10</td>
<td>RIGHT_LIP_RAISER</td>
<td>0 .. 1</td>
</tr>
<tr>
<td>11</td>
<td>LEFT_ZYGOMATIC</td>
<td>0 .. 1</td>
</tr>
<tr>
<td>12</td>
<td>RIGHT_ZYGOMATIC</td>
<td>0 .. 1</td>
</tr>
<tr>
<td>23</td>
<td>MOVE_RIGHT_EYE_HORIZONTAL</td>
<td>-1 .. 1</td>
</tr>
<tr>
<td>24</td>
<td>MOVE_RIGHT_EYE_VERTICAL</td>
<td>-1 .. 1</td>
</tr>
<tr>
<td>25</td>
<td>MOVE_LEFT_EYE_HORIZONTAL</td>
<td>-1 .. 1</td>
</tr>
<tr>
<td>26</td>
<td>MOVE_LEFT_EYE_VERTICAL</td>
<td>-1 .. 1</td>
</tr>
<tr>
<td>27</td>
<td>RIGHT_RISORIUS</td>
<td>0 .. 1</td>
</tr>
<tr>
<td>28</td>
<td>LEFT_RISORIUS</td>
<td>0 .. 1</td>
</tr>
<tr>
<td>29</td>
<td>MOVE_RIGHT_EYEBROW</td>
<td>-1 .. 1</td>
</tr>
<tr>
<td>30</td>
<td>MOVE_LEFT_EYEBROW</td>
<td>-1 .. 1</td>
</tr>
</tbody>
</table>

Table 3.2 The most important AMA procedures [Magn90]
values. We label AMAs as pseudomuscle-based because they do not model the muscles themselves. Instead they parameterise observed effects on the face. The approach is not dissimilar to the direct parameterisation approaches, even through some of the parameters are based on the causes of actions, not purely on observation of facial features.

AMA procedures are the lowest level in a three-level system called the Human Factory System developed by Magnenat *et al* [Magn90]. AMA procedures are grouped into higher level expressions to form phonemes (speech expressions) and emotions (e.g. smiling). For example a smile might be 30% of parameter A and 50% of parameter B. Level three is the script level. Individual tracks of the script control individual AMA procedures. Splines with bias, tension and continuity parameters are used to interpolate the values stored at specific time instants on the tracks. One point that must be noted about the AMA procedure approach is that, as with the parameters in the direct parameterisation approach, the AMA procedures are not independent. Thus, the ordering of actions is important.

Kalra [Kalr92, Kalr93], working with the Thalmans, uses RFFDs to shape change regions of the face. As noted in the previous section on conformation (section 3.2.2.4), these extend the general FFD technique [Sede86] by adding weights for each control point of the (tricubic) FFD parallelepiped grid. By using RFFDs, the general space deformation can be controlled by moving the control points or by adjusting the weights at control points. Kalra defines a set of regular grid control point movements that deform space in a particular way. Figure 3.18 shows these actions in 2D. By combining actions, the effect of particular muscles can be simulated on regions of the face.

![Figure 3.18: Kalra’s RFFD actions (after [Kalr93])](image-url)
A number of questions remain unanswered in Kalra’s work. No comment is made on the dependence between the orientation of the surface with respect to the RFFD mesh and the relative size of the RFFD mesh. No comment is made on why RFFDs are better than FFDs and when weights can be used rather than just moving control points.

In Magnenat-Thalman et al’s work the movement of groups of vertices in relation to AMA procedures is directly connected in a particular way. The use of RFFDs gives a more general technique in that regions of space are deformed not particular vertices. Kalra’s work fits neatly into the general Human Factory System developed by Magnenat-Thalman et al. Shape change is achieved using RFFDs, RFFDs are controlled by minimum perceptible actions (MPAs), not unlike Magnenat-Thalman et al’s AMA procedures, and finally MPAs are grouped to provide phonemes and expressions and then scripted from a higher level. In other words, a hierarchical approach is used to deal with the complexity of facial animation.

### 3.3.4 Muscle-based control of expression

The muscle-based approaches attempt to simulate the individual muscles themselves. Animation is achieved by varying the simulated muscle ‘contraction’ values which cause regions of the skin to deform. The problem is that accurate detail about the actions of individual facial muscles to produce expressions is not available. Work by Duchenne in the 1860s (referenced in [Park96]) on freshly guillotined heads looked at the contortions produced by applying electrical currents, but this only gives gross information about areas of the face affected by certain muscles. The work that has had the most significant impact on subsequent computer graphics work is the Facial Action Coding System (FACS) developed by Ekman and Friesen [Ekma78]. Here, individual muscles, or small groups of muscles, are described as Action Units (AUs) that cause facial activity or deformation of areas of the face. FACS is based on the anatomy of the face with 58 AUs defined. For example AU1 is the inner brow raiser and is based on contracting the inner frontalis muscle on the face. An interesting aspect of this work is that FACS is a system where emotional state is observed from visible facial distortion. The mechanics are not measured. Only visually distinguishable facial movements are described.

Waters [Wat87] based his work on FACS. He used a geometric distortion function, simulating a muscle, to move regions of the face. The idea is to give a general control technique that is independent of the topological facial representation. No attempt is made to accurately model the complex anatomical and mechanical characteristics of facial tissue. Instead the muscle model emulates the primary characteristics using a few parameters. With few parameters, independence from facial representation and emulation of the physical processes that drive the face, the technique initially appears very promising.

Based on a study of the general anatomy of a face, Waters presents three types of muscle: linear muscles that pull, sphincter muscles that squeeze and sheet muscles that act like a set of parallel muscle fibres that pull. The muscles are modelled as vectors. For a linear muscle, the direction of pull is towards the point of attachment on the bone and the magnitude is the amount of contraction. This varies from zero at the point of attachment to the bone and is at its maximum
at the point of attachment in the skin. Beyond the point of attachment in the skin, the magnitude drops to zero. A zone of influence is defined around the vector and the skin displacement is approximated by multiplying the length of the muscle fibre by the cosine of the angle of attachment of the muscle fibre to the tendon or surface tissue [Gray73].

Figure 3.19 shows the geometry of the linear muscle model. To calculate the displacement of a node \( p \) (e.g. a polygon vertex) to \( p' \), the following expression is used:

\[
a = \cos \left( \frac{a2}{a1/2} \frac{\pi}{2} \right)
\]

where

\[
p' = p + ak r \frac{pv_1}{\|pv_1\|}
\]

\[
D = \|v_i - p\|
\]

\[
r = \begin{cases} 
\cos \left( \frac{1 - D}{R_s} \frac{\pi}{2} \right) & \text{for } p \text{ inside sector } (v_1p_p p_mv_1) \\
\cos \left( \frac{D - R_s}{R_f - R_s} \frac{\pi}{2} \right) & \text{for } p \text{ inside sector } (p_p p_r p_s p_m) 
\end{cases}
\]

and \( k \) is a fixed constant representing the elasticity of the skin. Figure 3.20 shows the effects of changing the contraction factor of the muscle.

\[\text{Figure 3.19: The geometry of a linear muscle}\]
The sphincter muscle is modelled as an ellipsoid and contracts around an imaginary central point. When applied to a mesh, the surface “is drawn together like the tightening of material at the top of a string bag” [Park96]. Figure 3.21a shows the geometry of the sphincter muscle. $l_e$ is the semimajor axis and $l_s$ is the semiminor axis of the ellipsoid centred at point $c$. To compute the displacement of point $p$ to $p'$, the following expression is used:

$$f = 1 - \frac{\sqrt{l_e^2 p_x^2 + l_s^2 p_y^2}}{l_e l_s}$$

At the centre of the ellipse $f = 1$ and at the rim $f = 0$. $\sin(f \pi)$ can be used to displace each vertex within the ellipse. Figure 3.21b shows the sphincter muscle in action on a simple mesh. This has the effect of moving all the area within the ellipse. An alternative is to use two ellipses, one contained within the other and only deform vertices that lie in the region between the two
ellipses. Another alternative is to use the inner ellipse to demarcate the boundary between two areas as with the linear muscle.

Figure 3.21: (a) The geometry of a sphincter muscle and (b) its action on a simple mesh

The sheet muscle can be described as a series of almost-parallel fibres spread over an area. An example is the frontalis major muscle which lies on the forehead and is primarily involved in raising the eyebrows. Figure 3.22a illustrates the model of a sheet muscle.

Figure 3.22: (a) The geometry of a sheet muscle and (b) its behaviour

Waters [Wate88] gives the displacement d of p to p' as:

\[
d = \begin{cases} 
\cos \left( 1 - \frac{L}{R_f} \right) & \text{for } p \text{ in } ABDC \\
\cos \left( 1 - \frac{L}{R_f} \left( \frac{V_i}{V_i + V_f} \right) \right) & \text{for } p \text{ in } CDFE
\end{cases}
\]
CHAPTER 3. FACIAL ANIMATION

The following would appear to give better results:

\[
    d = \begin{cases} 
        \cos\left(\frac{L_t}{R_f} \frac{\pi}{2}\right) & \text{for } p \text{ in } ABDC \\
        \cos\left(\frac{L_t}{R_f} \frac{\pi}{2}\right) \times \cos\left(\frac{V_i - V_f}{V_f} \frac{\pi}{2}\right) & \text{for } p \text{ in } CDFE 
    \end{cases}
\]

Figure 3.22b shows the behaviour of this formulation, where \( V_i \) is made equal to \( V_f \) to simplify calculations. \( d \) is calculated at the intersection points of \( L_t=0 \), \( L_t=R/2 \) and \( L_t=R \) across the width of the muscle and \( V_f=0 \), \( V_f=V/2 \), \( V_f=V \_f \) and \( V_f=V_f+V_f \) over the length. In the region CDFE, the displacement drops to zero at the extremities of the muscle’s influence.

With each of the three muscles described, a cosine (or sine) function is used as a first approximation to the elastic properties of skin. This can be replaced by a power function to emulate the effects of ageing on the elastic properties of the skin. The effect on the linear muscle model is to ‘pinch’ the muscle at its centre, increasing the rate of fall-off to zero at the muscle boundaries. The use of this power function has some similarities to the work of Cobb [Cobb84] and Allan et al [Alla89] described in Chapter 2 where a measurement function or a decay function was used to determine how much to warp the surrounding points in relation to a point being moved.

Figure 3.23 shows the six primary facial expressions of happiness, anger, fear, surprise, disgust/contempt and sadness. Figure 3.24 shows the vertex model for the face, together with a representation of the 18 linear muscles used in Waters’s model and their zones of influence. The left hand side of the face is mirrored to produce the right hand side. Even using a simple mask for a face and not incorporating jaw movement, the essence of the six primary facial expressions is captured. (Version 2.0 of Waters’s program\(^7\) incorporates a jaw simulation. Figure 3.25 shows the jaw opening in the surprise expression.) The figures illustrate the efficacy of the technique. The control of a complex expression – surface deformation – is reduced to the manipulation of a few muscle contraction values. And the technique is independent of surface topology. This is shown in Figure 3.26 which uses FFDs to stretch both the side of the face and the muscles controlling the area (as described in section 3.2.2.4). The figure shows the same muscle contractions as used to produce the six primary facial expressions in Figure 3.23. Figure 3.27 shows the same operations when the chin area is pinched inwards. Figure 3.27 also reveals a small problem with the technique. In the expression for anger (top row, middle image), there is a certain amount of interference around the mouth area. This is because the FFD only affects the head and tail position of a muscle and not its zone of influence. Because the face is compressed, the zone of influence of each muscle should also be compressed.

Figure 3.23: The six primary facial expressions of happiness, anger, surprise, sadness, fear, and disgust/contempt
Figure 3.24: Vertices and muscles of the face

Figure 3.25: Surprise, incorporating jaw movement
Figure 3.26: An FFD is used to stretch the side of the face and also move the end points of muscles, before the six primary expressions are created using the same muscle contraction values as in Figure 3.23
CHAPTER 3. FACIAL ANIMATION

Figure 3.27: An FFD is used to pinch in the chin and also move the end points of muscles, before the six primary expressions are created using the same muscle contraction values as in Figure 3.23.

Whilst the use of such geometric muscles might be considered as a surface manipulation technique, the fact that the muscles do not directly manipulate the surface, but instead manipulate space containing the surface means that the technique could be considered as a general space deformation. It is not unlike an axis based deformation scheme with a tapering function. In reality, a more complex model needs to be employed to simulate layers of tissue and flow with respect to underlying bone structure.

Once a set of expressions has been built up these can be stored in a database and reused on other characters with the same muscle set. New expressions – new surface deformations – can be generated by varying the intensity of existing expressions using interpolation, by mixing expressions (as discussed in relation to the key pose technique in section 3.2.2.1) and, of course, by manipulating individual muscles or defining new muscles. The timing of expressions, e.g. onset, duration and decay, can also be controlled using simple spline curves to mimic acceleration and deceleration effects. With regard to manipulating individual muscles to form expressions, it should be noted that individual muscles do not act in isolation. As shown
in Figure 3.24, their zones of influence overlap. To cope with this a displacement for each muscle is separately calculated from the current facial position and the set of displacements is then averaged to calculate the final displacement value.

In order to increase the complexity and subtlety of expressions on a face, more muscles can be implemented, each affecting smaller and smaller areas. In the Pixar animated short film ‘Tin Toy’ (described in [Ree90]), 47 muscles – a combination of linear and sphincter muscles – were used to control the baby’s face. This meant a significant reduction in animation control overhead in comparison to manipulating individual vertices. However, with more muscles comes more complexity and more chances for interference between overlapping muscles. The film thus grouped the initial 47 muscles into 33 macro muscles, although 12 did most of the work. One macro muscle controlled several ‘ordinary’ muscles. The model of the baby’s face was extremely detailed and incorporated six thousand node vectors. Macro muscles could be used to give gross shape control and the ordinary muscles then enabled lower-level tweaking. This is an example of a hierarchy of detail approach to complexity control. There is also no reason why the concept of macro muscles couldn’t be further extended to use ‘expression’ macro muscles, so that the muscles contributing to a smile could be manipulated as a single parameterised expression.

Whilst all the techniques discussed so far in this chapter have used polygon models, Reeves's [Ree90] work is also interesting in that a net of bicubic Catmull-Rom patches was used to represent the face. Whilst the technique of using muscles can be discussed independently of representation, the representation does have an effect on the final results. As mentioned in the previous chapter, one of the main problems with using parametric patches is the problem of continuity between patches. With so many patches in the baby’s face this caused inevitable problems in the Tin Toy animation. Creasing effects on the cheek areas near the mouth are certainly noticeable. Wang and Forsey [Wang94] describe how hierarchical B-splines can be used as the facial representation to address continuity problems. Again, muscles are used for expression control. But, even with a “5-level hierarchical B-spline with 1584 individual bicubic patches” [the representation is] “affected by the underlying quadrilateral domain of its parameterisation” [Wang94]. The model is a basic toroidal form with the mouth as the opening. Whilst the hierarchical B-spline allows localised adjustment of facial features, the faces produced in the work are ‘too smooth’. They look like masks rather than faces.

Instead of using a surface model for the face, Terzopoulous and Waters [Terz93] embed the muscle structure in a more complex tri-layer model. The three layers of the model are the cutaneous tissue, the subcutaneous tissue and the muscle layer. Each layer is modelled as a deformable lattice which is an assembly of point masses connected by springs. Spring stiffnesses are set to reflect the different tissue properties. The model undoubtedly attempts to address the shortcomings or standard facial models by respecting the fact that the face is an elaborate biomechanical system, but the final images, in comparison to the less complex geometric approaches discussed earlier, do not seem to reflect the much higher complexity of the modelling technique.
An interesting aspect of Terzopoulos and Waters’ work is the use of video tracking to estimate the muscle contraction values that were used to produce a particular expression. This implies that it is difficult to produce a ‘realistic’ expression by tweaking muscle parameters and that such an expression can only be produced by measuring or estimating actual muscle contractions in some way. This can be compared with the situation in using motion capture data to drive articulated structures. The motion capture data gives realistic looking movement, but joining separately recorded motion capture sequences is problematic and can destroy the natural look that was aimed for by initially using the motion capture data. The same could be said for deriving muscle parameters from video information. It may give a natural-looking expression, but how would a sequence of expressions be joined together? Or, how would an expression be varied and still look natural? Having said this however, we note that the biomechanical approach offers a promising way forward for uniting the separate disciplines in facial animation in a single control strategy.

**Figure 3.28: Interpolating key pose vertex values versus muscle parameters**
CHAPTER 3. FACIAL ANIMATION

We might conclude by saying that the use of muscles for expression control is an example of a hierarchical approach to coping with complexity, and indeed one based on a natural technique. Gross muscle contraction values can be used to produce gross effects, but for subtle control it is not clear how to control the interactions between the many muscles needed. Figure 3.28 illustrates that for gross control, interpolating key poses gives just as good results as interpolating muscle parameters. Figure 3.28a shows an interpolation between the muscle parameters for the two key expressions of happiness and anger. Figure 3.28b uses the vertex positions for the interpolation. Figure 3.28c is the difference between the two sets of images multiplied by 4. The mouth area is the only area that shows any real difference and that is due to a slight parting of the mouth. It could be argued that the shape change differences between the individual faces are small. However, perhaps this itself is a pertinent fact with regard to facial animation.

3.4 Lip control - Visible speech

In the previous sections, we have seen that a range of medium-level animation techniques have been used to control facial conformation and expression. In contrast, the work on visible speech has, in most cases, remained at the low level of animation control. The mainstream technique used has relied on the use of visemes [Lewi87, Hill88, Broo92, Kane92, Wate93] to drive the movement of the mouth. A viseme is the static mouth shape related to a particular phoneme in speech work, where a phoneme is “a minimally distinct, abstract class of sounds in a language” [Bhas94]. Thus a sentence is broken into a sequence of phonemes, then these are matched to visemes and then this sequence is interpolated to produce mouth positions. The mapping is not one-to-one and one mouth shape usually suffices for more than one of the 45 phonemes ([Park96] quotes 45 phonemes, [Hill88] quotes 40 or so). Figure 3.29 shows four example mouth shapes from Waters and Levergood’s work [Wate93]. These are defined as sets of polygons, which would be used to adjust the vertices defining the mouth of the face accordingly.

![Figure 3.29: Four example visemes from a table of mouth shapes based on observation of real lips (after [Wate93])](image)

We can consider three types of system: speech-driven, text-to-speech and video-driven. In a speech-driven system, mouth positions (e.g. visemes) are synchronised to a speech track that already exists. This technique is used in traditional cartoon animation. A very simple automatic
version of this is to just open the mouth by an amount based on the volume at a particular instant. In a text-to-speech (TTS) system, the text is broken down into phonemes, which are then mapped to visemes. The speech information and the visual information are then played in synchronisation. This basic process must be enhanced by including information about prosody – intonation, rhythm and phrasal timing – and temporal structure and pausing. This is necessary to make the sequence of phonemes sound more natural.

In computer graphics work, TTS systems concatenate phonemes. In general speech work, one of the crucial issues in such systems is what size the portions are that are used in the reconstitution process. Keller and Caelen [Kell94] consider three levels of reconstitution. The simplest and possibly most banal level is to use whole sentences, as in information giving systems: “next stop the Castle”. In the second type, part of the recording is stable and part is generated from recordings of single words. Perhaps the most common information system employing this approach involves reading phone numbers: “The number you require is 1-2-3, 9-8-7”. ‘The number you require is’ is a recorded piece as are the separate digits. A comma can be used to signify that the digit it follows should be chosen from a recording of that digit with pause characteristics, i.e. with lengthened final syllables and special intonation contours. Another application for a word and phrase concatenation approach is a chess program [Pijp97].

The third type of reconstitution uses much smaller segments of speech or even abstract elements, and the challenge is to reconstruct almost any speech. The phoneme approach falls into this category.

In a video-driven system, both the speech and the image are available. Perhaps the simplest approach is to use automated rotoscoping. Lip shape is measured from video and applied to a facial model. Guiard-Marigny et al [Guia97] and Le Goff et al [LeGo97] compute six parameters from each frame of the video tape, five for the lips and one giving the jaw position. The five lip parameters are the width and height of the internal lip contour, plus three lip protrusion values, one for the upper lip, one for the lower lip and one for the lip contact position. Each of the lip protrusion values is measured from an imaginary reference plane behind the lips. These parameters are then used to drive a model of the lips overlaid on a skull with moveable jaw.

As stated, the computer graphics community has tended to use the viseme-based approach. The early work in this area tended to enhance speech work. Indeed, it is clear that such bimodal communication can help in noisy environments. However, a broadside to the viseme/phoneme based approach is given by the statement that “all efforts to string together phoneme-sized chunks of speech have failed” ([Klat87] quoted in [Bhas94]). Commercial applications using phoneme waveforms have met with “limited success, due to coarticulation problems” [Styg94]. How does mouth movement compare to speech creation? The viseme is a static entity. Interpolation techniques must be used to join viseme sequences together. Perhaps some dynamic representation of mouth movement is needed. Using a word as a basic unit in speech has had limited success, although de Pijper argues its case [Pijp97]. It would be interesting to see if a word based equivalent to the viseme would prove useful in visual speech work. Consider a typical vocabulary for a human of, say, 2000 words. De Pijper quotes 7,760 bytes
for the word ‘check’ on a chess CD-Rom (Microsoft Windows Wave format: mono, unsigned 8
bit, samples at 22,050 Hz). Rounding this up to 10K as an average for a word, a simple
calculation for 2000 words would give approximately 20Mb which is paltry in today’s memory
rich computing environment.

The word-based approach would however still suffer from the same problems that dog the
speech community, that of joining the pieces together so they seem like a natural sequence.
This is not dissimilar to the problems in dealing with general motion capture data. Within
computer games, motion captured facial animation is perhaps the norm, although it seems a
strange contradiction to go to the expense of driving a 3D model with motion capture data for a
limited number of speech segments, when a more general approach involving visemes would
not be too much of an extra cost for the flexibility given. Also with computer games, the small
size of on-screen characters would mean that some of the problems of using visemes would
barely be noticeable.

One aspect of mouth animation that we have not discussed is how it is mixed with general
expression control on the face. One simple solution is to use the viseme to calculate the mouth
shape and then to distort this to produce the expression values. The problem with this is that the
muscle vectors would be operating on a different part of the facial mesh if it had already been
distorted by the viseme shape. If instead the orbicularis oris muscle was simulated to drive the
mouth shape, this would still present problems when mixing with the other muscle vectors,
since the orbicularis oris is free-floating – it is not attached to the bone as are most of the other
facial muscles. It would deform the mouth area giving similar results to a viseme, but would
then itself be deformed when the other muscles acted to produce an expression. It would be
difficult to relate its deformed state to its control parameterisation and thus control of it would
be lost. In effect, every new face position would have to be created by applying values from a
rest position rather than incrementally calculating them.

So far we have just addressed the manipulation of the mouth area. For complete speech
animation, the rest of the face must be coordinated too. Pelachaud [Pelah9]a] addresses this in
her thesis work. A script specifying phonemes, emotion and intonation information is used to
drive a facial model. This includes such things as eye-blinks – voluntary and involuntary,
eyebrow actions based on speech information and rules for conversational turn taking. This
kind of system is the starting point for a high-level animation system, featuring conversational
characters who respond to tasks with in-built behaviour. This would be a hierarchical system,
with control systems offered at the geometric (physical) level – movement of vertices, the
structural level (active regions – muscle control), the expressive (feature) level – relationship
between features and the conversational level (emotional intent – in-built behaviour).

3.5 Summary and discussion

The chapter has presented facial animation as three areas of shape change control: facial
conformation, facial expression and lip movement for visual speech. In the section on facial
conformation, FFDs were used to enable new faces, such as caricatures, to be produced from a
CHAPTER 3. FACIAL ANIMATION

canonical face. This was integrated with the muscle method for facial expression control, since the same muscles that were used on the canonical face could be used on the conformed face. Thus a powerful synergy is offered, with two deformation techniques being used to give expressive control over any face. However, despite the effectiveness of the combination, the techniques still seem more suited to a stylistic face rather than a realistic face, as the movement doesn’t contain the subtlety of facial expression we are acutely attuned to. A problem still remains over fine or subtle control of facial movement, since this inevitably leads to extra muscles and thus more parameters to control. To a certain extent, macro muscles can be used to address this giving a library of expressions whose emphasis can be controlled using parameterisation.

To address realism aspects, the facial mask models can be enhanced with, for example, eyes, teeth and texture mapping for skin colour in a constructive modelling process. The eyes and teeth are however usually just accoutrements. They have no interaction with the rest of the facial model. In Parke’s work the teeth will protrude through the face if the parameters controlling the mouth area are given large values. By using large muscle contraction values in Waters’s model, the eyes are seen as nothing more than background layers with no interaction with the skin layer. For facial animation to become more realistic the separate aspects need to be integrated within the facial model and its control method. For the geometric techniques, a parameterised feature-based approach that relates parts of the face to each other is needed so that one part of the face deforms in response to another, e.g. the tongue pushes out the face or the lips wrap around the teeth. Perhaps a combination of Terzopoulos and Waters’ physically-based facial animation system with Pelachaud’s work on behaviour is the way forward, with the potential to include all the facial animation complexity in a single model.

Whichever control strategy is adapted, facial animation needs a standard set of metrics which it can analyse itself against for particular application areas. However, little work has been done in this area. Parke [Park91a] and Pelachaud et al [Pela94] have put forward metrics such as:

- image quality – this can be user-based as in “does it look right” depending on application or quantitative;
- realism – compare with real faces and measurements of movement;
- range of control;
- parameter complexity;
- number of parameters;
- intuitive natural interfaces (related to previous two);
- range of possible faces that can be specified (conformation parameters);
- range of possible expressions that can be specified;
- parameter quality measures (related to following three);
- subtlety – control of subtle variations in expression and conformation;
- orthogonality – independence of parameters, e.g. speech and expression;
- capability to serve as a basis for higher levels of control abstraction.
As we can see, most of the metrics address the way that a model is controlled. But usually the metrics remain as a proposal rather than being acted upon. For the realism metric, we could include the following: Cephalometrics, the study of measurements of the face to determine if proportions are natural; Extensiometers to measure load and displacement in tissue; Physiological measurements, e.g. acceleration characteristics of eye; Optical tests – compare rendered faces with photographs under different lighting; And dimensional studies of the movement of the face. Perhaps this also needs to be done in hand with a study of how we perceive 2D video images. This is related to the fact that perhaps the most successful facial animation to date is achieved using 2D morphing effects (e.g. [Beie92]) or using motion capture data or model-based coding for video conferencing [Huan95, Eise98].

An interesting thought to finish this chapter would be the idea of a visual equivalent of the famous Turing test. This could be done using a video conferencing set-up. A user would sit in front of a screen and communicate with an on-screen face. This could either be a non-speaking face with which just facial expressions could be traded or a conversation could be held. Three alternatives could be considered for the on-screen face:

1. A real human face on screen: This would be equivalent to plain video conferencing;
2. A synthesised face on screen: This would be performance-driven by a real human at a ‘remote location’ using, say, a gestural input device or with the system tracking his/her facial motion;
3. A synthesised face on screen, completely driven by software: The software would watch the human and respond accordingly.

The aim would be for the user to determine which one was being used. If he/she couldn’t tell the difference, then the system would have passed the visual Turing test.
Chapter 4.

Detail in Modelling and Animation

Introduction

Very detailed geometric models, made up of hundreds of thousands of polygons that capture minute surface detail, can be gathered using laser scanners. Surface colour information can also be gathered for each data point. However, such detail is overwhelming for all aspects of modelling, animation and rendering. Somehow the detail must be simplified to make control tractable. Polygon mesh simplification techniques [Turk92, Schr92, Hopp93, Hopp96] can be used to produce less detailed models or models at different levels of detail. High-level deformation techniques, such as those discussed in chapter 2, can be used to control animation. Combining the two ideas, hierarchical, layered models can be used with gross movement applied to low detail levels and fine movement applied at higher levels of detail. For example in facial animation, gross detail can be manipulated using the techniques described in the previous chapter. Small-scale detail, such as skin defects, can then be layered on to follow the general movement of the gross detail. Such detail can be layered on using texture mapping effects producing a hybrid approach – the gross representation is geometric and the detailed representation is a texture.

An advantage of a multi-layered approach is that animation of the small scale detail can be done independently of the gross detail. Yet this is also a disadvantage. Consider a scar on an area of skin, with the skin being the gross representation layer and the scar being a layered texture map. The scar should have an effect on the movement at the gross detail level, since the skin should not be so malleable around the scar. The movement at the gross level of detail would ‘stretch’ or warp the layer containing the small-scale detail. This may not be the desired effect, since the small scale detail should possibly prevent certain movements at the gross level.

This chapter will present three interpretations and implementation strategies for detail in modelling and animation:

- Polygons and their problems in interactive computer animation
- A case study of using polygons in shape change animation
- Modelling detail to imitate imperfections
CHAPTER 4. DETAIL IN MODELLING AND ANIMATION

The three areas will be discussed separately, giving the chapter a rather disjoint feeling. The aim though is to show that detail can be interpreted in many ways in the modelling-animation relationship and that that interpretation partly relates to the differences between two dimensions and three dimensions. Two-dimensional techniques can be used to give the appearance of three dimensions.

Section 4.1 will discuss the use of polygons as the de facto modelling primitive in interactive computer games, as a specific example of interactive computer animation. Speed is the paramount issue in computer games and, in general, models are represented using low numbers of polygons in comparison to off-line film animation. The use of detail must be restricted to where it is necessary. The use of multiple levels of polygon detail, together with techniques for smoothing silhouette edges, where the use of low numbers of polygons is most visible, will be discussed. The section will also consider the alternative strategy of using non-planar primitives such as parametric patches in order to generate detail when it is needed.

Section 4.2 will present a case study of the use of polygons in computer games to model cloth. The specific example will be the modelling of skirts. Here, the ‘shape’ of the polygon model is driven by the movement of a human figure. Detailed effects such as creasing are layered on using colour mapping [Catm74, Heck86]. The technique is thus an example of a hybrid, layered approach to detail. Shape change at the gross level is geometric, whereas at the detailed level it is merely a colour map that mimics shape change. The two are used in combination.

Section 4.3 will offer a different interpretation of the word detail. The contrast between the use of texturing techniques and geometric modelling techniques to represent very small-scale will be considered. The use of the word texture here needs careful explanation. We would normally regard a consideration of the detailed or small-scale geometry of a surface as a reference to the ‘texture’ of a surface – skin has a different texture to plastic, the bark of a tree is different to polished wood and the feel of an item of clothing depends on the material it is made of. In computer graphics, in general, the word texture is instead used to imply the visual appearance of real texture, achieved by the simple technique of modulating the coefficients of a shading method to alter the final colour of a rendered object, as in colour mapping (often referred to as texture mapping or material mapping) [Catm74, Heck86] or bump mapping [Blin78]. As an example, if a digitised image of a photograph of bark is wrapped onto a simple cylindrical surface, then the appearance of the real texture of a tree trunk can be given, as in Bloomenthal’s ‘mighty maple’ [Bloo85]. The distinction between appearance of detail and real detail is the important distinguishing factor. Texture mapping techniques create the appearance of detail, whereas geometric modelling techniques create real detail. Section 4.3 will consider the distinction between real and apparent detail by looking at the modelling of imperfections such as lumps, scratches and dirty marks within a 3D environment. These appear over time and could be considered as very small scale shape change.
4.1 Polygons and their problems in interactive computer games

Polygons as a modelling primitive in computer graphics are able to represent any topology and as long as enough polygons are used, the representation of an object will appear smooth. But herein lies their problem. When enough processing power is available, as in off-line work in the film industry, the numbers of polygons used are, to some extent, irrelevant – the model can feature as many polygons as it needs. Indeed, in this situation, polygon meshes also offer intuitive WYSIWYG editing and control, since extensive manpower can be employed to alter and ‘tweak’ individual vertices. But for interactive computer animation, speed is paramount. This implies that there is a tension between the processing time available and simply adding more and more polygons to represent detail. This section will look at strategies that attempt to extract as much detail/quality as possible within the time constraints of the particular application area of interactive computer games, where polygons are the de facto modelling primitive. The nature of this is that the total polygon count within a scene will change from frame to frame. Objects will possibly change their representation quality in each frame in what is a time critical process.

![Diagram](image.png)

**Figure 4.1: Frame rate versus image quality versus cost (after [Helm94])**

We can express the cost relationship between maximum realism of images and frame generation rate in computer games by:

\[
\text{cost} \propto \text{frame rate} \times \text{image quality}
\]

For a fixed environment, increasing the image quality would inevitably lead to a reduction in frame rate and increasing the frame rate would lead to a loss in image quality. In many
CHAPTER 4. DETAIL IN MODELLING AND ANIMATION

computer games the user is forced to make this decision by setting the screen and colour resolution at the start of a game on his particular platform. For a low specification platform, the screen resolution would be set to a relatively low value so that, with fewer pixels to update, the game would play at the required frame rate. For games involving polygons, a high number of frames per second is maintained by using low image quality (i.e. few polygons). Figure 4.1 illustrates this in relation to other interactive applications. As an example of the limitations for computer games, console machines like “the Sony PlayStation can support up to a few thousand polygons per frame with texturing” [Levi96]. Whilst such figures are transient, with updated hardware appearing regularly, especially in the PC world, they nonetheless illustrate the fact that the computer games world must find ways to disguise low geometric resolution detail when smooth objects, such as the human figure, or cloth, are animated. This is interrelated with the need to offer scalability of solution, because of the wide-ranging hardware that game-users have. A game must run on a low-specification machine, but it must also offer the user of the high-specification machine something extra with regards representation quality.

As an example of the use of polygon modelling in interactive computer games, Figure 4.2 show scenes from Actua Soccer 2\(^8\) (AS2). This is especially relevant to our discussion of the modelling-animation relationship since 22 players, each based on a polygonal model, can be in view at any one time and each must appear ‘organic’ rather than ‘robot-like’ to the user. Smooth gross movement is achieved by the use of motion capture data, but with up to 22 polygonal models on the screen at any one time, each one can only be modelled using relatively few polygons if the frame rate constraint is not to be compromised. (Frame rates of 15 frames per second are at the lower border of workability and 25-30 is what is usually aimed for.) In AS2, the maximum number of polygons used for a single character is 320. With few polygons for an individual player, representation quality is of paramount concern. In contrast, games involving a single human character and a small range of adversaries can use more polygons for the main human character since fewer characters are on screen at any one time. As an example, the maximum number of polygons used for a single character in Actua Tennis is 1500.

The problem of image generation time with regards to the representational quality of models is not unique to the computer games industry. Flight simulators grappled with the problems in the seventies and, more recently, the techniques have resurfaced in virtual reality research. To deal with collections of objects, hidden surface rendering techniques employing partition planes where much processing is accomplished offline can be used and specialised techniques can be used for particular kinds of objects, such as terrain [Coble94]. For individual objects, different levels of detail can be used [Clar76]. This means that as a viewer gets closer to an object and the object’s screen projection area is increased a more detailed polygonal representation is used, and as the viewer gets further away and the object gets smaller a less detailed representation is employed. It is the use of level of detail (LOD) that has perhaps found most

---

\(^8\) Actua Soccer 2 is a football simulation game produced by Gremlin Interactive. Data for this game, for Actua Soccer 3 and for Actua Tennis was supplied via Ian Badcoe, Head of Research and Development at Gremlin Interactive
use in computer games since it provides a relatively easy way to concentrate processing power. Section 4.1.1 will explore this.

Figure 4.2: Scenes from Actua Soccer 2
The very nature of polygonal models means that any model is made up of a collection of flat faces. Rendering algorithms, such as Gouraud interpolative shading [Gour71] and Phong interpolative shading [Phon75a] can be used to ‘trick’ the eye into seeing a smooth surface instead of a faceted one. But interpolative shading cannot disguise the piecewise nature of any silhouette edges. Section 4.1.2 will discuss silhouettes and present a simple 2D technique for alleviating them. Finally, instead of using planar, polygon primitives, the use of non-planar primitives such as parametric patches and subdivision surfaces will be explored in Section 4.1.3.

4.1.1 Level of detail

The simplest level of detail (LOD) approach involves representing an object with a small number of different polygon models, ranging from a model with many polygons to a model with few polygons, and then choosing a particular LOD to display based on criteria such as distance of the object from the viewpoint. It is this basic idea that has been employed in computer games. For instance, in Actua Soccer 3 (AS3), released in late 1998, three LODs are used for a player. For the high LOD, 480-520 polygons are used, for the medium level 280-320 and for the low-level 50-60. The number varies for each detail level depending on the complexity of the head model. These LODs serve for both the PC and PlayStation platforms. A fourth, very high-level of detail with 550-600 polygons is also used for high-end PCs. AS3 illustrates that the hardware profile aimed at for computer games is continually changing since the highest LOD in AS2 is approximately equivalent to the medium LOD in AS3.

The use of a fixed number of LODs means that a particular release of a game is not scaleable since its LODs are fixed. If a more sophisticated LOD technique was adopted, then the game would have visual appeal to offer on more advanced platforms, perhaps increasing its longevity. The issue of scalability is however tied up with a general process of load management. Load management techniques in relation to representation quality have, in general, been relatively unsophisticated in computer games. For a particular PC specification, the user makes the decision for screen and colour resolution at the start of a game, based on his or her hardware’s capabilities. For polygon counts, Levinson [Levi96] suggests that these can be dealt with in the pre-implementation stage for a game with detailed plans being used to determine what will be in view at any stage of the game and thus what the maximum complexity will be. The obvious disadvantage of this approach is that such a global decision means that visual quality could suffer in local sections of a game where the overall complexity was low and a more detailed model could be used. In AS3, the decisions about which player LOD to use are made dynamically during the game, based on the following factors: proximity to action; number of characters visible on the screen; number of polygons remaining in the polygon buffer and distance from camera. Interestingly, in AS3, LODs can also be mixed, for instance by using a medium LOD head on a low LOD body.

One of the problems of using a polygon LOD approach is the low number of LODs. Rather than use a small fixed number of LODs, a high detail polygon model could be employed and mesh optimisation could then be used to locally refine the polygon mesh to produce a polygon
CHAPTER 4. DETAIL IN MODELLING AND ANIMATION

mesh at the required LOD. The refinement process could be based on criteria such as implied surface curvature, calculated by measuring the variance in surface normals of triangles meeting at a vertex. Hoppe [Hopp96] gives a categorisation of the problems and advantages of mesh optimisation, as follows:

- **Mesh simplification:** this means reducing the polygon mesh to a level that is adequate for the quality required. For AS2 and AS3, the models were built manually. Mesh simplification techniques could be employed to ease the job of artists. They could also possibly be used as a pre-process to a game creating a fixed set of LODs from a very high LOD, based on the hardware capability of the particular machine.

- **LOD approximation:** using a fixed set of LODs can lead to abrupt visual transitions – ‘popping’ – as one LOD is instantaneously replaced by a more (or less) detailed LOD. This is most noticeable at silhouette edges. It can be alleviated by blending or interpolating (similar to the ‘inflation’ techniques used in terrain generation [Turk92]) between LODs over a series of frames. Typical real-time image generation systems provide between 8 and 33 LODs and use complementary blending [Jone94]. In AS2 and AS3, three LODs are employed and blending is not used, since to blend more than one LOD means drawing two LODs which would mean a reduction in representation quality in order to satisfy the frame rate constraint. Interpolation techniques could be used but, in practice, for fast games, the popping effects are often mitigated by the high speed of the game.

- **Progressive transmission:** Incremental data transfer could be used to progressively improve a mesh transmitted over a communication line. This could be used in network based games, depending on the type of game.

- **Mesh compression:** as well as using mesh simplification, the storage space for a mesh data structure can be compressed. With a greater number of LODs, some technique for storing differences rather than the full mesh at each LOD could produce a smaller memory footprint.

- **Selective refinement:** it should be possible to use the LOD in a context dependent manner. Hoppe gives the example of a user flying over a terrain, with the terrain only shown in detail near to the viewer and within the field of view. This has obvious application in games. The example given for AS3 above involved using different LODs for different parts of the body, namely the head and the rest of the body. This idea could be extended depending on the type of model.

Addressing these issues, Hoppe [Hopp96] proposes progressive meshes. In a pyramidal approach the coarsest level of detail $M_0$ is stored together with the information to ascend from layer to layer to the highest detail layer $M_n$. The hierarchy is initially constructed by using mesh optimisation techniques on the highest detail layer to construct each of the lower layers. This involves using a single edge collapse operation – two vertices combined into one – to simplify a mesh from one level to the next as shown in Figure 4.3, where $V_t$ and $V_s$ are the two vertices in the finer mesh that are collapsed into one vertex $V_s$ in the coarser mesh. In effect two faces are collapsed into two new edges. The reverse process from low to higher level of detail is achieved by storing the inter-level transformation details needed to split a vertex back into two
vertices to form an edge. Hoppe quotes an example of an object with 13,546 faces which was simplified to an $M_0$ of 150 faces using 6,698 edge collapse transformations. The data for the model is thus stored as $M_0$ together with the 6,698 vertex split records.

![Diagram of edge collapse and vertex split transformations.](image)

**Figure 4.3: Hoppe's edge collapse transformation (after [Hopp96])**

Hoppe also defines a continuum of geomorphs between any two levels of detail by using a blending parameter to linearly interpolate the position of one vertex of an edge towards the other vertex of the edge, thus geometrically morphing between two successive LODs. Texture coordinates, together with other scalar attributes such as colour, can be interpolated in the same way.

The crucial question remaining is how to select a particular edge for collapse from one level to the next. This can be done using a simple heuristic approach or by a more rigorous method that measures the difference between a particular approximation and the original mesh. A simple metric that can be used to order the edges for collapse is:

$$\frac{|V_s - V_t|}{|N_s \cdot N_t|}$$

which is the length of an edge divided by the dot product of its vertex normals. The metric works well for meshes containing many polygons, but when it is continually applied the mesh will reach a point where it begins to ‘collapse’, as illustrated in Figure 4.4. The top image of Figure 4.4 shows a teapot represented as 1074 polygon facets. The middle image shows that at 488 polygons the representation is just adequate – the handle is starting to distort. At 292 polygons, the teapot shape has collapsed completely. A more considered approach is necessary to prevent such problems. Hoppe uses a more sophisticated energy minimisation approach for the mesh optimisation process and prevents the ‘collapse’ problem by only permitting an edge transformation if it does not change the topology of the mesh. For example, edges that are part of a boundary must be handled differently to preserve the shape of the boundary.

A final point regarding the use of LODs is that it does not solve the main limitation with using polygons for modelling smooth objects, which is that if too few polygons are used the object’s silhouette edge appears to made up of linear segments instead of appearing smooth. Indeed as we have stated, the use of LODs only accentuates the problem since popping (and shape
change) is most visible at silhouette edges. The next section will present a technique to produce smoother silhouette edges for a polygon model.

![Image of teapots with different numbers of polygons]

Figure 4.4: Using a simple metric for edge collapse. From top to bottom, the teapot mesh contains 1074, 488 and 292 polygons respectively.

4.1.2 Polygons and silhouettes

When enough polygons are used to represent an object, image quality is not compromised. Figure 4.5 illustrates this using a set of teapot images, modelled as (a) 128, (b) 512, (c) 2048 and (d) 8192 polygons (each created by applying a uniform subdivision scheme to a parametric model). The piecewise nature of the silhouette edge, also referred to as geometric aliasing, is visible in both Figures 4.5a and b, whereas in Figures 4.5c and d it appears smooth. For real-time rendering of images, figure 4.5d would be inappropriate because of the number of polygons involved. Depending on the number of teapots to be animated even 4.5b and c may be ruled out. If the teapot of Figure 4.5a was used then the piecewise nature of the silhouette edge would be visible. However, since the rendering of the internal areas of the surfaces, in general, is adequate for each of the teapots, a combination approach may be germane with fewer polygons used in the internal areas and more used on the silhouette edge (see figure 4.6).
Figure 4.5: Polygon counts for different rendered subdivisions of a teapot (a) 128 (b) 512 (c) 2048 (d) 8192

(a) Few polygons
Fast to process
But silhouette is visible

(b) Many polygons
Slow to process
But silhouette is smoother

(c) Combination
Few polygons ‘inside’ object
More polygons at silhouette
Quicker to process than (b)
Silhouette smoother than (a)

Figure 4.6: The final image quality can be improved by using fewer polygons for internal areas and more on the silhouette edge
One of the earliest approaches at smoothing silhouette profiles is given by Phong and Crow [Phon75b]. The silhouette edges are detected by using vertex normals. A simple dot product between a vertex normal and the view direction will indicate whether a vertex normal is pointing towards the viewer or away. An edge is flagged as a profile edge if one of its vertex normals points toward the viewer and the other points away. Since an edge points to its relevant abutting polygons in Phong and Crow’s data structure, a sequence of profile polygons and thus a sequence of vertex normals that point towards the viewer can be found. This sequence of vertices is then smoothed by cutting off corners, as shown in Figure 4.7. Corners are repeatedly cut off until the sequence of straight line segments is regarded as smooth enough. The sequence of new vertices is joined to the inner surface vertices thus forming a set of triangles for shading. The main problem with the approach is that the silhouette of an object becomes (noticeably) smaller. The technique would be inappropriate for computer games since the combination of a low number of polygons for an object and a view-dependent corner cutting process could cause large visual difference depending on the orientation of the polygons.

**Figure 4.7: Cutting corners (after [Phon75b])**

Although Phong and Crow’s paper dates from 1975, the solution appears to have been ignored and little further work seems to have been done until Max’s work in 1989 [Max89]. Max uses a similar approach to flag silhouette edges and vertices in view space. However, instead of just taking the front facing vertex normals, he fits a curve between the two vertices of the edge that spans the profile, one vertex visible and one not. If a normal were ‘interpolated’ along this curve it would start out pointing towards the viewer and end up pointing away as it moved between the two vertices. His technique predicts where this normal would be perpendicular to the line of sight. An extra ‘profile vertex’ is created at this point. The sequence of extra profile vertices are then joined using a parameterised cubic curve in the plane of the normals at the profile vertices. This curve is then subdivided to produce a polygonal arc with enough sides to appear smooth and thus give a smooth approximation to the profile. The vertices of the polygonal arc are then joined to interior points to create polygons. The algorithm is straightforward for an initial set of triangles since a triangle will have at most two edges that are used in calculating the profile curve. Max suggests that a method for dealing with polygons
that have four or more profile vertices needs to be devised although he does not suggest one. For some reason Max fails to suggest the obvious, simplest solution which would be to triangulate each of these offending polygons thus reducing them to the simple case of at most two profile vertices.

Whilst Max’s approach would give a good representation of the silhouette edge, a quicker approach would be to calculate the silhouette edge in screen space using a crude curve fitting technique. The same test can be used to label profile edges and as with Phong and Crow’s method only front facing vertex normals need be maintained. These are projected to screen space and, instead of cutting corners, a curve is fitted through them to give a smooth profile, albeit a false one. Thus the technique does not suffer from a shrinking silhouette and is much quicker than Max’s approach. Figures 4.8 and 4.9 show initial results of using this technique. Figure 4.8 shows a very poor polygonal approximation – 8 polygons – to an elliptical shape, such as, say, the upper arm of a character, and the resulting silhouette edge created using just the polygonal information. Figure 4.9 shows the results for a more complex model of a teapot comprised of 512 polygons. There appear to be some problems for tightly curving silhouette pieces (although for the teapot image these are compounded by non-planar polygons in the object model) and it may be that the extra complexity in dealing with these could significantly lessen any potential time savings. The figures serve to show that the technique could be useful for simple low resolution models of the kind found in games, if the problems identified could be solved. Another aspect that would need addressing is how to compute the shading values for the new areas created by the smoother silhouette. Figure 4.10 suggests that the shading values for extra polygons to fill this space could be calculated from neighbouring polygons, although concavities would mean subtracting information from polygons that may already have been rendered. In the case of Figure 4.10, a continuous sequence of bicubic Bezier curves is fitted to the imaginary silhouette segments in screen space and shading values are calculated as part of the curve subdivision process in the same way as control point positions are calculated.

The main problem with the technique is that the information for all the polygons that abut the silhouette edge must be gathered together in screen space so that the connectivity information between them can be used to order the false profile vertices and fit a curve through them. This means that an extra data structure is required to store the relevant polygons as they come out of the viewing pipeline into screen space. The silhouette polygons are treated differently to the non-silhouette polygons. The problem is the curve fitting process. If only two vertices are available then each curve piece is separate and what should be a smooth edge potentially turns into a corrugated edge. Some guiding values need to be available so that a curve can be fitted between the two vertices and join continuously with other curve pieces for pairs of points from neighbouring polygons. It is not clear how this could be done. A possibility is to project all polygons together for an initial frame and then use some form of frame to frame coherence to track silhouette edges, as used in 3D by Markosian et al [Mark97].
Figure 4.8: Smoothing the silhouette edge of a poor polygonal approximation to an elliptical shape such as, say, the upper arm of a character
Figure 4.9: Fitting silhouette curves to a polygonal approximation of a teapot
Two final points can be made about the use of silhouette edge smoothing techniques, one for rendering and one for general animation. If ray tracing were used to render the scene rather than a standard interpolation technique such as Phong interpolative shading, then coarse silhouette edges would still be visible in the reflections of other objects. For general animation, collision detection would be done using a coarse polygonal representation, whilst display, incorporating silhouette edge smoothing, would possibly lead to visually overlapping objects.

### 4.1.3 Non-planar primitives for computer games

As previously mentioned the use of a discrete set of LODs can lead to popping, which is especially visible around silhouette edges. In the previous section we looked at a simple technique to smooth silhouette profiles. In this section we consider instead using a continuous, curved modelling primitive - a parametric patch. For parametric patches, the necessary LOD is created on-the-fly by recursively subdividing a patch into subpatches until eventually a subpatch is within some limit. The corners of the subpatch then become the corners of a polygon. The criteria for controlling the depth of subdivision can be object space based or screen space based. In screen space pixel metrics are used in the tests; in object space a comparison is made between the true surface and the approximating facets using object space units. Screen-space criteria are more relevant to computer games since they provide the potential of being able to adjust the polygonal approximation to match the projected size of the object on screen. It is however instructive to start by considering object space tests since the subdivision process itself is carried out in object space.

For object space subdivision, two simple categories can be identified: uniform subdivision which involves the user specifying a level at which subdivision of each starting patch is to terminate, and non-uniform subdivision which involves stopping the subdivision process for a
CHAPTER 4. DETAIL IN MODELLING AND ANIMATION

starting patch when a ‘flatness’ criterion is met. The subdivision process is a simple extension of de Casteljau’s algorithm for parametric curve subdivision, which is described in detail by Lane et al [Lane80b]. The teapots of Figure 4.5 were created using this uniform subdivision algorithm.

There are two major problems with uniform subdivision algorithms. First, a fine level of subdivision is needed to produce smooth silhouette profiles. As we have seen, such a fine level of subdivision is not necessary for smooth shading of internal areas in screen projections. The second problem is that as many polygons will be produced in low areas of curvature as are produced in high areas of curvature. Thus an initially flat patch would be subdivided as much as a patch that was tightly curved. This can be solved using non-uniform subdivision.

For non-uniform subdivision, areas of the patch that are ‘flattish’ (exhibit low curvature) are subdivided less than areas where the curvature is high. The patch is subdivided based on local curvature. A simple flatness test is to fit a plane to three non-collinear control points and then measure the perpendicular distance of the other control points from this plane. If one of the control points lies outside a pre-specified distance the patch fails the test. When a patch passes the test, its corner points are used to make a (non-planar) 4-sided polygon or two planar triangles. Although the 4-sided polygon is, in general, non-planar, standard shading algorithms can cope with this. Also, it is quicker to transform 4 vertices rather than 6, which has obvious advantages for computer games. (In fact, for computer games a modified, iterative subdivision process that produces triangle strips would be more effective, since these involve transformation of even fewer vertices on specialised hardware.)

The main problem with non-uniform subdivision is that since different areas of a patch can be subdivided by different amounts, small cracks can appear in the surface (see Figure 4.11). Clarke describes a technique to solve this based on considering patch edges in the subdivision process [Clar79] whilst Von Herzen and Barr [Herz87] makes use of a quad-tree to ensure that cracks do not occur between adjoining parts of a patch (a similar approach is also used in the commercial system from LightWork Design Ltd. [Ligh97]). Either of these approaches could be used although both mean extra complications in the database. For Clarke’s approach, an edge representation is needed; for Von Herzen and Barr’s approach, a secondary data structure (the quad-tree) is needed. Both mean an increase in data storage costs.

![Figure 4.11: Tears produced by non-uniform subdivision of patches](image)

The methods described so far would suffice for computer-aided design work. For interactive computer games, an object may be viewed from many different viewpoints and form
projections of widely varying size in screen space. It is then more appropriate to use screen space criteria to control the subdivision process since if the object appears small then a low LOD (few polygons) should be used so that time is not wasted transforming many small polygons. At a particular level of subdivision, sample points from the patch together with the polygon that is an approximation to the patch are projected into screen space and compared. Pixel length units are used as the metric. The LightWorks commercial rendering system offers tests based on minimum pixel area occupied by a patch, screen space flatness of a patch or screen space flatness of the silhouette edge [Ligh97].

For the minimum pixel area test, the patch is deemed to be sufficiently subdivided when the pixel area is within a predefined limit. Catmull [Catm74] used this test for direct display of patches by subdividing patches until each occupied the area of a single pixel when the result could be written directly into a z-buffer, although such an approach would clearly be prohibitively expensive for a computer game. For a game, it may be more appropriate to use the approximating polygon rather than the patch control points in the test. But this could cause problems since it may be a poor approximation to the patch and may even project as a line when the set of patch control points project as an area.

The screen space flatness test is similar to the object space flatness test but uses pixel space metrics instead of object space units. The patch control points are again measured with respect to a plane through three non-collinear control points but this time using pixel units. An alternative technique is to create the corner points for the approximating polygons at the next level of patch subdivision, project these into screen space and compare them against the single approximating polygon for the current level of patch subdivision. This involves fewer tests in screen space, albeit at a slight subdivision cost in world space (which may have to be done anyway if the flatness test fails). This would be quicker and thus more appropriate for use in a game.

The screen space flatness test for silhouette edges is the same as the general screen space flatness test, but is only applied to patches that contain a silhouette edge. This means that silhouette edges need to be detected first. For patches, Elber and Cohen’s [Elbe90] method can be used as a quick test to determine if a patch contains a silhouette edge. It is based on using a parallel projection with the image plane at $z = 0$ and using the fact that the $z$ component of the surface normal will then be zero for points on a silhouette edge.

It is difficult to determine whether uniform or non-uniform patch subdivision would be appropriate for computer games. Although fewer polygons are produced using non-uniform subdivision, it may still be more appropriate to use uniform subdivision. The cost of the flatness tests in non-uniform subdivision are, in general, high and perhaps do not offset the savings in transforming and rendering fewer polygons. It really depends on the complexity of the model. For very large numbers of polygons, which could possibly be produced using a uniform subdivision algorithm, bandwidth limitations could be a problem on current graphics cards for computer games, especially when that bandwidth is shared with lots of texture information. Thus non-uniform subdivision would seem to be better, although texturing could be a problem for the non-uniform subdivision process since it may create widely varying
CHAPTER 4. DETAIL IN MODELLING AND ANIMATION

polygon sizes, which could cause distortions in the texture mapping process. Uniform subdivision would not suffer from the same problem.

Perhaps the main hurdle in using patches as the object representation for computer games is that they must eventually be converted into polygons for the rendering hardware (at least until the graphics cards provide routines to deal with patches in the same way as polygons). This means that the patch subdivision process must be efficient. Lane and Riesenfeld [Lane80a] present a way to speed up the patch splitting algorithm, although they don’t give figures on the amount of speed-up. The use of frame-to-frame coherence could be used to help speed up the splitting process, with the previous patch LOD being maintained and used to seed the splitting process in the next frame. This would increase the data storage requirements but would lead to significant savings in the subdivision process. A further enhancement of this idea might involve a combination approach. A patch could, in effect, ride ‘piggy-back’ on a polygon and be used to produce more polygons when needed. This could be produced as part of the modelling effort, or could be generated automatically. Van Overveld and Wyvill [Over98] describe a method that can be used to produce a smoother model from a less detailed approximation. They do this by using vertex normals to fit a parametric patch to each polygon. The patches are then subdivided to produce a new smoother polygon model, albeit most likely not the original model that was initially approximated by polygons.

The main disadvantage of the piggy-back method is that if polygons were treated independently, this could lead to the same ‘cracking’ problems as with non-uniform subdivision. Also hard edges, such as the hard edge between the side and top of a cylinder, would need to be clearly marked so that they would remain hard and not be smoothed into neighbouring polygons. An advantage of the piggy-back model is that it can be used to produce an initial model quickly. For games, a simple polygonal model (low LOD) of an object could be produced manually in a relatively short amount of time. This would approximate the salient features of the object, for example the main body components for a human figure. It would also have the relevant control handles attached, for example a skeleton for the human figure. The control handles would then be retained whilst van Overveld and Wyvill’s technique was used to produce smoother versions of this starting model, i.e. higher LODs. The final results may not be the true object but they are smoother versions of the salient features and would perhaps suffice for the purposes of an interactive computer game. The use of this process could cut down on the time taken to produce a set of LODs. Another advantage is that the relationship between the starting set of polygons and the skeleton control handles would be carried into any new set of polygons produced from the patches, since each patch occupies the same area as a starting polygon and thus the new set of polygons can be attached to the same part of the skeleton as the starting polygon.

More recent research has expanded on the idea of using a mesh of planar primitives as the starting point for modelling work. Subdivision surfaces [Catm78, Schw96, Zori96, Zori97, DeRo98] involve taking an arbitrarily complex polygon mesh and using it as the starting point for a subdivision process that produces a limit surface. As with parametric patch techniques, such subdivision could be used dynamically in a game to produce varying levels of detail. The
advantages however are that the subdivision scheme can be altered on different parts of the surface more easily, and also the initial starting model can be any polygon mesh net, thus addressing some of the topological restrictions of quadrilateral nets of parametric patches. A detailed consideration of subdivision surfaces is not given here, but Table 4.1 gives a brief comparison with polygon meshes and nets of bicubic Bezier patches for use in computer games. Given the advantages outlined in the table, we might postulate that the adoption of subdivision surfaces could lead to an increase in the range of organic objects and deformation effects used in computer games.

<table>
<thead>
<tr>
<th></th>
<th>Polygons</th>
<th>Patches</th>
<th>Subdivision surfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intuitive</td>
<td>Positive: WYSIWYG</td>
<td>Negative: although they can be intuitive for certain classes of object, for games there are problems with continuity, connection, skinning and topology</td>
<td>Positive: not quite as intuitive as polygons, since editing surface is an approximation of the final subdivided limit surface</td>
</tr>
<tr>
<td>editing</td>
<td></td>
<td></td>
<td>---------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Topology</td>
<td>Positive: any topology can be represented</td>
<td>Neutral: extra complexity if non rectangular patches used, but using just rectangular patches puts some limits on topology</td>
<td>Positive: any topology can be represented</td>
</tr>
<tr>
<td>Levels of</td>
<td>Neutral: can be generated but data at each level is separate, e.g. texture data</td>
<td>Positive: in general many LODs can easily be generated, but some cost implications</td>
<td>Positive: in general many LODs can easily be generated, but some cost implications</td>
</tr>
<tr>
<td>detail</td>
<td></td>
<td></td>
<td>---------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Connection</td>
<td>Positive: easy to connect separate meshes by sharing vertices</td>
<td>Neutral: connectivity problems can cause knock-on effects for modelling</td>
<td>Positive: easy to connect separate meshes</td>
</tr>
<tr>
<td>Smoothness</td>
<td>Negative: inherently angular; lots of polygons for smooth surface: silhouette problems</td>
<td>Positive: inherently smooth</td>
<td>Positive: smooth when required</td>
</tr>
<tr>
<td>Small-scale</td>
<td>Neutral: must use bump-mapping or extra geometry</td>
<td>Neutral: must use bump-mapping or extra geometry</td>
<td>Neutral: must use bump-mapping or extra geometry</td>
</tr>
<tr>
<td>features, e.g. wrinkles</td>
<td></td>
<td></td>
<td>---------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Creases</td>
<td>Positive: easy to add sharp creases</td>
<td>Neutral: add complexity to connectivity information and to shading</td>
<td>Positive: less so than polygons, as extra subdivision rules need to be defined</td>
</tr>
<tr>
<td>Skinning</td>
<td>Skinning means smoothing character, to form a continuous mesh between two objects, e.g. the upper and lower arm of a mesh</td>
<td>Negative: problems in joining separate nets of patches, especially when the two nets must then move independently</td>
<td>Neutral: more positive than polygons, since a single subdivision surface could cater for a whole object</td>
</tr>
<tr>
<td>Texturing</td>
<td>Positive: easily textured</td>
<td>Positive: readily textured and LODs are all textured as one process</td>
<td>Neutral: possible distortion problems in LODs, although recent work by [DeRo98] addresses this</td>
</tr>
</tbody>
</table>

118
Continued…

<table>
<thead>
<tr>
<th>Shading</th>
<th>Polygons</th>
<th>Patches</th>
<th>Subdivision surfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive:</td>
<td>Positive: hardware set up for fast rendering</td>
<td>Positive: easily shaded, but at extra cost in converting to polygons</td>
<td>Positive: readily shaded</td>
</tr>
<tr>
<td></td>
<td>hardware set up for fast rendering</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bandwidth</td>
<td>Negative: storage and processing costs escalate with number of polygons needed for smoothness, small-scale surface features, and skinning</td>
<td>Neutral: less initial representation but extra cost in producing final representation</td>
<td>Neutral: less storage than standard polygon model, since subdivision process produces smoother surface, but extra cost in producing final surface</td>
</tr>
</tbody>
</table>

Table 4.1: Comparing polygons, bicubic Bezier patches and subdivision surfaces for use in computer games

4.2 Using polygons in shape change animation – a case study: cloth

The earliest computer graphics work attempted to model the way that cloth hangs or drapes over an object [Wei86], whilst more recently, the clinging effects of skirts and other items of clothing have been attempted [Bar98]. Ng and Grimsdale [Ng96] present a detailed survey of cloth modelling techniques, categorising them into main families: geometrically-based (e.g. [Wei86, Ng95]) physically-based (e.g. [Terz87, Voli96, Ban98]) and hybrid models. We briefly consider the distinction between geometrically-based and physically-based models.

In the geometric approaches, mathematical functions are typically used to produce the folds of cloth and geometric collision detection algorithms may be used to constrain the final cloth position. The technique is most suitable for cloth interaction with simple static objects, although the physical properties of cloth, e.g. elasticity, are not incorporated. The technique is less suitable for the complex interactions that occur when cloth is covering a moving human body. The advantage of the physically-based approaches is that as they are defined, in the main, using differential equations, they are active and able to react to applied forces and constraints. By simulating relevant physical properties, the dynamic shape of cloth as it folds or wraps over surfaces or billows in the wind can be computed automatically. The main problem with the physically-based approaches is that they are compute-intensive. Volino et al [Vol96] quote running times of 30 seconds to 5 minutes per frame for each garment (depending on garment complexity) on a 150Mhz SGI Indigo 2. (No figures are given for garment complexity although Baraff and Witkin [Bar98] estimate Volino et al’s garment complexity as approximately two thousand triangles.) Baraff and Witkin’s work on relatively large time steps in cloth simulation produces improved compute times of 10 seconds per frame for a skirt modelled as 8844 triangles on a dancing character running on a 195MHz SGI Octane R10000. This is still nowhere near real-time, although, in each case, the figures are compounded by the fact that for modelling the interaction of cloth with an underlying surface a large number of modelling primitives are needed around the contact areas. A further disadvantage of the physically-based methods is that the stability of the solutions can be drastically affected by small changes in the simulation parameter values or the modelling elements.

We shall concentrate on the use of a geometric technique to produce the motion of a skirt on a human character. This is in line with the general computer games need for a fast technique that
uses simple polygon primitives. Also, a small number of polygons must be used, otherwise polygons would have to be ‘borrowed’ from other areas of the scene. Whilst the fact that few polygons are being used could mean that a physically based approach could possibly be used in real time, it is not clear that this would produce superior movement when so few polygons are used. Also, in computer games work, appearance is generally more important than physical accuracy, at least as far as characters are concerned. Thus approximation of character movement is acceptable.

In general, most computer games character animation involves the use of motion capture data, because it produces fast, effective results. A library of pre-stored motion capture pieces including, for example, a run sequence, a walk sequence and a jump sequence, can be easily built up. Input from the game player then results in the database being accessed for the relevant motion capture piece to be applied to the game character. In some arcade games this is very apparent as a sequence of quick key presses can be executed and the user can then wait to see the sequence of actions that will result. A lot of work goes into joining the motion capture pieces to make the transitions between pieces appear natural and often pieces are mixed together to extend the range of movements of a character. To side-step these problems, we will use a long motion capture sequence of a dance sequence to test the movement of a skirt added to a character (some frames of which are shown in Figure 2.30). We will also test the skirt movement using a run-cycle produced using sine curves.

Figure 4.12 shows the skirt attached to the pelvis in the character hierarchy. Figure 4.13 shows the polygon model used for the skirt. 64 triangles are required for the skirt, 32 for the inner face and 32 for the outer face. (With some rendering architectures, two-sided polygons could be used.) The use of three layers of vertices corresponds to the use of the skirt. Only skirts up to knee length will be used. Thus the first layer wraps around the character’s waist, the second layer is used to accommodate the posterior and hips and the final, lower layer wraps around the legs above or at knee level.

Throughout the animation sequence, the first layer of vertices in the skirt is attached to the top of the pelvis, the second layer follows the gross movement of the pelvis but is set at a larger fixed radius (which can be varied by the user) and the third layer is dynamically updated to accommodate the movement of the legs. (The second layer could be dynamically updated in a similar way to the third layer, but this was not thought necessary in order to illustrate the technique.) The user can also vary the length of the skirt. Figure 4.14 shows how the position of the legs is used to update the skirt vertices. For each leg, a circle of vertices is formed in the plane that is normal to the leg’s main cylindrical axis. The vertices are calculated using the leg’s other two axes. The two circles of vertices are then used to position the skirt vertices by distributing the skirt vertices at set positions to the fore and aft of each leg and diagonally to the sides. This means that the area of skirt between the two legs will ‘stretch’ to accommodate the movement of the legs. The set of polygons defining the skirt will not limit the movement of the legs – in effect, the skirt is infinitely elastic.

Figure 4.15 shows the technique applied to the run cycle and Figure 4.16 shows it applied to the dance sequence. A very similar approach to this (using an average of the main cylindrical
axes of the legs), which the author collaborated on, is used in Actua Tennis to model the short skirts of female tennis players (see Figure 4.17). Short skirts are a simpler problem, since their movement is very limited and a simple stretching technique such as this one is unnoticeable within the game.

Figure 4.12: The character hierarchy with attached skirt
Figure 4.13: The skirt model

Figure 4.14: The lower layer skirt vertices are calculated from the position of the legs
Figure 4.15: The skirt in the run cycle
Figure 4.16: The skirt in the dance sequence
The technique currently has a number of limitations. The skirt does not interact with external objects, so the character's hand or foot can pass through it, or other objects can penetrate it. This could be easily rectified for other objects, but dealing with the character's own movement would mean interfering with the motion capture data and possibly destroying the naturalness of any motion. Another problem is that the contained objects, i.e. the pelvis and upper legs, can become visible for extreme movements, as the skirt polygons are drastically stretched. This problem is compounded by the nature of the dance motion capture since the upper leg twists quite markedly which means that the circle of vertices created for each leg overlap. The technique is however fast and could also be used for ankle length skirts, if the movement of the legs was restricted.

As we have already noted, the real-time nature of games means that the appearance of small-scale detail must be produced using a combination of low detail polygon models (low geometric resolution) and texturing techniques. Figures 4.2, 4.17 and 4.18 give examples of this. The skirts in Figure 4.17 have few polygons and the creases are mimicked using simple painted dark areas in the texture map. In Figure 4.18 the face of the footballer is a texture map painted onto a low resolution polygon model. An interesting point about the shirts and tee-shirts in Figures 4.2, 4.17 and 4.18 is that they do not exist as separate objects. They are the polygons that make up the bodies of the human figures, with a texture map, again incorporating dark areas to mimic creases, painted on. The texture maps in all these examples are static, but
could easily be animated by moving the dark bands of the texture map using the equations for travelling sinusoidal waves presented in section 2.3.2, to produce the appearance of a flag wrinkling in the wind. (Indeed a similar technique is used in Actua Soccer 3, which the author collaborated on.) This use of texture to give the appearance of detail will be explored in the following section.

![Image: A scene from Actua Soccer 2](image)

**Figure 4.18: A scene from Actua Soccer 2**

### 4.3 Modelling detail to imitate imperfections

In computer graphics, it often seems that the pursuit of photo-realistic imagery, indistinguishable from real-life photography, subsumes all rendering work. Immaculate, pristine imagery pervades. (Interestingly it is only relatively recently that work in the computer graphics world has begun on how to judge the realism of such images [Rush95].) But the world is not perfect. It is full of imperfections. Objects *change shape in small ways over time.* Scratches gather dust and sharp-edged chips in objects are eroded over time. Although Perlin and Hoffert model eroded surfaces using hypertexture [Perl89], Becket and Badler [Beck90] appear to have been the first to have addressed imperfections *per se.* They advocated a rule-based approach to modify the positioning of localised blemishes produced using fractal subdivision techniques. A simple language was used as the interface and images of scratched and blotched objects were produced. However, they only considered surface effects that could be layered on using texture maps. Close-ups of imperfections, which would have needed volume modelling effects, were not considered.
Becket and Badler state clearly that they considered simulation of natural processes over time as “not currently possible”. More recently, Dorsey and Hannahan [Dors96a] have addressed this, concentrating on the processes that produce a specific kind of imperfection. Their work considers the modelling and rendering of metallic patinas by simulating the effect of weathering or atmospheric corrosion. A surface is presented as a series of layers and operators such as coat, erode and polish are used to modify layers in combination with local environmental factors such as the slope of a surface and the amount of water present. Impressive sequences of images showing a copper statue weathering over time are produced. The use of water flow as a weathering agent on surfaces is further explored by Dorsey et al [Dors96b]. The water flow is modelled as a particle system with individual particles acting on a surface. Deposits held in the water are released over surfaces with particular absorbency. The technique captures the weathering effect on walls due to fluid flow and the effect of sediment transport and deposit below rusting metal fittings well. It is difficult to see how similarly effective results could be produced by modelling the appearance of the results rather than the process that produces them. Dorsey et al’s work concentrates on the particular process of weathering. A recent Pixar film ‘Toy Story’ considers the more general problem of worn appearance in rendered scenes. They achieve their effects, such as dirt and scuffs, by using painted textures within a programmable shader [Fren95].

In each of these works the effects that have been modelled can be considered as surface effects, especially as no really close-up views are ventured. Texture maps are prevalent and views are from a ‘normal’ distance. It could be argued that Dorsey et al’s work addresses this concern by using an algorithmic process that could be varied in rendering detail dependent on viewpoint distance. However, Dorsey et al only consider weathering effects rather than the more general problem of surface imperfections. Instead we might consider imperfections as belonging to two families and deal with them accordingly, dependent on distance.

We can classify imperfections in terms of the structural implications of the imperfection into surface and volume imperfections. In general, a volume imperfection alters the geometry of an object whilst a surface imperfection changes the appearance of the object’s surface. Table 4.2 presents an overview of the split between volume and surface imperfections.

<table>
<thead>
<tr>
<th>Imperfection class</th>
<th>Type of imperfection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume imperfection</td>
<td>Lamp — attachment of piece, e.g. bubble, growth</td>
</tr>
<tr>
<td></td>
<td>Hole – removal or apparent removal of piece, e.g. chip, dent, crack, scratch, notch</td>
</tr>
<tr>
<td></td>
<td>Wear – removal of large area, possibly at extremes of object</td>
</tr>
<tr>
<td></td>
<td>Warp – bending of large area</td>
</tr>
<tr>
<td></td>
<td>Partial attachment – near detachment of piece, e.g. peel</td>
</tr>
<tr>
<td>Surface imperfection</td>
<td>Stain – graffiti, spreading addition, e.g. blot, speck, smudge, spill</td>
</tr>
<tr>
<td></td>
<td>Deposit – layered addition, e.g. dust, dirt, sediment</td>
</tr>
<tr>
<td></td>
<td>Erosion – layered removal, e.g. corrosion, dulling, weathering</td>
</tr>
</tbody>
</table>

Table 4.2 Types of imperfections

Using this categorisation, we can see that Becket and Badler’s work is mainly addressed at ‘stain’, whereas Dorsey et al cover aspects of ‘deposit’ and ‘erosion’. The techniques used in
'Toy Story' are a mixture of stain, deposit and erosion. All rely heavily on standard computer graphics texturing approaches such as colour modulation (where the diffuse reflection coefficients in the Phong reflection model are modulated with the corresponding colour from a texture map) [Catm74], which we shall refer to as texture mapping, and bump mapping [Blin78]. Figure 4.19 shows how bump mapping can be used to produce a chewed pencil.

![Figure 4.19: A chewed pencil rendered using bump mapping](image)

These common texturing techniques can be used to model many of the volume and surface imperfections of Table 4.2 for both 'normal' and distant viewpoints. The exceptions are warping and partial attachments. Aliasing can be a problem for distant viewpoints, or when the texture is small relative to the object screen projection, or when the texture is mapped at acute angles to the viewpoint direction or when any periodicity in the texture approaches the pixel resolution, although mip-mapping [Wil83] can alleviate these problems. For close-up views, both texture mapping and bump mapping can be unsatisfactory. The individual pixels of a colour texture map become apparent and blurred if an aliasing technique is used, and silhouette edges remain smooth for bump mapping (see Figure 4.20) since the geometry has not been changed. Whilst a higher resolution texture map can be used in texture mapping (but at greater memory cost) or a procedural generation technique can be used, the inadequacies of bump mapping cannot be so easily rectified.

![Figure 4.20: A close-up of the chewed pencil rendered using bump mapping](image)

As the viewpoint moves closer a switch must be made to a geometric technique that alters the surface geometry. Displacement mapping [Cook84] (which uses a height field to perturb a surface point along the direction of its surface normal) can be substituted (see [Beck93]) or constructive solid geometry (CSG) techniques can be used. Figure 4.21 shows the results of a
CSG subtraction to form a chip in a mug and Figure 4.22 shows CSG techniques being used to model a tumour on an eye used in Waters’s face model (see Chapter 3).

Figure 4.21: A mug and a chip created using CSG techniques

Figure 4.22: A tumour, created using CSG techniques, applied to an eye for Waters’s face model (see chapter 3)
Waring and Macklock [Ware95] have employed these imperfection ideas, using a mixture of texturing techniques, CSG and warping of Bezier patches (all available within the freeware PovRay ray tracing system), to produce a sequence of five images of a desktop (as shown in Figure 4.23a..e) at a number of set points in time. In Figure 4.23a, which shows the scene with no imperfections, all objects are laid out in straight lines to enhance the appearance of a perfect image. In Figure 4.23b, slight imperfections, such as the curl of the paper and a chewed pencil are introduced. Also the position of the objects is altered, some more than others. In Figure 4.23c, the calendar is beginning to curl at the edges, the mug is introduced and the desk has a mug stain on it. In Figure 4.23d, there are more marks on the desk, coffee stains on the piece of paper where the mug was and one of the pencils is now very short and sharpened at both ends. In the final image, Figure 4.23e and f, the mug has been knocked over causing a spill on the table, the table has doodles, scratches and stains on it, the stains on one of the pieces of paper are clearly visible, the calendar has fallen partly away from the wall and the ruler is broken. Despite the inference that the desktop is evolving over time, it is not clear how to produce a general technique that can automatically produce such changes in wear or very small scale detail in an environment. The work of Dorsey et al and Dorsey and Hanrahan is perhaps the closest to this goal, although they are both limited to erosion and sediment deposit effects. A general technique is needed that will produce imperfections based on the properties of the environment and the range of objects within the environment.

Figure 4.23a: Stage 1 in the evolving desktop scene: no imperfections
Figure 4.23b: Stage 2: slight imperfections – warped paper and one chewed pencil

Figure 4.23c: Stage 3: Mug stain on the desk and curling edges on the calendar
Figure 4.23d: Stage 4: More marks on the desk, coffee stains on the piece of paper where the mug was and one pencil is now very short and sharpened at both ends

Figure 4.23e: Stage 5: The mug has been knocked over causing a spill on the table, the table has doodles, scratches and stains on it, and the ruler is broken
When an object is modelled it is usually assigned particular material properties, perhaps including a material texture. Extra properties could be assigned to the object to model its interrelationship with its environment. This extra property could be related to the imperfections listed in Table 4.2. A steel object is unlikely to dent so the ‘hole’ value could be assigned a low probability factor, whereas a wooden object could be given a high probability factor of gaining a dent. This imperfection factor could be associated with a standard time period in which it would be likely to occur. The shape of the object would also need to be considered. For instance a hollow metal object is more likely to dent than a solid one. A further factor is the usage of the object. A wooden table is more likely to receive a dent than a wooden picture frame. Finally, an environmental factor could be associated with an object. A wooden table with heavy, sharp edged objects on it is more likely to receive a dent than one with a tablecloth on it. If a tin of paint is left on the table, the environmental factor of ‘stain’ associated with the table would be increased. Such factors could be gathered into a matrix for an object and called a ‘wear-and-tear transformation’. Table 4.3 gives a speculative example for a wooden table with an open tin of paint resting on it. The material factor only corresponds to the fact that it is a wooden object. The shape factor is only concerned with the shape of the wood, which in this case is a large, rectangular, thin piece of wood (for the table top) which thus has a greater chance of warping than a thick chunk of wood. The usage factor considers the object as a table and the use that is normally made of a table. The environment factor considers the other objects in the environment, which in this case is an open tin of paint. Thus the factor for the stain imperfection is greater than 1. The hole imperfection has a factor of 1.1 to indicate that the paint container is made of tin which may dent the wood.
### Table 4.3 Possible wear-and-tear transformation information for a wooden table with an open tin of paint resting on it

In this example, all the factors are encoded into the table. In practice, some of the factors could be automatically calculated based on what other objects were in the environment or in the vicinity of the object. For instance, if a sharp metal object was placed on the table, the likelihood of a scratch would increase so the environment factor for the one of the subcategories of ‘hole’ would be increased.

#### 4.4 Summary and discussion

Three interpretations and implementation strategies for detail in modelling and animation have been presented in this chapter. In the first case, the use of polygons in interactive computer animation, specifically computer games, has been explored. The study of LOD was concerned with a pure 3D approach to ensuring maximum image quality within the frame rate constraints. This inevitably leads to a consideration of where the problems lie in producing best image quality.

Interpolative shading schemes visually smooth over internal edges in an object, but cannot deal with silhouettes. It is interesting to note that whilst much effort has been expended in computer graphics in dealing with aliasing problems in rasterised images, relatively little work has been done on the geometric aliasing shown by silhouette edges. We seem prepared to tolerate geometric aliasing. Yet the combined approach of allowing the interpolative shading algorithm to deal with internal areas of an object and using a smoothing approach to silhouettes would seem germane for improving image quality at small cost. The technique presented in this chapter for smoothing silhouette edges in two dimensional screen space initially seems a promising approach. However, the need for extra data structures so as to put silhouette edges into linked order as they are projected to screen space means that the technique interrupts the normal polygon-by-polygon flow of a renderer.

The alternative approach to dealing with detailed models is to use higher level primitives so that polygons are replaced by, for instance, Bezier patches. These allow silhouette edges to be easily dealt with as part of the subdivision process for turning patches into polygons. The conversion process is necessary as current hardware is tailored for polygons. This means that the inertia of the polygon mesh is high, with other reasons being the low complexity mathematics required to handle them and the fact that interpolative shading makes such a good
CHAPTER 4. DETAIL IN MODELLING AND ANIMATION

job of making internal object edges invisible. A brief comparison of polygons, Bezier patches and subdivision surfaces, with regards their use in computer games, indicated that the higher order primitives had a lot of advantages to offer. Indeed, over the last few years Pixar has moved from polygons to patches and now to subdivision surfaces [DeRo98] for use in offline computer animation. Whilst, for computer games, the polygon still has the upper hand, it would be interesting to see the impact of a higher level primitive being implemented on a graphics card.

The second interpretation of detail within the chapter looked at a case study using polygons for shape change animation. The case study involved the implementation of a skirt for a computer game. The polygon model was dynamically updated in 3D to give gross shape change and 2D texture maps were tattooed on to the polygons to give fine detail such as creases, thus giving a layered approach to detail. For a computer game, the technique is very effective. The example suggests that there is a tension between 3D and 2D in producing detail in modelling and animation. This was further explored in the modelling of detail to imitate imperfections. A tension between 3D techniques giving real detail and 2D techniques giving apparent detail was explored. In focussing on imperfections, the issues of form and function again surfaced. The most successful images of imperfections to date have considered the functional aspects of how an imperfection was created. The use of wear-and-tear transformations was proposed as a general way to address this.

In essence, the chapter has moved from a purely 3D polygon model, and how to create best detail, to a position where there is a tension between whether 2D or 3D techniques are used to create detail. We can take this one stage further and briefly consider working solely in 2D. The most effective strategy for effecting shape change animation in 2D is morphing (e.g. [Beie92]), which has been used to produce some impressive results in film and TV. In fact, in recent years there has been a surge of interest in the use of 2D to cope with 3D scene complexity. Image-based rendering [Chen95, Debe96, Sei96, Leng97, Shad98], as the techniques can be collectively referred to, mainly attends to the equivalent effects that can be obtained in a standard 3D rigid body animation, such as walkthroughs of environments or movement of rigid objects. But this 2D to 2D process is in effect image warping, which is morphing, which can give the same perceived final results as 3D shape deformation. Thus it may be that the techniques developed for image-based rendering, such as splitting a scene into separate image layers, have a role to play in producing shape change animation.
Chapter 5.

Conclusions

This thesis has presented an investigation into the modelling-animation relationship, with shape change animation being a particular area of focus. The motivation was to bring together the mixed bag of modelling and motion control techniques produced by the computer graphics community, with the main goal being to lay better foundations for addressing the control of shape change animation. The main finding is that general, high-level, efficient and intuitive control of complex models remains elusive. Rather than becoming a tool for animators, computer animation has become a separate medium producing its own computer distinctive productions which are very different to manual animation. Almost certainly this is due to the failure of a modelling method to emerge which truly supports animation.

For offline animation, time and resources are not generally a problem, and a mixed bag approach to creating animation is adopted. Techniques can be used that may require extensive manual tweaking, but there seems to be no impetus to change this approach. Impressive work is produced, even if the amount of manpower used is excessive. In order to better understand the range of modelling and animation techniques available, Chapter 2 presented a taxonomy which was based on the general idea of relating the structural composition of an object to the refinement techniques used to mould it and alter its shape. Whilst the general efficient and intuitive control of complex models could be said to remain elusive, the exposition of form and function inherent in the categorisation of Chapter 2 suggests that the functional aspects of a model in relation to its structure need to be attended to more. Where this is done, as in the case of hierarchical structures, complexity is brought under better control.

We note that current techniques in computer animation carry a trade-off which results in a contradiction: low-level motion specification can be inordinately tedious (depending on the number of degrees of freedom in the model) but offers full expressive control, whereas high level specification is easier for an artist to understand and use, but generally results in less artistic freedom. This contradiction can only be resolved by offering the ability to ascend and descend a hierarchy to impose control at whatever level is required. However, this itself can cause problems if, for example, the autonomous behaviour of a character needs to be overridden. Instead more intuitive control of autonomous behaviour needs to be offered so that the animation process is similar to the director-actor relationship in, say, a film.
CHAPTER 5. CONCLUSIONS

Chapter 3 explored the specific application area of facial animation. FFDs, with their intuitive notion of visualisation of containment of deformation and ability to localise effects, controlled facial conformation and pseudomuscles, based on the natural driving forces of facial movement, were used for expression control. The FFDs could also deform the muscles so that the same muscles that were used on the undeformed face could be used on the deformed or conformed face. Thus a powerful synergy is offered, with two deformation techniques being used to give expressive control over any face. However, despite the effectiveness of the combination, the techniques still seem more suited to a stylistic face rather than a realistic face, as the movement doesn’t contain the subtlety of facial expression we are acutely attuned to. This is an indication that we have yet to fully address the realism of movement problem inherent in three-dimensional facial animation. A problem still remains over fine or subtle control of facial movement, since this inevitably leads to extra muscles and thus more parameters to control.

Added to this problem is the current way of building more realistic faces using a constructive layering approach, with eyes, teeth and hair added as extras. These features need integration into the facial model. A parameterised feature-based approach that relates parts of the face to each other could help so that one part of the face deforms in response to another, e.g. the tongue pushes out the face or the lips wrap around the teeth. The physically-based work of Terzopoulos and Waters would appear to offer the best way forward here. Indeed, integrating a more complex physically-based model based on facial biomechanics with Pelachaud’s work on behaviour has the potential to include all the facial animation complexity in a single model.

It is also worth noting that much of the facial animation work is locked into facial expression work. There is an apparent contradiction between scripting motion and the realism produced. In contrast, impressive facial animation work has been done by using photo-texture maps and morphing based on motion capture data or model-based coding in relation to video conferencing. Indeed, a number of computer games have effectively used motion capture pieces for facial animation, but, in essence, the technique is just a simple finite state machine based on limited inputs. The future is not in pre-recording. It must be in 3D. Such things as avatars in virtual reality worlds will need to exhibit autonomous behaviour and react to the infinite range of inputs from video images of users.

For interactive computer animation and, in particular, for computer games, the polygon mesh is the de facto modelling primitive. This brings its own problems with regard shape change animation, as one of the main problems to be addressed in games is how to provide adequate visual realism in a real-time interactive environment with its inherent frame rate constraints. Thus trade-offs must be made with regard to the modelling of surface detail. For instance people accept geometric aliasing more than image aliasing, specifically for polygon meshes. They don’t like aliasing at pixel level, but piecewise linearities along a silhouette edge are tolerated. But by addressing silhouette edges the apparent detail of a model can be increased for little cost. Coarse representations can be used for internal areas of objects, as interpolative shading algorithms make internal edges invisible and more polygons can be concentrated at silhouette edges to smooth piecewise boundaries.
CHAPTER 5. CONCLUSIONS

Another aspect of the need to produce best image quality for least cost is the tension between 2D and 3D. A mixture of geometry and texture can be used to create detail. Chapter 4 explored this in the context of a cloth skirt for computer games. A hybrid technique, with the polygon model responding to gross shape changes and fine detail layered on using texture maps gave effective results. However, in effect, this is also a failure, since the texture maps are not amenable to 3D control. They can only ever mimic the appearance of something. To fully represent, say, the creases in a skirt, they would have to be driven by the movement of the 3D polygon model. The extra cost in doing this could possibly be better employed increasing the geometric resolution of the model.

As the geometric resolution of any polygon mesh increases, we come up against another problem with polygons. They don’t admit shape change easily – high level tools must be used, but these can limit artistic control. Indeed, a profession of artists in both computer games work and film work has emerged whose primary function is to struggle and conquer these limitations, spending endless time tweaking vertices. The inertia of the polygon mesh in such situations is extremely high – due to hardware optimisations, very simple mathematics and the aforementioned fast shading, which makes internal edges invisible. But these successes are also its failures. We need to move away from the polygon as the basic modelling primitive.

Parametric patches and subdivision surfaces offer alternatives. In chapter 4, a brief comparison of polygons, Bezier patches and subdivision surfaces, with regards their use in computer games, indicated that the higher order primitives had a lot of advantages to offer. Indeed, over the last few years Pixar has moved from polygons to patches and now to subdivision surfaces [DeRo98] for use in offline computer animation. Whilst, for computer games, the polygon is still supreme, it would be interesting to see the impact of a higher level primitive being implemented on a graphics card.

In the discussion so far we have concentrated on the techniques used in modelling and animation. We have commented little on the interface issues in shape control work. The implementation of a user-friendly interface for both editing of models and control of complex animation remains an open problem. Most control techniques resort to the use of 2D to manage the complexity of the 3D operation required. Perhaps this is because of the common use of 2D input devices. Higher level input techniques are needed. In addressing this, Hegron and Amaldi [Hegr92] distinguish between two type of interface: symbolic interfaces and instrumental interfaces. For symbolic interfaces, which are language oriented and include standard 2D tools driven by mice and keyboards, work on gestures using 3D input devices such as the data glove and spaceball could be explored, so that the level of motion control could be raised to a more intuitive level. For instance, a sign language could be used to drive a character’s movement or the expressiveness of a gesture could be used to influence the expressiveness of a piece of animation. Instrumental interfaces are defined as physically-based interfaces, for example force feedback devices, that interact with multiple user senses. Such interfaces, in conjunction with immersive virtual environments, whilst still relatively immature as a discipline, could offer the opportunity of reducing the parameter clutter that can be a feature of complex control systems, by bringing the user parameter space and the computation parameter space closer together.
CHAPTER 5. CONCLUSIONS

Finally, a summary of the three general approaches to producing animation is offered. The first approach is the off-line creation of animation, where time is not at a premium. Here a mix of approaches to modelling and animation is undertaken, often with no particular reasoning behind a particular choice being made, except that it works for what it works for. Second, in interactive animation and, specifically, computer games, much use is made of pre-recorded animation pieces for such things as articulated structure movement and facial animation. In effect, these are finite state machines controlled by an interface. The limitations of this approach are obvious and it will inevitably change. This will be tied up with a move away from the route of increasing polygon model complexity to the use of higher level modelling primitives such as patches and subdivision surfaces, as they offer a higher level modelling-animation relationship. The third approach is the way forward. In future, more autonomy needs to be incorporated into computer animation work. Behaviour driven approaches will need to be further investigated so that they can replace the low-level tweaking of geometric detail. Natural behaviour and functional aspects of objects must be attended to in more detail.
Bibliography

References between the years 1990-94 are shaded in light grey.
References from 1995 to the present are shaded in dark grey.


Bard94 Ian Bardell, South Yorkshire police e-fit artist, personal communication.

Barr84 Barr, A.H., ”Global and local deformations of solid primitives”, Computer Graphics, 18(3), (Proc. SIGGRAPH 84), July 1984, pp.21-30


BIBLIOGRAPHY


BIBLIOGRAPHY

Carl82  Carlson, W.E., “Techniques for the generation of three-dimensional data for use in complex image synthesis”, PhD dissertation, Ohio State University, 1982 (as quoted in [Cobb84]).


<table>
<thead>
<tr>
<th>Reference</th>
<th>Title</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Title</td>
<td>Journal/Citation</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Faux, I.D.</td>
<td>“Simple cross-sectional designs based on Bezier patch surfaces and a comparison with indirect Bezier methods and traditional proportional development methods”</td>
<td>Proc. CAM 78 (National Engineering Laboratory, East Kilbride, Glasgow), 1978, paper 2.5.</td>
</tr>
<tr>
<td>French, L.</td>
<td>Toy Story, Cinefantastique 27, 2, 1995, pp. 36-37.</td>
<td>1995</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Title and Publication Details</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------</td>
<td></td>
</tr>
<tr>
<td>Author(s)</td>
<td>Title</td>
<td>Journal/Conference</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Klat87</td>
<td>Klatt, D.H., “Review of text-to-speech conversion for English”, JASA, 82(3), 1987, pp.737-793 (quoted in [Bhas94]).</td>
<td></td>
</tr>
</tbody>
</table>
BIBLIOGRAPHY


<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Title</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angeles, California, August 3-8, 1997</td>
<td>&quot;Smooth appearance for polygonal surfaces&quot;, The Visual Computer, 5, 160-173</td>
<td>1989</td>
</tr>
</tbody>
</table>
BIBLIOGRAPHY


BIBLIOGRAPHY


Sieb95 Siebert, J. P. and C. W. Urquhart, “C3D: a Novel Vision-Based 3-D Data Acquisition System”, in Paker, Y. and S. Wilbur (eds.) ‘Image Processing for Broadcast and
### BIBLIOGRAPHY

<table>
<thead>
<tr>
<th>Source</th>
<th>Reference</th>
</tr>
</thead>
</table>


Ware95 Waring, M.D. and S.Maddock (supervisor), “Imperfection Inclusion in an Evolutionary Environment”, Final year undergraduate dissertation project, Department of Computer Science, University of Sheffield, UK, 1995.
BIBLIOGRAPHY


