PhD

1989

E. H. Blake

Complexity in Natural Scenes:

A Viewer Centered Metric for Computing Adaptive Detail.

Edwin Haupt Blake

PhD Thesis.

Department of Computer Science, Queen Mary College, London University, Mile End Road, London, E1 4NS.

Abstract

Complexity in Natural Scenes:
A Viewer Centered Metric for Computing Adaptive Detail.

Natural scenes are complex and have seemingly limitless detail. Animation compounds these problems. This dissertation describes research into controlling the complexity by imposing a *priority metric*. The metric ensures that only visually relevant detail is processed and presented to the viewer. Two kinds of metric are identified: a *static metric* for spatial detail, and a *dynamic metric* which measures detail in time.

Theories of the way the natural world is seen, particularly optic flow perception, are integrated with accounts of the effects of computer displays. The mathematical formulation of the priority metrics is based on this synthesis.

The object oriented approach to computer animation is critically investigated. This programming paradigm provides the most appropriate abstraction for modelling the elements of the environment and their interactions. If physical objects are to be modelled it should be extended to include part-whole hierarchies. This approach, together with the priority metrics, allow different hierarchical representations to be used together.

Three experimental implementations were used to investigate the metrics:

a) A stick figure, with its non-uniform hierarchical structure, provided the first test. The Smalltalk implementation demonstrated adaptive detail effects which depended on the spatial priority metric.

b) A uniform hierarchy of detail levels, fractal surfaces, were implemented in C++ and rendered on a shaded display. The metric allowed faster rendering with smaller memory requirements than non-adaptive algorithms.

c) The dynamic metric was used with textured images moving in three-dimensions to limit motion detail. Animation sequences were synthesized using the various orders of optic flow effects, without continual reference to a three-dimensional model.

Computational complexity was controlled by adapting accuracy to the evolving demands of the viewer. This resulted in faster animation and smaller memory requirements.

Table of Contents

Chapter I — Introduction. 11

§1.1 Aims of the Study: Dealing with complexity. 11

§1.2 Introducing the Spatial and Temporal Priority Metric. 14

§1.3 Previous Work on Representing Objects with Adaptive Detail. 16

§1.4 Metrics for Computing Adaptive Detail: Summary. 24

Chapter II — The Appearance of Natural Scenes: a viewer centered approach. . . 33

§2.1 Scene Simulation and the Problem of Realism. 33

§2.2 Describing Natural Scenes 39

§2.3 Image ~ Object Relationships and the Optical Transfer Function. 45

§2.4 The 'Atmosphere'-VDU-Eye transfer function. 53

§2.5 Information Channels: From Computer Model to Viewer's Mind. 61

§2.6 Summary and Conclusion. 65

Chapter III — Object Oriented Representation in Animation. 67

§3.1 Structuring Complex Programs and Data. 67

§3.2 Animating Jointed Figures. 74

§3.3 Smalltalk and C++. 79

§3.4 Representing Physical Objects: The part hierarchy. 83

§3.5 Experience with Using the Part Hierarchy. 89

§3.6 Implications of the Part Hierarchy for Object Oriented Languages 91

§3.7 Conclusion. 92

Chapter IV — The Formulation of the Spatial and Temporal Metrics. 94

§4.1 Introduction: What is Detail? 94

§4.2 The Spatial Metric. 101
§4.3 The Temporal Metric. 104
§4.4 Data Types. 114
§4.5 Conclusion: Practical Priority Metrics for Animation. 116

Chapter V — The First Experiment: Using the static metric on a model with a non-uniform hierarchy. 118
§5.1 Discrete Detail and a Continuous Metric. 118
§5.2 Implementing the System of Priorities. 123
§5.3 Results from Running the Experimental Implementation. 129
§5.4 Conclusion. 131

Chapter VI — The Second Experiment: The spatial metric applied to continuous detail levels. 132
§6.1 Introduction: Digital Representation of Landscapes. 132
§6.2 Synthesizing Pictures of Landscapes. 133
§6.3 Rendering Landscapes Efficiently on a Raster Display. 141
§6.4 The Results of using the Spatial Metric. 148
§6.5 Conclusion. 153

Chapter VII — The Third Experiment: Priority based execution of motion. . . . 159
§7.1 Moving Figures and the Dynamic Metric. 159
§7.2 Approximating Projected 3-D Motion with Optic Flow Effects. 161
§7.3 A Simple Priority Based Animation System. 174
§7.4 Results of Using the Temporal Priority Metric for Animation. 178
§7.5 Conclusion. 182

Chapter VIII — Conclusion. 188
§8.1 The Main Implications of the Space-Time Priority Metric. 188
§8.2 Future Extensions. 195

Appendix A — Implementing a Part-Whole Hierarchy in Smalltalk. 200

Appendix B — Timings of the Continuous Spatial Metric Experiment. 202

Appendix C — Timings of the Dynamic Metric Experiment. 207

Appendix D — Concurrent Object Oriented Animation. 212

§D.1 Concurrent Object Oriented Animation: A research proposal. 212

§D.2 Controlling Figure Animation. 213

§D.3 A System for Concurrent Figure Animation without Feedback. 214

§D.4 A System for Concurrent Figure Animation with Feedback. 216

§D.5 Conclusion. 217

Bibliography 219

Tables

Table 2.1. — Steps in the transmission of information to the viewer. 62
Table 2.2. — Summary of the properties of the spatial and temporal metrics. . . . 66
Table 6.1 — Typical results of profiling with a regular (sine) height field. 150
Table 6.2 — Summary of Timings from Appendix B for 512×512 Images. . . . 151
Table 7.1 — Summary of timings from Appendix C. 180

List of Figures

Frontispiece — 'Joe': The Stick Figure. 10

Figure 2.1. — Demonstration of perspective distortion by Leonardo da Vinci. . . . 36

Figure 2.2. — The ambient optic array. 41

Figure 2.3. — Optic flow. 42

Figure 4.1. — Scale Space Filtering. 97

Figure 4.2. — Perspective projection of a re-inverted pinhole camera. 106

Figure 4.3. — Increased detail by further generation of a fractal. 115

Figure 5.1. — The hierarchy of detail levels in representing a human figure. 120

Figure 5.2. — The part hierarchy for the stick figure. 121

Figure 5.3. — The subclass hierarchy of the class ObjectWithParts. 124

Figure 5.4. — Accessing the part-whole hierarchy with a compound message. . . . 127

Figure 5.5. — Stills from a sequence of the camera flying in to the right hand. . . . 128

Figure 6.1. — Visibility relations on the unit sphere. 136

Figure 6.2. — Recursive subdivision of a square into quadrants. 139

Figure 6.3. — Occlusion compatible ordering with quadcodes. 143

Figure 6.4. — Interactive fractal generation. 147

Figure 6.5. — Facet plan view. 149

Figure 6.6 — The inverse relation between facet size and execution time. 151

Figure 6.7. — Fractal mountain at high resolution. 155

Figure 6.8. — Fractal mountain viewed at two different resolutions. 156

Figure 6.9. — Still of the process of fractal generation. 157

Figure 6.10. — The fractal generation method applied to a sine wave field. 158

Figure 7.1. — "Re-inverted" image projection. 165

Figure 7.2. — Rotation by multiple shearing of an image. 173

Figure 7.3. — The part hierarchy of selected classes in the animation system. . . . 175

Figure 7.4. — Part of the class hierarchy of the animation system. 176

Figure 7.5. — Colour effects of smoothing pixels in a colour table. 183

Figure 7.6: a & b. — Multiple actors executing different motions. 184

Figure 7.7: a-f. — Frames from a sequence of distorting 2-D pictures which mimic 3-D movement. 185

Figure 8.1. — Filtering high frequencies of a “stick”. 197

Acknowledgements

I would like to thank Mel Slater and Steve Cook for their helpful comments on earlier versions of particular chapters. I am very grateful for the assistance William Roberts and Tony Lin provided in producing the illustrations for this thesis.

Dr. Hilary Buxton provided a great deal of help during the research and endless encouragement in producing this dissertation.

My thanks to you all, and to the other members of the Computer Science Department at QMC.

thank you josje

Frontispiece ‘Joe’: The stick figure used in examples on the part-whole hierarchy and adaptive detail with a non-uniform hierarchy. Not all parts are fully enumerated, for example the toes have been suppressed and many of the fingers. This is due to the action of the spatial priority metric for adaptive detail.

Chapter I

Introduction.

The issue of computational complexity in the animation of natural scenes is introduced. This provides an outline of the purpose of the study and the context in which the research was conducted. The viewer centered *metric* for computing adaptive detail is presented qualitatively as a methodical way of using adaptive detail in animated scenes. Object oriented programming is introduced and adapted to deal with the computational complexity of interactive computer graphics and computer animation. This methodology helps in thinking about, designing, and implementing these computer systems.

The basic literature in computer graphics on which this research depends, and which it extends, is surveyed. This includes previous approaches to adaptive detail and various hierarchical data structures. The chapter concludes with a summary and an outline of the rest of the thesis.

§1.1 Aims of the Study: Dealing with complexity.

Natural scenes are well known to be very complex and to have seemingly limitless detail. The problem we face is to synthesize pictures based on a three-dimensional representation of the natural environment. Objects will move in this environment and we also have observers which move about amongst the richly textured fields.

The principal goal of this research was to formulate measures of detail for such scenes which would reduce the computational burden of synthesizing the pictures. The original idea was that picture synthesis should be regarded from the point of view of the final observer and not from the point of view of the modelled object. This shifts the emphasis from realistic object models to production of convincing pictures.

The basic viewer-centered approach was then generalized. The aim is to provide appropriate information at all levels of communication between the entities in an animation system. Redundant information at a particular level should be hidden. Adaptive detail becomes the special case in which a modelled object hides redundant information from a synthetic camera. The result of this research has been to formulate two *priority metrics* and a *methodology* which can be applied to animated three-dimensional representations of the natural environment.

The *metrics* measure the importance to the viewer of both spatial and temporal features of the changing scene. Irrelevant detail can then be eliminated from the picture with

predictable visual consequences. This reduces the complexity of the computation and results in faster execution and the use of less memory.

The *methodology* for tackling these animation problems is based on object oriented programming. The main extension to the general paradigm was modelling actors with a *whole-part hierarchy* and distinguishing this from the normal hierarchy of object types (the class hierarchy). The viewer centered approach added a new data type to the model of an actor: *an Appearance*. The Appearance mediates between an object and a synthetic camera so as to minimize the computer resources required to render the object.

The prime purpose and most important result of the experimental work conducted has been to test the application of the detail measures. There are two priority metrics: a *static (or spatial) metric* and a *dynamic (or temporal) metric*. Their use and benefit was tested in three implementations:

(1) simple mobile stick figures with a moving camera,

(2) multiple views on a static textured landscape, and,

(3) continuous motion of highly detailed objects.

In these implementations certain key features of natural scenes, that is, animated landscapes with moving figures and a changing point of view, were picked out. The choice of features was guided by an analysis of the perception of natural scenes. The features incorporated into the experiments were: textured fields receding into the distance with a gradual loss of detail and animals represented as moving jointed stick figures. Movement is conveyed by synthesizing a sequence of textured pictures corresponding to the sequence of incremental times.

Another purpose of the experimental implementations arose form the fact that computation for graphics and animation is complex not just in an *algorithmic* sense but also in a *programming* sense — if we can entertain this distinction for the moment. By this is meant that, quite apart from the complexity of the algorithms used, another kind of *programming* complexity arises due to the scale of the overall system. This is because animation systems are typically very large integrated programs making use of a wide range of techniques. They deal with large and disparate databases which represent the environment. They simulate the concurrent interaction between a large number of actors. Typically a number of output

formats are possible and in many cases they allow for real time interaction with a user. Systems which represent natural scenes are found in aircraft simulators, in video games and the production of special cinematic effects.

Object oriented programming developed from simulation languages and from projects to manage programming complexity. Object oriented programming provides a natural way of dealing with the concurrent actions of entities which have to be modelled in animation. The principles of data abstraction which are embodied by object oriented languages allow a number of different internal representations to coexist. All information is stored in terms of active objects with an internal state (actors) and this seems to appeal to the humans which have to program and use the systems interactively.

The use of object oriented programming in the implementations has largely been vindicated, although the need for a part hierarchy as an integral part of any language for representing physical objects was identified and added into Smalltalk. Smalltalk proved useful for prototyping and for the complex modelling required for jointed figures. The need for speed (albeit with simpler textured fields) was met by using C++. The type checking and type hierarchies of C++ allowed the fairly large programs required for the second and third experiments (~ 10,000 lines of code, 40,000 words) to be developed quickly.

Some other points which also emerged from this project are summarized in §1.4.1.

The single unifying idea which lies at the heart of this research project is that there is an optimal information content at the various levels of interaction in a (human-)computer system. When object modelling is approached in this manner one even finds details in a model which were included for physical realism but never used in practice, such unused detail can be dispensed with altogether.

This can also be called a *subjective approach to computation*, that is, entities should exist only for the benefit of the user; whether that user is a programmer or the final viewer of the results. When applied to programming a computer animation system this means that the data and programming structures should closely reflect the way the programmer thinks about the problem: actors moving about and interacting in a three-dimensional environment. When applied to complexity in generating the images it is known as the viewer centered approach to graphical computation.

Occam's razor now becomes:

> ''Entities are not to be multiplied beyond those required to convince the viewer''.

The rest of this dissertation shows how these ideas can be made precise and refines them to be practical tools for producing pictures. In doing this we shall be providing a theoretical basis for some of the existing techniques used to ''fake'' pictures by making them appear to contain more physical information than is actually modelled.

§1.2 Introducing the Spatial and Temporal Priority Metric.

We want to produce pictures that are convincing for the purpose for which they are made; this purpose could be producing a 'realistic' science fiction film or a 'realistic' flight simulator. In neither case does such realism have to imply the simulation of the physical laws of nature [Reeves, 1987; Schachter, 1981]. Our sole aim is to induce in the viewer a sense of conviction about the validity of the scene: a willing suspension of disbelief†. We shall return to this question in Chapter 2.

Seen in this light, *detail* in shape representation ought to exist only to the extent necessary for convincing the viewer. This could be achieved by using hierarchical data structures. Detail would then be increased by progressing to further hierarchical levels. Numerous such data structures have been proposed. It has been largely an unsolved problem however to create a relation automatically between the data structures and the rendering process so that exactly the necessary level of detail is available [Badler & Carlbom, 1984‡].

First we will introduce the approach taken in this research and then in the next subsection we shall examine previous work in this area (§1.3). The metric is introduced now in order to motivate the body of theory which has to precede its mathematical formulation. This theory deals with how we actually see nature. The mathematical formulation of the metric depends on ideas taking from vision research and sampling theory.

---

† Coleridge [1817] discusses how the power of poetry can arise either from a faithful adherence to nature or using imagination: ''so as to transfer from our inward nature a human interest and a semblance of truth sufficient to procure for these shadows of imagination that willing suspension of disbelief for the moment, which constitutes poetic faith.''

‡ ''... methods of building object models with several levels of detail (selected according to the size of its image on the display screen) have been proposed, but automatic generation of such hierarchies requires further research.''

A good example of one kind of problem which we want to tackle is described in the following observation of Leonardo da Vinci [in MacCurdy, 1954, p351]:

> In every figure placed at a great distance you lose first the knowledge of its most minute parts, and preserve to the last that of the larger parts, losing, however, the perception of all their extremities; and they become oval or spherical in shape, and their boundaries are indistinct.

This kind of figure is the subject of the first experimental implementation and is discussed in Chapter 5. Some implications for object oriented programming arising from modelling such figures are mentioned in Chapter 3.

The hierarchical detail representations are related to one another by means of a priority metric. There are two separate metrics that can be defined in an animated scene. The intuitive notions are:

Spatial (static) Priority.

> Those objects further away from the view point are visually less important to the picture being generated than those closer by.

Temporal (dynamic) Priority.

> Objects moving quickly with respect to the observer need to be redrawn more often than those at relative rest.

These simple ideas are extended and mathematically formulated so that they can be applied. The extensions include allowance for:

- Atmospheric effects (by means of a weighting of the distance).
- Trade-offs between temporal and spatial resolution in human vision.

The spatial priority metric, even allowing for atmospheric effects, is relatively straight forward: at least it can be resolved into a single measure. The temporal priority of an object is more complex since there are several ways in which the moving image of an object can change from one frame of an animated sequence to another. A sound theoretical basis for these changes can be provided by optic flow theory (see Chapters 2 and 4).

The trade-off between spatial and temporal detail still largely lacks a computationally applicable theory: although some relations between spatial frequency and temporal frequency can be given (see Chapter 2).

With these qualitative ideas as motivation we can now examine the relevant literature in computer graphics.

§1.3 Previous Work on Representing Objects with Adaptive Detail.

One of the earliest problems for picture synthesis was hidden surface removal; once satisfactory algorithms had emerged for solving this, and with the advent of raster displays the pursuit of realistic pictures could begin in earnest. Subsequently synthesis techniques proceeded on three fronts [Amanatides, 1987; Magnenat-Thalmann & Thalmann, 1987]: (1) rendering and the effects of light, (2) modelling and the representation detail, and, (3) sampling, which links the picture to the model.

This research makes a contribution to object representation, but the means adopted straddles all three fronts. In this section we will examine the literature in a number of areas of computer graphics because we depend on results from these areas. The immediate aim is not, however, to produce better anti-aliased or ray traced images, but to show that the way our images are viewed has important implications for the way they are represented. Another important aim of this section is to bring out a common underlying theme of adaptive viewer-centered detail. This theme can be found even when the authors of a technique lay the emphasis on their adherence to physical reality, as for example, in so many papers on ray tracing.

Warnock's hidden surface algorithm is an early example of a technique which adapts itself according to the visible detail in a picture [Sutherland, Sproull & Schumacker, 1974]. It proceeds by splitting the image up into smaller and smaller pieces until the pieces become simple or the size is reduced to a single pixel. From Rogers [1985, †] it is apparent that the algorithm resulted from the application of a viewer-centered approximation approach. Warnock's algorithm is a clear example of divide-and-conquer adaptively applied and the resulting data structures are rather similar to quadtrees [e.g., Samet, 1984]. This basic idea recurs in the application of the spatial detail metric to various data structures.

---

† ''... basic ideas behind the Warnock algorithm are very general. They are, by analogy, based on an hypothesis of how the human eye-brain combination processes information contained in the scene. The hypothesis is that very little time or effort is expended on areas that contain little information. The majority of the time and effort is spent on areas of high information content. ... the area of interest narrows, and the level of detail sought increases.''

Clark [1976] sets the goal of a single unified structuring of the three-dimensional environment using hierarchical models. The normal object hierarchy [Foley & van Dam, 1982] is extended to include sub-hierarchies which contain objects modelled in greater and greater detail. Searches and traversals proceed only down to the smallest resolvable level of detail. Visible surface algorithms can use bounding volumes to achieve logarithmic dependence on complexity.

Clark mentions the ‘‘interesting possibilities’’ of having full detail at the centre of attention of the scene and also of making the amount of detail inversely dependent on the speed of motion. In Chapter 2 we shall see that a theoretical basis for the computation of trade-offs between spatial and temporal detail are only now emerging.

Another important contribution of Clark is the notion of the ‘‘graphical working set’’ which is the set of objects in view at any one time. In animation we shall show that the idea of a working set can be extended. We shall distinguish between visible objects according to the nature of their motion relative to the viewer (see also §1.3.5). These ideas find their eventual expression in the temporal priority metric.

Rubin & Whitted [1980] developed a homogeneous hierarchical representation of objects based on a uniform spatial enumeration; nodes could be procedurally generated. The entire representation contained nothing but bounding volumes. At the initial levels the bounding volumes were arbitrary parallelepipeds, while at lower levels they were subdivided to become very much the now familiar octrees [Meagher, 1982]. As with Clark (above), the determination of which objects are visible became a logarithmic search of the object space. The nodes could be explicitly stored or else generated only when needed.

The problems with a uniform spatial enumeration are two-fold: firstly it forces all objects to be reduced to a uniform representation which is difficult to model directly, and secondly, these data structures imply a sampling of the object space which means that, in general, motion involves an expensive resampling process (see also §1.3.3). We are looking for a more general solution. With a detail metric any abstract data type which provides the common protocol (or functional interface) can be incorporated. No common underlying representation is required and no uniform sampling of object space is implied.

Rubin [1982] briefly described a scheme where the representation is expanded to allow geometric transformations between nodes. The hierarchy consists only of parallelepipeds,

but each has an associated transformation matrix which allows arbitrary scaling and rotation. Unlike octree encoding there is a system of distributed local coordinates. The use of distributed local coordinates greatly simplifies the modelling task and reduces the global dependence of objects on each other. To prevent scintillation small objects are not removed at the resolution limit but are faded according to a simple heuristic. Fading is an example of the way in which loss of spatial detail can be approximated on a raster display.

More recently, work in interactive graphics has introduced the idea of adaptive refinement of images to better and better quality levels [Forrest, 1985; Bergman, Fuchs et al. 1986]. Images on a workstation, or images transmitted over slow transmission lines, are rendered quickly to a coarse approximation and then refined while the user examines the image. The cost of good anti-aliasing is incurred only after the user has been given some idea of what the image looks like. The problem which we address is related but rather different: we would like to identify automatically those objects where the loss of detail will never be noticed. For moving textured objects of the types commonly found in natural scenes, some form of anti-aliasing cannot be avoided.

1.3.1 Limiting the Complexity of Ray Tracing.

Most complex and realistic scenes are produced by ray tracing. This is a way of sampling the multidimensional simulated environment by applying the laws of geometrical optics. The basic algorithm is very simple and very expensive, it depends on the squared complexity of the modelled environment. The high computational burden arises from intersecting millions of rays with the surfaces in the modelled environment.

Reducing the number of ray/surface intersection calculations by means of hierarchical bounding volumes has received a great deal of attention. The first implementation of Clark's ideas to ray tracing was by Whitted [1980]. Methods of producing a hierarchical description of the environment were developed in Weghorst, Hooper & Greenberg [1984]. The number of studies still produced in this area reflect its interest and importance [Bouville, 1985; Fujimoto et al., 1986; Glassner, 1984; Goldsmith & Salmon, 1987; Jansen, 1986; Kay & Kajiya, 1986; Snyder & Barr, 1987]. The sampling density required depends on the scene information content and the display resolution and the animation frame rate.

Ray tracing is actually a general technique which can be applied to sampling the environment along many dimensions. Stochastic sampling is an elegant generalization of ray

tracing which can be used to reduce the samples required in both space and time [Cook, 1986; Cook, Porter & Carpenter, 1984; Dippé & Wold, 1985; Lee, Redner & Uselton, 1985]. The model remains undersampled and the success of stochastic sampling depends to a large extent on a particular feature of human vision: we are much less sensitive to noise than to to coherent aliasing.

Ray tracing produces very realistic pictures but to achieve that physical realism the complexity of the problem is reduced as much as possible. Firstly, objects are merged hierarchically into larger groups. Secondly, sampling requirements are relaxed when this will not be noticed by the human eye. Even with these techniques ray tracing remains expensive. We would like to have a framework which subsumes it in a more general array of methods which can be called upon as needed.

1.3.2 Variable Resolution Texture Mapping.

Some of the ways in which texture mapping is done are also of interest. Texture mapping is a technique which allows relatively simple three-dimensional objects to appear very complex [survey — Heckbert, 1986]. Texture mapping can be used as a cheap alternative to ray tracing in complex natural environments [Cook, Carpenter & Catmull, 1987]. Flat textures are projected onto the surface of the three-dimensional object. In order to perform texture mapping one has to apply geometrical transformations and prevent aliasing artifacts.

From the point of view of the proposed priority metrics, we are interested in the adaptive detail filtering techniques associated with texture mapping, rather than the geometrical transformations. Although in applying the temporal metric to moving objects there are often cases where the image of an object is a slightly distorted version of the one from the previous frame.

Something somewhat akin to texture mapping can be used to generate images from one frame to the next directly from the two-dimensional image; that is, a frame can be regarded in a first approximation as a two-dimensional affine mapping of the previous frame. But texture mapping of itself does not provide an automatic method of making these frames. The whole theory which relates the distortion of the image of a three-dimensional object from one frame to the next has still to be provided. Thus we adapt some of the geometrical transformation techniques for the experimental implementation described in Chapter 7 [Braccini & Marino, 1980; Catmull & Smith, 1980; Weiman, 1980].

As the scale of the object changes the sampling of the texture also changes. To avoid aliasing the texture should be filtered to remove frequencies higher than the sampling rate. To avoid doing this repeatedly textures can be prefiltered to various resolutions and organized hierarchically [Williams, 1983: ‘‘Pyramidal Parametrics.’’; Crow, 1984: ‘‘Summed-Area Tables’’; Glassner, 1986]. These hierarchies store the texture maps as a series of detail levels.

We are going to be defining hierarchies of detail for natural scenes. These hierarchies will be applied to models of objects (not just their textures) and will apply to both space and time dimensions.

1.3.3 Adaptive Detail with Octrees.

Octrees were independently developed by a number of researchers as a generalization of quadtrees (quadtrees themselves are a two-dimensional generalization of binary trees which are used for encoding images) [Meagher 1982]. Octrees can represent three-dimensional objects to any specified resolution. The representation enumerates space into cubical cells stored in a tree structure. The size of the cells decreases geometrically with increasing depth on the tree. Octrees are simple uniform representations and so allow parallel hardware implementations for real-time graphics [Meagher, 1984; Oliver, 1986].

Octrees are generated by a machine, they are not a representation people use. Spatial enumeration is automatically done by some medical imaging systems but otherwise the octrees must be generated from other representations (e.g., CSG [Samet & Tamminen, 1985]). Octrees can be easily translated and rotated through 90 degrees [Jackins & Tanimoto, 1980], but for animation the disadvantage is that rotating a general octree encoded object through arbitrary angles is difficult (being essentially a conversion from one sampling of space to another, with attendant aliasing problems) and is usually performed by recomputing the octree after rotating the original representation. Similar sampling problems appear in shapes with complex detail [Meagher, 1982, Ahuja & Nash, 1984, Weng & Ahuja, 1987].

Octrees do allow the detail in the projected image to be varied by changing the depth to which the octree is processed. This can be done adaptively depending on the resolution of the display using a ray tracing rendering algorithm [Sandor, 1985].

In this study we shall be abstracting the general features of recursive subdivision. By keeping our definition abstract, and by not specifying exactly what the underlying data

representation must be, we can avoid the rigidities of octree space subdivision. We shall therefore define recursive detail data structures in terms of their response to the static priority metric. The characterization hierarchical detail data structures is not by specifying some underlying concrete implementation, but by specifying the functional interface to the abstract hierarchical detail type.

1.3.4 Hierarchical Data Structures for Adaptive Detail.

At this stage it is useful to distinguish between *geometrical hierarchies* which explicitly contain all levels of detail as geometrical models, and those *procedural hierarchies* whose most primitive elements are procedures which know how to render the primitive on a display. A third category, *homogeneous hierarchies*, must be introduced to deal with those hierarchies where the distinction is less clear, like the octrees presented in the previous subsection. Such hierarchies have a completely uniform representation of space (also know as spatial enumeration or decomposition) which can be subdivided until the sub-model is atomic , i.e., it can no longer be resolved. Such primitives can then be directly rendered.

There are very many representations which can be stored in a hierarchical fashion:

1) D. Marr's generalized cylinder models of people and animals [Marr & Nishihara, 1978; Marr & Vaina, 1982] arise from work in model based vision.

   - Many shapes have inherent axes. Pipe cleaner (stick) people and animals are very convincing despite their simplicity.
   - They need object centered coordinate systems which are distributed over the figure so that each resolvable part has its own coordinate system.

2) B.B. Mandelbrot's fractals [Mandelbrot, 1982b; Fournier, Fussell & Carpenter, 1982; Pentland, 1984 & 1985; Miller, 1986; Oppenheimer, 1986; Voss, 1985]

   - rough natural surfaces are best described by fractals.
   - Fractal surfaces are statistically similar at all levels of detail.
   - The range of objects which can be represented by fractals complements those represented by generalized cylinder models. Eroded hillsides are an example of such surfaces,

3) Procedural branching models [Aono & Kunii, 1984; A.R. Smith, 1984]. These models yield very realistic trees and plants.

4) Particle systems [Reeves, 1983; Kajiya & von Herzen, 1984; Reeves & Blau, 1985] are stochastic systems which can represent things like dust and fires.

5) Procedural approaches can be generalized by developing special languages for describing shapes and textures which can be applied in a recursive fashion [Henderson, 1982; Perlin, 1985; Beyer & Friedell, 1987].

As we shall see in the coming chapters the main thrust of the research is to develop the metrics of adaptive detail. Concrete implementations of the abstract hierarchical data types will be used to test the theory. The first and second experiments make use of the first and second types of hierarchy respectively. For the third experiment we build a procedural hierarchy of temporal detail levels directly on top of two-dimensional rasters.

1.3.5 Dynamic Effects: Frame-to-frame Coherence and Motion Blur.

Moving from static image synthesis to dynamic image synthesis we can extend the notion of object coherence, which gave us the hierarchical bounding boxes, to frame-to-frame coherence. We create dynamic effects by synthesizing a sequence of still frames. From one frame to the next the picture does not change very much and so it seems attractive to use the information about the previous scene when calculating the next frame.

This idea was first used in Schumacker's hidden surface algorithm [Sutherland, Sproull & Schumacker, 1974] developed for use in real-time flight simulation. In this algorithm the relative priority of faces of planar objects (which must be convex and linearly separable), are precalculated, possibly with human intervention. A less restrictive frame-to-frame coherence algorithm was presented by Fuchs, Kedem & Naylor [1980], and an implementation described [Fuchs, Abram & Grant, 1983]. The application is effectively limited to using static environments because all visibility relations have to be recalculated when the environment changes.

The coherence experienced by an observer moving through a static environment is described in Hubschman & Zucker [1982]. This is a definitive analysis of the work on frame-to-frame coherence for static scenes consisting of stationary, closed, convex, non-intersecting polyhedra. Shelley & Greenberg [1982] provide a very practical implementation for interactively

specifying the path of the moving observer through a static, linearly separable, polygonal environment. By examining the sequence of positions and view directions along the path of the observer, notions of frame-to-frame coherence can be applied as preprocessing steps before rendering is attempted. In forthcoming chapters the idea of frame-to-frame coherence will be formalized in terms of the equations of optic flow. Optic flow applies to all kinds of environments and also applies to all possible moving images (e.g., the continuously changing ones picked up by the human visual system) not just sequences of coherent ‘‘frames’’.

Recently Clark’s idea of a graphical working set (§1.3) has been extended to dynamic environments [Hegron, 1987]. The problem is to maintain a dynamic database of visible objects as the synthetic camera ranges over the environment. Some data about the speed-up resulting from using such algorithms is presented in Crocker [1987].

The ‘‘converse’’ of frame-to-frame coherence is motion blur. It is the converse in the sense that frame-to-frame coherence is used with relatively static objects. Fast moving objects, on the other hand, are blurred by their motion. In 1983 a number of papers appeared in the graphics literature which dealt with the topic of temporal anti-aliasing or motion blur [Korein & Badler, 1983; Potmesil & Chakravarty, 1983; and Reeves, 1983]. Optic flow analysis can equally be applied to fast moving objects to give a measure of the blurring which would result from their motion through the image.

In keeping with our viewer centered approach we shall also be approaching this problem from the subjective point of view. Optic flow, which has already been mentioned above, can be described as the way a participant experiences these motion effects. This theory will be presented in the next chapter and the mathematics developed in Chapter 4. In applying optic flow to the analysis of moving images we will be able to characterize various orders of frame-to-frame coherence (see Chapter 7). These orders of coherence can be used to divide the objects into a number of classes. The graphical working set can then be further refined so that it distinguishes not just the fact of visibility, but also the ‘‘importance’’ of the movement.

§1.4 Metrics for Computing Adaptive Detail: Summary.

The basic thesis of this research programme is that space and time priority metrics may be defined for a moving three-dimensional modelled object. These metrics measure the spatial and temporal detail which will be visible to the viewer of a synthesized picture containing that object. These priority metrics allow the determination of the importance of the objects to be made automatically. The properties of objects which allow such adaptive detail may be abstractly defined in terms of their response to the priority metrics, and without requiring the existence of a uniform underlying primitive representation.

The previous subsection surveyed the ways in which adaptive detail has been used in computer graphics and computer animation. Procedural models are used in many of these cases because they simplify the representation of detail where it can be structured in some way. The divide-and-conquer approach was a natural companion of the hierarchical data structures.

It is quite apparent that there are very many techniques with a common aim of simplifying the ultimate computational complexity. It would seem very useful to formulate a metric which allows one to relate the various techniques to one another and a programming methodology which could provide a uniform interface and hide the internal representations when these are irrelevant to the computational task.

In the context of computer science our thesis is that the computational complexity of animation systems may be reduced by applying the spatial and temporal priority metrics. This benefit is felt both in storage requirements and in execution times. We will also argue that object oriented programming is particularly suited to thinking about and writing computer animation programs.

1.4.1 Other Results.

This chapter has been concerned with introducing the main thesis. A few other subsidiary points also emerged from this project. They form minor themes which were interesting but which were not directly relevant to the arguments presented so far. It is useful to mention a few now.

Quaternions have distinct advantages over matrices in describing the coordinate transformations required in animation and they are very easily made an integral part of the object oriented languages used.

Another point which emerged was the conceptual advantage of working on projections onto the *unit sphere* rather than planar perspective projection. The unit sphere description is often closer to the subjective experience of the observer than is a description involving the image plane.

Although *Fourier techniques* formed the basis of many theoretical analyses, the final algorithms invariably avoided the use of discrete Fourier analysis.

Some new results regarding *quadcodes* were used in the recursive subdivision of fractal landscapes to derive limits on storage requirements.

A fast one-pass method of achieving general *two-dimensional transformations of images* formed part of the implementation of the dynamic detail metric.

A number of these results concern the conceptual advantages of one approach over another, while others are more to do with practical spin-offs in terms of a new algorithm or theorem. A theme which also clearly emerges, in Chapter 2 and elsewhere, is an analysis of what *realism* means in computer graphics.

1.4.2 Outline of the Rest of the Dissertation.

This is an investigation into an integrated approach to dealing with complexity in computer graphics and computer animation. A very general approach is refined and made applicable by tackling representative problems. So Chapters 4, 5 and 6 contain discussions of fully worked out experimental computer implementations of ideas which first make their appearance here, in this chapter, as general ways for regarding and understanding problems. In the intervening chapters the theory to fit the general paradigm is developed. These theories are used to analyse problems and provide mathematical formulations.

The introduction has set the scene and introduced the priority metric as a methodical way of using adaptive detail in animated scenes, as opposed to the disparate collection of techniques reviewed in the previous section (§1.3). That was the first of three literature surveys which appear in the first three chapters. The next chapter, Chapter 2, deals with natural scenes and how they appear. This allows us to extract the essential features of natural scenes as the basis for the experimental implementations. Chapter 3 then contains the introduction to object oriented animation and a survey of the progress in functional animation. It also gives the first original result: the clear difference which exists between class hierarchies and

whole-part hierarchies. In Chapter 4 we can finally formulate the priority metric and discuss the data types which the implementations will require.

The three Chapters devoted to computer implementations and experimental results follow. Finally, in Chapter 8, there is a discussion of the results and the conclusion. The conclusion and Appendix D also outline some specific proposals for further research.

1.4.3 Annotated Table of Contents of the Remaining Chapters.

2. The Appearance of Natural Scenes: a viewer centered approach.
   This chapter answers the question: ‘‘What essential information about natural scenes would have to be simulated to provide convincing animated pictures?’’. This chapter not only surveys a great deal of vision literature but gives an original synthesis of that body of theory for the purposes of computer animation.

   2.1 Scene Simulation and the Problem of Realism.
   Perspective, Art, Naturalism & Computer Graphics. The question answered here is: What are ‘‘convincing animated pictures’’?

   2.2 Describing Natural Scenes.
   We survey very briefly: Gibson’s Ecological Optics, Morphology (Form of animals, trees, grass, etc.), Texture and Fuzzy Objects (Fractal description of nature). The introduction to Optic Flow is an important part of this subsection.

   2.3 Image ~ Object Relationships and the Optical Transfer Function.
   The Fourier analysis of moving image formation is presented here.

   2.4 The ‘Atmosphere’-VDU-Eye transfer function.
   We discuss the filtering characteristics of the human eye.

   2.5 Information Channels: From Computer Model to Viewer’s Mind.
   The preceding subsections provided the theoretical basis for describing what has to be simulated in order to generate good computer animation. This subsection draws the conclusions concerning the spatial and temporal metrics.

   2.6 Summary and Conclusion.

3 Object Oriented Representation in Animation.

3.1 Structuring Complex Programs and Data.

(or: Data Abstraction is a ‘‘good thing’’.) We introduce the notion of object oriented or actor based programming. We discuss instances and subclasses, Actor/Message passing and overloading of operators. It contains a literature survey on the use of actors (or object oriented programming) in animation. The important final subsection discusses functional animation.

3.2 Animating Jointed Figures.

This subsection introduces part-whole and coordinate hierarchies as important aspects of models of physical objects in general and animals in particular.

3.3 Smalltalk and C++.

Discussion, using the example of quaternions, of how object oriented programming was used in the implementations.

3.4 Representing Physical Objects: The part hierarchy.

This subsection discusses why Smalltalk and other object oriented languages need to be extended in order represent physical objects which are made up of parts.

3.5 Experience with Using the Part Hierarchy.

Results from the use of our extension to Smalltalk which used message forwarding to access the parts of an object.

3.6 Implications of the Part Hierarchy for Object Oriented Languages

3.7 Conclusion.

The use of object oriented programming in animating natural environments.

4 The Formulation of the Spatial and Temporal Priority Metrics.

4.1 Introduction: What is Detail?

Introduction to the mathematical theory of the properties of objects which the metrics will measure.

4.2 The Spatial Metric.

Derivation of spatial metric based on an analysis of spatial frequencies present in images.

4.3 The Temporal Metric.

The temporal metric is more elaborate than the spatial metric. Objects move in three dimensions and the changing images which result from this can be divided into a number of categories depending on the optic flow effects.

4.4 Data Types.

Discusses how the metric may be applied to data structures. It derives features needed in the protocol of the abstract data type which will interface which the various metrics.

4.5 Conclusion: Practical Priority Metrics for Animation.

Conclusion and summary of the chapter.

5 The First Experiment: Using the Static Metric on a Model with a Non-uniform Hierarchy.

5.1 Discrete detail and a continuous metric.

A hierarchy of coordinate systems is derived to describe stick figures, and their interaction with the metric.

5.2 Implementing the System of Priorities.

Simple moving figures with a mobile camera. Smalltalk version. Showing coordinate system hierarchies. Only the spatial metric applied here.

5.3 Results from Running the Experimental Implementation.

The costs and benefits of using the metric are analysed. The depth complexity of a scene is reduced. The method has low computational cost and imposes very

little extra work when a model is designed.

5.4 Conclusion.

6 The Second Experiment: The Spatial Metric Applied to Continuous Detail Levels.
Investigates the application of a static metric to those data structures with the pleasant property that they can be broken down to any desired level of detail.

6.1 Introduction: Digital Representation of Landscapes.
This subsection characterizes landscapes as chaotic single valued functions which may be evaluated over some grid.

6.2 Synthesizing Pictures of Landscapes.
Discussion of how landscapes can be modelled and rendered on a computer. Includes ways of approximating fractals, hidden surface removal and clipping. Some new results regarding the use of Quadcodes are presented.

6.3 Rendering Landscapes Efficiently on a Raster Display.
A recursive algorithm for using the spatial priority metric is presented with an object oriented implementation. The quadcode analysis from the previous subsection allows the recursive algorithm to communicate needed information efficiently from one recursion to another.

6.4 The Results of Using the Spatial Metric.
Analysis of costs and benefits. The major benefit, borne out by timings of the programs, was a great reduction in execution time. The storage requirements were also reduced. The cost is again minimal for the problem area chosen: that is, landscapes. But sampling artefacts have to be corrected if used for other purposes.

6.5 Conclusion.

7 The Third Experiment: Priority Based Execution of Motion.

All aspects of the temporal priority metric were investigated: adaptive updating, adaptive breakdown of 3-D motion into 2-D components and the trade-off between spatial and temporal detail.

7.1 Moving Figures and the Dynamic Metric.

Introduction to the application of the dynamic metric and the formulation of an experiment to test it.

7.2 Approximating Projected 3-D Motion with Optic Flow Effects.

The concept of frame-to-frame coherence is extended to include the notion of ‘‘Orders’’ of frame to frame coherence. The optic flow analysis from Chapter 4 is developed into an algorithm for modifying planar facets of 3-D objects. Algorithms for 2-D transformations of images of objects are developed.

7.3 A Simple Priority Based Animation System.

The object oriented implementation is outlined. Emphasis is placed on the need for dynamic binding in such systems.

7.4 Results of Using the Temporal Priority Metric for Animation.

A summary of the results achieved. For adaptive updating timings indicate much faster execution. The practicality of approximating 3-D with 2-D motion automatically under certain circumstances was demonstrated. The trade-offs between spatial and temporal detail was less conclusively beneficial.

7.5 Conclusion.

The most important and novel result from this experiment was the inversion of optic flow analysis to synthesize moving pictures.

8 Conclusion.

8.1 The Main Implications of the Space-Time Priority Metric.

What have we gained from all this?. The implications of the research both for

computer animation and object oriented programming are discussed.

8.2 Future Extensions.

Extending the results achieved by the experimental implementations. A detailed proposal for one aspect is found in Appendix D.

A Implementing a Part-Whole Hierarchy in Smalltalk.

Some practical details of the part-whole hierarchy introduced in Chapter 3.

B Timings of the Continuous Spatial Metric Experiment.

Tabulated raw timings of the second experiment.

C Timings of the Dynamic Metric Experiment.

Tabulated raw timings of the third experiment.

D Concurrent Object Oriented Animation.

This appendix contains a detailed proposal for using the part-whole hierarchy for parallel processing, and it discusses some of the expected benefits.

D.1 Concurrent Object Oriented Animation: A Research Proposal.

The object oriented approach is naturally conducive to a parallel processing implementation. This corresponds to the concurrency of events in nature. Part hierarchies provide a natural way of distributing a global name space. They also provide a hierarchy for controlling synchronization of actors.

D.2 Controlling Figure Animation.

Animation can either be externally controlled or else we can allow internal control by means of feedback between independent actors. In the latter case there are difficulties with ensuring termination of frame generation cycles.

D.3 A System for Concurrent Figure Animation without Feedback.

The part hierarchy and the system of ‘Appearances’ together provide a way of implementing the concurrent animation clock cycle in an extended version of

Smalltalk.

D.4 A System for Concurrent Figure Animation with Feedback.
The part hierarchy enables the building of a system of constraints and to detect when the processing for a single frame has terminated.

D.5 Conclusion.

Chapter II

The Appearance of Natural Scenes.

This chapter serves to answer the question: ‘‘What essential information about natural scenes would have to be simulated to provide convincing animated pictures?’’ This answer is arrived at through a survey and tutorial of parts of the vision and image analysis literature. The aim is to provide a useful synthesis of aspects of these fields which is relevant to computer graphics.

This chapter has a broad and qualitative approach, the more rigorous mathematical formulation of specific areas is left to Chapter 4.

§2.1 Scene Simulation and the Problem of Realism.

Computer graphics has been governed by a pragmatic approach to producing realistic pictures, with little attention being given to providing a theoretical basis for the techniques used. This pragmatism is found both in entertainments applications and in demanding applications like flight simulation where it is explicitly stated that the main goal is training effectiveness, not physical realism [Schachter, 1981]. The large collection of *ad hoc* techniques has become unmanageable. A better foundation has to be sought but it would be unfortunate if computer animation was forced into a restrictive framework.

The parallel quest for realism and improved theoretical standards has recently caused some controversy. On one side there are the often dogmatic adherents of physical correctness [e.g. Greenberg, 1988†], and on the other the defenders of ‘‘faking it’’ [e.g. Reeves, 1987 & 1988‡]. To some extent this is the difference between people interested in scientific visualization and those who use graphics for training (e.g. flight simulation) or entertainment. But there is a danger that physics will become the only theory underlying all animation, by default, because no alternative has been presented.

It is hoped that this thesis will help to dispel the notion that computer graphics improvements must be based on the laws of physics. This plea is obviously not against a scientific approach, but the science needed is much broader than simply physics. It must be a science

---

† ‘‘Just as VLSI advances were made by material scientists and biological advances were made by cracking the genetic code and molecular modeling, so must computer graphics improvements be based on the laws of physics.’’ [Greenberg, 1988].

‡ ‘‘You do what you can, and then fake it. That’s nothing to be ashamed of. I enjoy fooling you.’’ [Reeves, 1988].

which can incorporate ‘‘faking’’ and provide an explanation of how it works and which can also encompass principles of traditional animation like ‘‘exaggeration’’ [Lasseter, 1987].

We shall consider natural scenes: these are the obvious proving ground for realism. I propose that the basis for a sound theory for computer animation of natural scenes lies in an appreciation of what is visually important in the environment and integrating this with a theory of how we perceive artificial images.

Aspects of this theory are outlined in this chapter, but first some phenomenology:

The natural world as described by physics differs from the world of sensory experience. The purpose of computer graphics is not to simulate the former for its own sake but to stimulate the latter. This distinction is rather difficult to convey to those who do not accept it, and mundanely obvious to those who already accept it†. The world of physics is the world of objective facts about what ‘‘really’’ happens in the realms of energy and matter. The world of sensory experience is a world of interesting or boring pictures, convincing or unconvincing images. It depends as much (more?) on the perceiver than on the objects actually perceived.

Maintaining this distinction was crucial in formulating the approach taken in this research programme although it may not be as crucial in understanding the resulting theory. However, without this distinction there is a danger that the results may appear to have been pulled out of a hat and the ideas in the thesis may seem unrelated.

The contrast between physical representation and experiential modelling may be illustrated by means of a simple experiment [Varela, 1984; Maturana & Varela, 1988; but it dates back to Otto von Guericke, 1672]: Consider a white wall on which both a red light and a white light is being shone. If we put our hand in front of the white light we get a shadow surrounded by a pink background. Since only the red light is being shone on the shadowed area it is reddish. But if we obscure the red light instead we do not get a whitish shadow. It is aquamarine! No amount of ray-tracing ‘‘The Red, Green and Blue’’ components will ever give that colour.

---

† Philosophical Note: This view of computer graphics is compatible with the phenomenology expounded, for the benefit of computer scientists, in Winograd & Flores [1986]

Of course this argument does not mean that there is no real world, nor that studying tristimulus colour theory is worthless. It simply means that we have further options to consider in computer graphics and greater scope for creativity. The only test for pictures is the conviction they carry. Remarks concerning physical faithlessness are irrelevant. A corollary is that a new technique should not be accepted only because it models the underlying physical reality more accurately.

The argument concerning the appropriateness of physics, chemistry, biology, physiology and other ‘‘hard’’ sciences to the study of computer graphics is one of *levels of description*. The physical reality of the four elementary forces, of quantum theory, is clearly irrelevant to computer graphics. But so also is classical mechanics, radiation transfer functions and the neurophysiology of the eye. These are too low level. They will form components of an integrated theory. But such a theory also has to take into account the centuries of practical investigation into human visual experience conducted by artists.

2.1.1 Artificial, Natural, and Synthetic Perspective.

An example of the relevance of artistic experience to computer graphics is given by the old topic — the *sine qua non* of realism — perspective projection. The knowledge that the eye perceives only the solid angle subtended by an object, and that more distant objects subtend smaller angles, is ancient [Euclid’s Optics — third century B.C.]. This is commonly called *natural perspective*. Natural perspective is closely related to projection on the unit sphere surrounding the observer. Straight lines project onto great circles on the unit sphere.

Artificial perspective is the mathematically accurate perspective projection of three-dimensional scenes onto a two-dimensional plane. The first exponent was probably Filippo Brunelleschi in the early fifteenth century. On the plane, artificial perspective will reproduce the natural perspective solid angle correctly for only *one* particular viewing point. Straight lines project onto straight lines on the plane [see e.g. Carlbom & Paciorek, 1978].

The study of perspective much occupied the Renaissance painters; the term perspective covered a broad range of effects: perspective projection on a flat plane (artificial perspective), natural perspective (Euclid’s viewer centered pyramids; cf. Gibson’s optic array in §2.2.2), motion transformations (§2.2.2) and atmospheric effects (§2.2.3).

It is useful at this stage to dispel the notion that perspective projection as practised in computer graphics is absolutely rooted in physical reality. Pictures, including moving pictures,

are normally viewed without any regard for the one point at which the planar projection re-creates the visual solid angles of the scene. But we have all learned to unscramble the off-centre views to make them look realistic [Haber, 1983]. Artists also tend to avoid wide angles where artificial perspective diverges very much from natural perspective (unless the distortion is deliberately sought, as in anamorphic images).

Leonardo da Vinci [Ackerman, 1978] gave a good example of this distortion: If a row of circular columns parallel to the picture plane are projected then those further away will be *bigger* on the picture (see Figure 2.1). Various curvilinear perspectives have been proposed as alternatives but ultimately the advice has been to avoid situations yielding gross distortions. Synthetic perspective is the name given to various techniques to reduce the counter-intuitive distortions of artificial perspective.

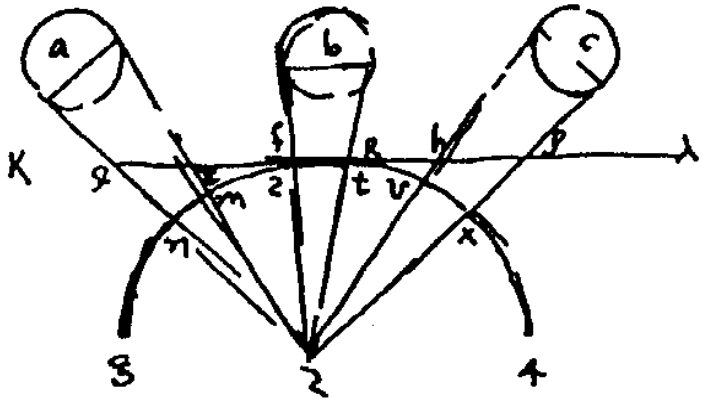


Figure 2.1. Demonstration of perspective distortion by Leonardo da Vinci. The image of a planar projection of the more distant columns is larger, while on a unit sphere they are smaller [Institut de France, manuscript A, folio 38^{r}, from Ackerman, 1978].

One final point can be made regarding realism: what seems very realistic today becomes dreadfully artificial tomorrow, what is realistic in one culture is stilted and artificial in another.

> When you look at a bed sideways, or in front, or from any other position whatever, does it alter its identity at all, or does it continue really the same, though it appears changed? ... Does painting study to imitate the real nature of real objects, or the apparent nature of appearances? ... Painting therefore is busy about a work, which is far removed from the truth.
> [Plato, *The Republic* 598, ~ 375 B.C. — quoted in Wright, 1983, p 35].

Plato would never have accepted that a theory of computer graphics which allowed perspective views could be compatible with physical laws — it would be faking things!

This discussion of realism has attempted to show how unsure our footing is in even that exemplar of realistic picture production: perspective projection. Actually, this problem of realism and perspective has been confusing artists (or rather art theorists) at least since the Renaissance. The particular question of spherical versus planar projection will recur when we define the spatial and temporal priority metrics.

2.1.2 Requirements for Synthetic Three Dimensional Moving Images.

The appearance of nature has long been investigated by artists and scientists. Centuries of experience in rendering it are available, and the standards are high. Nature has an intrinsically pied beauty. Gerard Manley Hopkins praises God for dappled things, for ‘‘Landscapes plotted and pieced — fold, fallow and plough.’’

But computer animation has an even more ambitious programme: rendering three-dimensional movement. The third dimension brings in a completely new set of changing visibility relations and deformations when movement occurs in the scene.

To make convincing three-dimensional animations there has to be a model of the world being animated. The background objects and all the mobile actors need to be represented in the machine, including a model of the mobile camera. That is, the system must ‘know’, perhaps implicitly, how images are seen. The visible scene then has to be rendered with a resolution in space and time which will satisfy the final user of the system.

Vision, our most important sense, is firmly utilitarian, irrelevant detail is ignored. Display devices are even more prosaic offerings in terms of their field of view, and spatial and temporal resolution. Our system can exploit these facts. The idea of the research presented in this dissertation is that one can arrive at an object space measure of how important the modelled objects really are to the viewer.

To derive such a metric we must allow for effects in two distinct realms:

First, we have to have a theory of how information would have been transmitted to an observer in nature. We then want to mimic these natural information transmission effects (§2.2 & §2.3).

Second, the viewer is actually sitting in front of some raster display device or perhaps a movie screen. So, we also need a theory of how information is *in fact* transmitted to the viewer who is actually watching the synthetic images. We wish to allow for the way these artifacts affect information transmission (§2.4 & §2.5).

These two theoretical descriptions have to be combined to yield a theory of how to simulate natural viewing conditions on a computer display unit (a Surface-Atmosphere-Camera-Raster-Observer Viewing theory!). Even this simple naming of the theory contains assumptions, for example, it is not clear that modelling a synthetic camera is at all necessary: we need the information about the imagined scene converted to a form which is intelligible on a display, we do not need to model a camera *per se* [cf. Haber, 1983]. Although for the moment we may be content with doing that.

These descriptions can be thought of in terms of changes in knowledge representation levels. It starts at a high level of knowledge representation in the machine: a complete simulation of a natural environment. It proceeds downwards in steps to a very low level: two-dimensional moving light patterns on a display. The human visual perceptive apparatus then extracts the information from this display to re-create the high level representation. Thus computer animation means solving the inverse problem to that addressed by computer vision [Eklundh & Kjelldahl, 1985].

A Theory for Scene Simulation.

The spatial and temporal metrics which were introduced in the previous chapter (§1.2), have the aim of producing a satisfactory image as efficiently as possible. The metric measures information required from the earlier stages of processing for achieving this end.

Section 2 of this chapter discusses the appearance of natural scenes, and shows in which ways the natural world of perception differs from the natural world described by physics. Subsequently we discuss image formation from the perspective of Fourier analysis (§2.3) and then the way information is transmitted to the viewer (§2.4). We conclude with the implications of this survey for our thesis (§2.5), and a brief summary (§ 2.6).

These metrics and their associated theoretical background should make some contribution to providing a theoretical foundation for realistic computer animation; a foundation which can embrace and explain a wide range of techniques. Such a basis should provide a fertile

ground for deriving further applications and new techniques.

§2.2 Describing Natural Scenes

This section is firmly based on the work of Gibson on ecological optics [Gibson, 1979]. In computer science, knowledge of Gibson's work is mainly limited to computer vision researchers. Ironically Gibson gives a very detailed description of the complex stimuli available in the environment, while avoiding any theory of the processing involved in perception. That fact seems to make his work more applicable in computer graphics. The relevant features of his research and of others will now be reviewed and examined. Recently a good discussion of the relevance of vision research to computer graphics has appeared [van de Grind, 1987].

As a starting point Gibson insists on the distinction between the world of physics, and the environment as perceived by animals. He also maintains that the observers and their environment are complementary. These topics have already been mentioned in the previous section. The major departure from other approaches to investigating vision was Gibson's insistence that visual perception operates properly only in a situation where the observer is free to move about, use both eyes, and observe rich changing scenes. Clearly a visual display unit is limited in this respect, but various computer generated environments where the viewer is immersed in a scene in which he is free to move about, could create such a situation artificially [e.g. Brooks, 1987, Fisher et al., 1987].

2.2.1 Natural Surfaces: Appearance and Simulation.

The most important feature of the natural environment (and the one currently giving such rewarding problems to computer graphics) is that the components of nature exist in many levels of detail. These levels of detail are simultaneously perceived as *nested* within one another: Grains of sand on beaches in bays along the coast; stones amongst rocks below cliffs on the mountains in the distance; everything covered with vegetation; fine fuzz on leaves on trees in forests. At every scale there are forms within forms.

Thus objects consist of components, and these components have further, smaller, components. These components form a hierarchy in some important cases [see §4.4.1], but in general there is an overlapping network of relationships.

There is no fixed scale with which to measure things, scale adapts to the situation [‘‘... no atomic units of the world considered as an environment.’’ Gibson, 1979, p9]. Equally, there is no absolute flow of time. Instead of time there is change and sequences of events. No events means no time and no process. It can be seen therefore that an aspect of the proposed static metric is to provide the appropriate distance scale to measure the environment in the given situation, while the dynamic metric measures the flow of time appropriate to a given circumstance.

The environment is characterized by *persistence* of solid *substances* and their layout. There are semi-solids which change shape and liquids which are contained by solids and perturbed by waves. The air is not a substance in our view but a *medium* permitting vision and locomotion. Solids are perceived by the layout of their textured *surfaces*.

Gibson argues that in ecological optics it is more important to distinguish between *animate and inanimate* objects than between living and non-living, as biologists do. This is because observers generally treat plants as part of the background along with the rocks and the soil.

The texture, or graininess, of the surfaces of particular substances remains relatively constant and is one of their most distinctive characteristics [Haber, 1983]. A textured ground stretching away from the observer gives information both on the slope and the distance of the surface.

Mandelbrot [1982b] has advocated the use of fractals to model many natural phenomena. The name fractal is meant to invoke the idea of broken, irregular objects. Fractals have (statistically) self similar detail at all scales. They are very compact procedural models from which convincing images of rivers, coastlines and mountains can be generated. Pentland [1984] presents some evidence to indicate that fractals capture what naive observers mean by surface roughness.

In contrast, it can be argued that we can recognize animals quite well without having to reproduce their surfaces [Marr & Nishihara, 1978]. The success of stick figures and pipe cleaner animals bear this out. The essential feature of such figures is a hierarchy of coordinate systems arranged along the natural axes of the parts of the figures.

Stick figures serve for many natural shapes whose form was achieved by growth. Other natural shapes, perhaps because they were produced by random weathering, can be described by fractals. Plants belong to both camps: they are sticks with random texture.

Both Mandelbrot (‘‘to see is to believe’’ [1982b, p. 256]) and Marr & Nishihara (‘‘as we see ... animal shapes are portrayed quite effectively’’ [1978, p 271]) appeal to the convincing images presented to justify their models: this is a very good basis for a representation to use in computer graphics.

2.2.2 The Effects of Motion on the Appearance of Objects.

The natural environment contains many surfaces and a great many textures. All of them reflecting light in their varied ways. Light is not only transmitted by air, but rebounds between all the surfaces to reach an equilibrium. This ambient illumination is structured with information about all the nested surfaces of the environment.

Gazing in any direction an observer is apparently at the convergence point of a dense structure of intersecting visual pyramids. There is a pyramid for every discernible feature in the scene — it is the solid angle which light from the outline of the object subtends at the eye of the observer. Together the visual pyramids from all objects in the scene form the *optic array* (Figure 2.2).

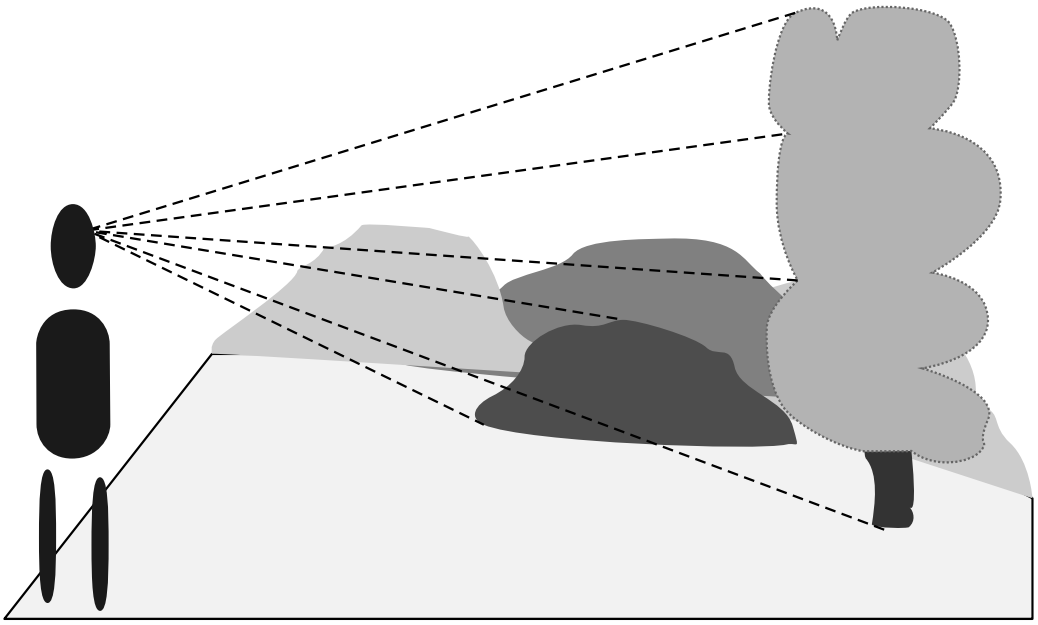


Figure 2.2. The ambient optic array. The optic array is formed by the nested solid angles which radiate from the viewpoint.

Some of these interlocking pyramids belong to animals and have their own independent motion, but superimposed on these independent proper movements are the global effects of the observer’s own translations and rotations. These movements of the optic array constitute

the *optic flow field*. The optic flow field is the foundation of depth vision [Lee, 1980], especially in the absence of binocular vision. The notion of optic flow originated with Gibson and has been in wide use in computer vision research [e.g., Buxton, 1984; Buxton & Buxton, 1983; Koenderink & van Doorn, 1975, Koenderink, 1986; Lee, 1980; Waxman & Ullman, 1985]. In the computer graphics literature there are some reviews: [Neumann, 1984 — brief] and [Van de Grind, 1987].

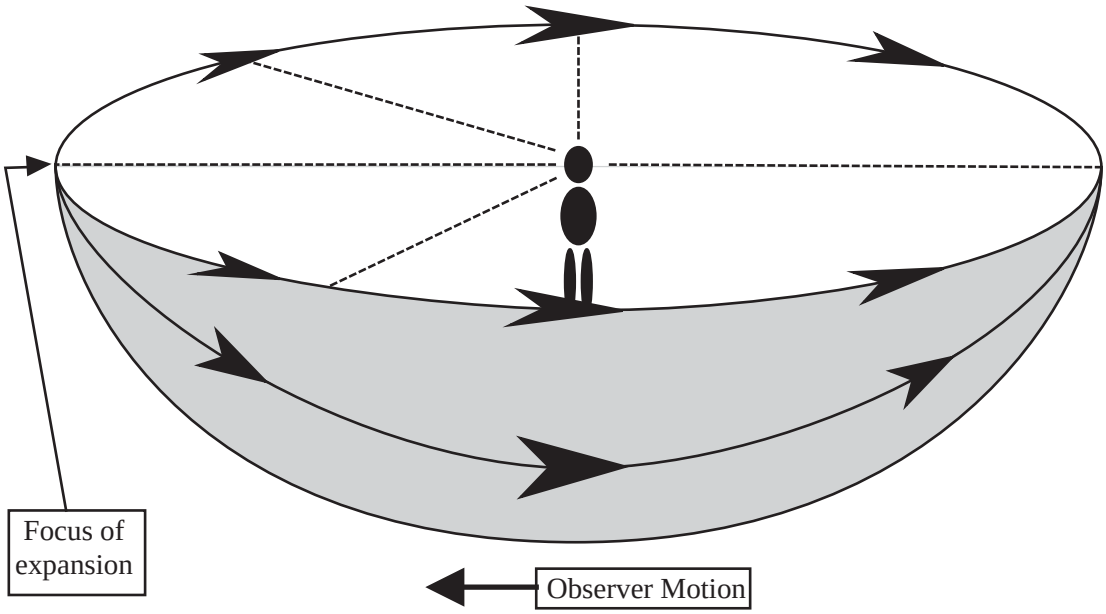


Figure 2.3. Optic flow. The flow of optic array resulting from observer locomotion. The vectors represent the angular velocities of texture elements in the scene.

We shall be returning to the optic flow field in Chapter 4 when formulating the dynamic metric. We shall then give a mathematical formulation which will lead to an analysis of the hierarchies of optic flow effects. For the present we shall give a qualitative summary of the features of the optic flow field [Koenderink, 1986] (see Figure 2.3).

Global Features of the Flow.

- *Rotations* around an axis through the vantage point (eye movements to a very good approximation) yield no information about spatial layout, or depth.
- *Translations* of the vantage point always yield the same pattern of field lines. The shape, but not the value, of the flow field is independent of spatial layout. In other words: when moving forward the general trend will be for objects to move

backward, radiating out from a focus of expansion and this fact is independent of where the objects are in the environment.

- Optic flow consists of *piece-wise smooth regions* within which the flow is smoothly varying, separated by discontinuities.

Local Features of the Flow.

- The *Average flow* velocity at a point in the field.
- The *Motion parallax* field is the structure of the local variation of velocity in the immediate neighbourhood of the point. This important topic is amenable to vector field theory analysis and this is discussed in §4.3.2.

The analysis of Chapter 4 is based on the observation that any surface patch can be described as a Taylor series expansion in terms of distance in a given direction. The various Taylor series terms (distance, orientation, curvature, etc.) can each be used to specify local optic flow (or motion parallax) effects — 0, 1st, 2nd, ... order optic flow effects. Moreover, the 1st order terms lead to an affine transformation of the image of the patch which in turn can be decomposed into basic transformations: translation, scaling, rotation and shear.

In Chapter 7 these various orders of optic flow effects are implemented and reformulated as various orders of frame-to-frame coherence.

2.2.3 Atmospheric Effects.

A detector of light pointing in a certain direction through the air, will not only pick up light reflected off objects in its path, but also photons from objects near the target (adjacency effect). This results in the apparent field of view being enlarged and high spatial frequencies being lost. In addition there is also light scattered directly from the sun (‘‘sun’’, since we are considering natural scenes) and from ground reflectance along the path. This causes loss of contrast, for example, black surfaces seem lighter [Kaufman, 1984; Gorraiz & Horvath, 1983].

In deriving an expression for light transmission through the atmosphere various simplifying assumptions can be made. These include:

(a) every volume element of the atmosphere is illuminated by the same amount of light.

(b) both extinction coefficient and scattering function are constant along the path of sight.

The radiance, L, reaching the eye from a specific direction can be written as follows:

$$L = L_S + L_A + L_O + L_P \tag{2.1}$$

where:

L_S *Signal*: Radiance reflected by the surface and directly transmitted: this is the attenuated signal providing information about the object. The light is reduced by the factor $\frac{exp(-\sigma_\varepsilon \cdot r)}{r^2}$ by atmospheric extinction and the inverse square law (r is distance and σ_ε is the extinction coefficient of the aerosol).

L_A *Adjacency Effect*: Reflected by surface and scattered by the atmosphere. Diffuses radiation between fields and reduces the apparent resolution of the sensor.

L_O *Scattered Sunlight*: Radiance of direct sun beam scattered by the air. Independent of the surface reflectance and causes loss of contrast.

L_P *Path Radiance*: Reflected by surfaces along the path of the light and then scattered towards the sensor. Depends strongly on the albedo of surfaces between the sensor and the target and can often be neglected. It has similar effects to L_O as far as the sensor is concerned.

Summary.

- Environment is *what we perceive* (*not* the world of physics):
- Distinguish animate (animals) from inanimate (including plants).
- Components and events of nature fall into nested, overlapping, levels of detail. This gives rise to the *ambient optic array = nested visual solid angles*.
- Observer motion transforms the ambient optic array and creates the *optic flow* field.
- The minimal natural environment can be modelled by a richly textured field and animals.

§2.3 Image ~ Object Relationships and the Optical Transfer Function.

In this section some concepts from Fourier analysis, which are needed later to discuss how images are actually perceived (§2.4), are reviewed. It can be skipped if the ideas are familiar.

2.3.1 Analysing Image Formation.

Changing images are functions of two space variables and of time, they can be written as $f(x, y, t)$, where f is the time (t) varying illuminance at the image plane; x and y being the horizontal and vertical axis respectively. The image can be analysed as the sum, or integral, of a large number of space and time frequencies, which constitute its spectrum. The Fourier transform operator performs this spectral analysis of f, producing $F(\xi, \eta, \nu)$. Where ν represents temporal frequency and ξ and η represent the horizontal and vertical spatial frequencies. In optics F is invariably non-zero only over a finite part of its domain; that is, f is always a band-limited function. The bandwidth of f is closely related to the information content of the changing scene.

Optical information is transferred from an object through the air and the lens system of the eye, to the retina. From there sensors transmit the information to the cortex via various visual ‘‘channels’’. At each stage transmission is governed by the characteristics of the medium; at each stage the information is filtered.

Duffieux [1983, p86] discusses two ways to model transmission:

1. *Transmission controlled from the outside*:- by means of transformation invariants between the object and image spaces, or an initial to a final state, or input excitation to output response of a filter. Convolution theory yields a model which can be applied to the domain of image formation where the mode of transmission not determined.

2. *Transmission controlled from the inside*:- a mode of transmission is defined that may be progressively integrated from the object to the image. Equations of propagation are: ray-tracing equations, Fermat’s principle, Huygens’ principle, wave equations, etc.

In computational terms we may loosely call the first a black-box, declarative approach, and the second the explicitly simulated, procedural approach. We shall use Fourier or convolution method for reasoning about the process of image formation, but we shall generally use the other methods when producing pictures.

2.3.2 Linear Systems and Convolutions. [Bracewell, 1978; Duffieux 1983; Yu, 1976; Pearson, 1975]

In physical imaging problems linearity and spatial invariance is often an inaccurate but useful approximation. There are also doubts about the degree to which neural sampling and transmission can correctly be analysed with a linear theory†. However no general techniques exist for analysing nonlinear problems.

Necessary and sufficient conditions for linearity correspond to the *principle of superposition* (''the response of a linear system due to several inputs acting simultaneously is equal to the sum of the responses of each input acting alone''). Another consequence is that a scale factor of the input is preserved in the output (this is sometimes referred to as ''homogeneity'', but in this context that can also mean space-invariance, which is not necessarily connected with linearity).

A system with constant parameters will also possess the important property of being *time-invariant* and *space-invariant.* Under these conditions the response of a linear system to harmonic input is itself harmonic at the same frequency. The response is then related to the stimulus as a *convolution* (alias *Faltung* or composition product). In optics an instrument which can be described in this way is said to have an isoplanatic spread function. This spread function gives the way a point of light energy is dissipated, it is also called the impulse response.

Let $f(x,\ y,\ t)$ be the image which a ''perfect'' optical system would transmit, f can also be regarded as the input to the system. Let $d(x,\ y,\ t)$ be the observed effective image. If we write $g(x',\ y',\ t')$ for the point spread function, then the convolution is:

$$d(x,\ y,\ t) = f(x,\ y,\ t) * g(x,\ y,\ t) \tag{2.2a}$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',\ y',\ t') \cdot g(x-x',\ y-y',\ t-t')\, dx'\, dy'\, dt'$$

---

† Discussed in [Westheimer, 1972]. Linear example: [Campbell & Robson 1968]; non-linear example: [Fiorentini, 1972].

This can conveniently be written in vector notation as:

$$f(\mathbf{r}) * g(\mathbf{r}) = \int_{-\infty}^{\infty} f(\mathbf{r}')\, g(\mathbf{r}-\mathbf{r}')\, d\mathbf{r}' \tag{2.2b}$$

where the integral is over all dimensions.

The convolution can be interpreted as a superposition of the spreads of each point of the perfect (or *stigmatic*) image. The convolution has a smoothing effect on sharp changes in the stigmatic image. The convolution product is commutative and associative and also distributive over addition — hence the asterisk (∗) notation.

We define a unit impulse as:

$$\delta(x) = 0 \quad x \neq 0 \tag{2.3a}$$

$$\int_{-\infty}^{\infty} \delta(x) = 1$$

In three dimensions:

$$\begin{aligned} \delta(x, y, z) &= \delta(x)\, \delta(y)\, \delta(z) \\ &= \delta(x, y)\, \delta(z) \end{aligned} \tag{2.3b}$$

while for circular symmetry in space:

$$\delta(x, y, t) = \frac{\delta(r)\delta(t)}{\pi r} \tag{2.3c}$$

where

$$r = \sqrt{x^2 + y^2}$$

For the sake of brevity we express our ideas in one dimension for the moment.

The impulse response has already been introduced intuitively above, it has the property that:

$$\delta(x) * g(x) = g(x) \tag{2.4}$$

An invariant linear filter is completely specified by its impulse response, g, since the response to any input is a convolution with g.

If we regard information as being transmitted in stages, as a series of images, from the object $f_0(x)$ to a final image $f_n(x)$, then for each intermediate stage $f_1, f_2, \cdots f_{n-1}$ there is a corresponding convolution, and we can write:

$$f_0 * g_1 * g_2 * \cdots * g_n = f_n \tag{2.5}$$

Each successive convolution is a frequency filter.

2.3.3 Fourier Analysis of Band-limited Functions.

Using the same vector notation as that used above we may write the Fourier transform pair as:

$$F(\mathbf{k}) = \int_{-\infty}^{\infty} f(\mathbf{r}) \exp(-i 2\pi \mathbf{r} \cdot \mathbf{k}) d\mathbf{r} \tag{2.6a}$$

$$f(\mathbf{r}) = \int_{-\infty}^{\infty} F(\mathbf{k}) \exp(i 2\pi \mathbf{k} \cdot \mathbf{r}) d\mathbf{k} \tag{2.6b}$$

A band-limited function is one whose Fourier transform is non-zero only over a finite range. In optics we are only concerned with band-limited functions. We are, of course, ultimately limited in the spatial detail we can see by the wavelength of visible electromagnetic radiation. However, long before this limit is reached, the various transmission systems between our mind and the outside world would limit spatial and temporal frequencies far more drastically.

In dealing with band-limited functions two special functions which form a transform pair are very useful: the rectangle function of unit height and base $\Pi(x)$ and the filtering or interpolation function $\operatorname{sinc} x$.

These are defined as follows:

$$\Pi(u) = \begin{cases} 0 & |u| > \frac{1}{2} \\ \frac{1}{2} & |u| = \frac{1}{2} \\ 1 & |u| < \frac{1}{2} \end{cases} \tag{2.7}$$

$$\text{sinc} \;=\; \frac{\sin \pi x}{\pi x} \tag{2.8}$$

with:

$$\text{sinc } 0 \;=\; 1$$

$$\text{sinc } n \;=\; 0 \qquad n \text{ non-zero integer}$$

$$\int_{-\infty}^{\infty} \text{sinc } x \;\; \mathrm{d}x \;=\; 1.$$

In three dimensions we have:

$$\Pi(\xi, \eta, \nu) \;=\; \Pi(\xi) \cdot \Pi(\eta) \cdot \Pi(\nu) \tag{2.9}$$

$$\;=\; \Pi(\xi, \eta) \cdot \Pi(\nu)$$

and

$$\text{sinc } (x, y, t) \;=\; \text{sinc } x \;\; \text{sinc } y \;\; \text{sinc } t \tag{2.10}$$

$$\;=\; \textit{sinc}\,(x, y) \cdot \;\text{sinc } t$$

If we have $\sqrt{x^2 + y^2} \;=\; r$; that is, circular symmetry, then we have the following Fourier transform pair:

$$f(r,t) \;=\; \Pi(r)\Pi(t) \qquad \text{(disc)} \tag{2.11a}$$

$$F(k, \nu) \;=\; \frac{J_1(\pi k)}{\pi k} \;\text{sinc } \nu \tag{2.11b}$$

where J_1 is a first order Bessel function of the first kind.

The properties of sinc and Π given above apply to all *separable* functions; that is, (in two dimensions) if $f(x, y) = f_1(x) \cdot f_2(y)$ then the Fourier transform is $F(\upsilon, \nu) = F_1(\upsilon) \cdot F_2(\nu)$.

The convolution of sinc x with another function removes all components above a cut off frequency but leaves all below unchanged; that is, it performs ideal low-pass filtering.

If f is a band-limited function with Fourier transform F, we can write Φ for the function Π which has been shifted and stretched so that its range (the values of the domain for which it

is unity) coincides with the range of F. The transform of Φ is ϕ. We then have the following identities:

$$F\Phi \equiv F \tag{2.12a}$$

and

$$f * \phi \equiv f \tag{2.12b}$$

2.3.3.1 Multiple Convolutions.

Consider the succession of images from Equation 2.5:

$$f_0 * g_1 * g_2 * \cdots * g_{n-1} = f_n \tag{2.13a}$$

$$F_0 G_1 G_2 \cdots G_{n-1} = F_n \tag{2.13b}$$

$$\phi_0 * \phi_1 * \phi_2 * \cdots * \phi_{n-1} = \phi_n \tag{2.13c}$$

$$\Phi_0 \Phi_1 \Phi_2 \cdots \Phi_{n-1} = \Phi_n \tag{2.13d}$$

here f_0 and f_n are the initial and final images and the g_1 ... g_{n-1} are the intermediate point spread functions. The Fourier transforms (2.13b) are simple products. If we neglect time then we can call the Fourier transforms the spatial spectra. The Φ_0 ... Φ_n are called the spectral ranges, while the ϕ_0 ... ϕ_n are the spread functions or internal diffraction functions. The final image has a spectrum which is the intersection of the ranges of the intermediate stages.

The G_1 ... G_{n-1} are called the transfer functions of the filters. The impulse response completely specifies the filter and so does its Fourier transform, the transfer function. Measuring the transfer function for optical devices directly from the point spread function is usually difficult. The usual procedure is therefore measure the relative contrast for sinusoidal test patterns of all frequencies and orientations. In this way the attenuation as a function of frequency can be found and thus the transfer function determined.

2.3.3.2 The Optical Transfer Function. [Hall, 1979]

The response of an imaging system to incident radiation can be described in terms of the optical transfer function (OTF). As seen in the previous subsection the OTF is simply the Fourier transform, G, of the impulse response or point spread function, g.

The magnitude or modulus of G is called the modulation transfer function (MTF):

$$M = |G| \tag{2.14}$$

2.3.4 Sampling.

In order to represent a image on a video screen it has to be scanned, or sampled. For a raster display this is a two-dimensional process. The sampling symbol (called *shah* or comb) III(x) is defined as:

$$\mathrm{III}(x) = \sum_{n=-\infty}^{\infty} \delta(x-n) \tag{2.15}$$

The symbol III(x) is its own transform.

Sampling is represented by multiplying some picture f by III. If sampling takes place at intervals τ this is represented by $f(r)\,\mathrm{III}(\frac{r}{\tau})$. The transform is $\tau\,\mathrm{III}(\tau k) * F(k)$. $\tau\,\mathrm{III}(\tau k)$ is a row of impulses at spacing τ^{-1}. Thus the spectrum F(k) is replicated in the frequency domain at τ^{-1} intervals.

The sampling process is completed by extracting a single copy of the replicated spectrum with the $\Pi(k)$ function. This will be possible provided the replicated spectra were not mixed up through overlap, that is, if the interval of replication τ^{-1} was large enough to prevent the spectra from overlapping. If the range of $F(k)$ is from $-k_c$ to k_c, where k_c is the highest frequency present in the picture then $\tau^{-1} > 2k_c$ is the criterion for non-overlap and hence reproducibility of f from the samples $f(r)\,\mathrm{III}(\frac{r}{\tau})$. For critical sampling we have $\tau^{-1} = 2k_c$. Thus to reproduce a band-limited function we have to sample at intervals not exceeding ½ its shortest wavelength.

If the sampling is too coarse for the range of frequencies present in the image then some of the duplicated higher frequencies will overlap with the lower frequencies. The high frequencies will be indistinguishable from the lower frequencies. This phenomenon is known as aliasing: the high frequencies appear under another alias. Preventing this from happening, or minimizing its effects, is known as anti-aliasing.

2.3.5 The Fourier Transform of an Image in Motion.

In subsection 2.3.1 we wrote the Fourier transform of a changing image, $f(x, y, t)$, as $F(\xi, \eta, \nu)$, where ν represented the temporal frequency and ξ and η represented spatial frequencies. Motion induces a shear in the temporal frequency dimension. If the velocity of the image is (u, v) then the new temporal frequency ν' is related to ν by

$$\nu' = \nu + u\xi + v\eta \tag{2.16}$$

The spectrum of a stationary image lies in the ξ, η plane, when the image moves the spectrum lies in an oblique plane through the origin [Adelson & Bergen, 1985; Watson & Ahumada, 1985].

Summary

- We use Fourier analysis to deduce the properties of moving images.
- A sequence of convolutions describe image formation to a reasonable degree of accuracy. Imaging is then a sequence of filtering operations.
- The components of an optical system are characterized by their impulse response or point spread function.
- In optics we deal with band-limited functions. When such functions are convolved together the resulting range of frequencies is limited to that of smallest range taking part in the convolution.
- The Optical Transfer Function is the name given to this concatenation of convolutions.
- Sampling effectively replicates the spectrum of a sampled function. The interval between these replicated copies of the sample's spectrum is inversely proportional to the sampling interval. Band-limited functions have to be sampled with a frequency at least twice their highest spectral frequency.
- When an image moves its spectrum is sheared in the temporal frequency dimension.

§2.4 The 'Atmosphere'-VDU-Eye transfer function.

We wish to portray changing scenes and we would like to adapt the information displayed to meet the minimum requirements of convincing the human eye. Animators already use psychophysical principles implicitly: the essential effect upon which animation depends is the way the eye integrates changes in the visual field over time. A consequence of this is that a succession of static samples from a changing optic array can be indistinguishable from the original continuously changing array. This effect makes cinema possible — different frames are presented twenty-four times a second, flashing on and off at about three times that rate.

Colour video displays already exploit the psychophysical model expressed in the tri-stimulus theory of colour to reduce the variation in wavelength distribution greatly (i.e. down to the responses of three coloured phosphors).

In this section we discuss various broad principles of human vision. This is relevant in two distinct areas of research:

- The synthetic camera which roams our artificial model of the natural environment is not necessarily meant to mimic a camera. It is there to get the information from the scene that an observer would get.
- When we present the information to the real observer, the viewer of the animation, it is as if everything is being seen for a second time. This time the act of observation is real but the display is artificial.

In both these cases features of human vision are important. The first step of perception is light entering the eye (§2.4.1). This light is sampled by the retina (§2.4.2), where certain fundamental sampling trade-offs hold. The next stage of perception occurs in the cortex and here we can observe pathways or channels which specialize in the different dimensions of the stimulus (§2.4.3).

Having traced the information, we can make certain observations about spatial (§2.4.4), brightness (§2.4.5) and motion (§2.4.6) vision. Finally we emphasize the fact that display screens form an essential link between our artificial eye and the real observer (§2.4.7).

2.4.1 The Eye as a Receiver of Optic Information.

Light, as the physical conveyor of optical information about the environment, can vary only in terms of luminance, wavelength distribution, and spatio-temporal pattern.

Mainly because we want to have a tractable problem, but also because of hardware limitations imposed on this research, our main concern will be with conveying information by means of spatio-temporal variation of shaded, (more or less) monochrome, images. These images will subtend angles of about twenty or thirty degrees at the viewer.

All rays reflected from the environment cross over at the eye's nodal point and are mapped onto retina with a central (polar) projection. This mapping results in a perspective transformation of figures, as in a camera. But the eye is nothing like a camera when it comes to capturing motion. The motion camera has a shutter which serves to neutralize movement and produce a series of static images. The eye has no shutter, it is constructed for continuous recording of optical change over time. The spatio-temporal flow in the incoming array of light, the *optical flow,* is in fact needed to get a neural response to the information encoded in the light energy. Essentially the eye analyses only retinal flows, it deals with time-continuous perspective transformations [Johansson, 1978].

The purpose of computer animation therefore is to induce a perception of optic flow which contains the necessary information, not (necessarily) to produce a sequence of complete static images.

2.4.2 Sampling Trade-Offs in the Eye.

As pointed out many times already it is the human eye which is to be simulated. The synthetic camera can then incorporate a model of the sampling processes occurring in the human eye.

D.M. MacKay [1981] discusses space vs. frequency domain representations and touches on sampling theory as it applies to a simple cell in the visual cortex. In terms of Gabor's [1946] theory of communication, if we want to represent a signal by a sequence of samples, we have to accept a compromise between uncertainty in time and in frequency. The uncertainty relation is $\Delta f \cdot \Delta t \geq \frac{1}{2}$, where Δf and Δt are imprecisions in frequency and in time respectively. For the product $\Delta f \cdot \Delta t$ to be a minimum, the elementary samples should not have sharp cut-offs in time or frequency. For analogous reasons functions of space are most efficiently sampled with elementary spatial functions with Gaussian profiles. For any linear

system then, a given spatial sampling spread results in a corresponding spatial frequency preference. The narrower the spatial profile (Δx), the poorer the frequency selectivity.

The receptive fields of simple cells are near optimal in sampling optical images in both frequency and spatial domains. The spatial-frequency bandwidth is relatively wide (½ wavelength or roughly an octave) suggesting that the information-theoretic compromise is weighted in favour of spatial resolution.

The theory presented above is in terms of one dimension. Vision operates on three dimensions: the two space dimensions and time. Daugman, [1985] has discussed Gabor's theory for the two space dimensions. Basically the uncertainty relation given above applies to both the x and y directions separately. The joint lower limit to resolution is thus ¼.

A salient feature of the response of a simple cell of the visual cortex is a preference for edges or bars at specific orientations [Hubel & Wiesel, 1962]. A preferred orientation and a preferred spatial frequency are the polar spectral variables parameterizing the 2D frequency domain of 2D linear spatial filters, which could equally well be expressed in the cartesian coordinates of width and length in the frequency domain.

Thus the same uncertainty relation limits the attainable joint resolution in space, spatial-frequency and orientation. The inescapable trade-offs can therefore be achieved in different ways and there seems to be a division of labour between cells with different cells doing different jobs by using a particular set of trade-offs.

2.4.3 Human Visual Perception.

A fundamental insight of sensory physiology is that there are many parallel pathways within a sensory system, each specialized to carry information about a different stimulus element [Westheimer, 1972; Bradick et al., 1978]. Information from different parts of the visual field about various different properties of that field are carried to different neurons for further processing. Each neuron can be said to have a receptive field, which is a certain part of the total stimulus space to which the visual system can respond. The dimensions of the field are not just the two position dimensions but also dimensions such as orientation or colour.

The complete range of a stimulus dimension can be regarded as subdivided into sub-ranges, with combinations of subranges forming receptive fields. An independent structural element of the visual system, dealing with its particular subrange of the stimulus dimension, is called

a channel. A classic example of channels in visual processing is the idea (first enunciated by T. Young in 1802) that different structures in the visual system respond to different wavelength ranges of light.

Image/Object relationships in optical systems can be specified in terms of Fourier theory (§2.3). If an object is a sinusoidal grating, i.e., a grating across which luminance variations are sinusoidally modulated, it forms an image which is also a sinusoidal grating, albeit with reduced contrast or modulation. The eye can be characterized by a graph showing the demodulation of sine wave gratings as a function of their spatial frequency, the so-called modulation transfer function (MTF — see §2.3.3.2). This function is the Fourier transform of the line-spread function and contains the same information.

To the extent that the visual system can be treated as linear, the MTF provides a complete description. This is generally the case where input variations are small (i.e., no saturation, accommodation, etc. and homogeneous response to spatial variation). In a sinusoidal grating modulation may be defined by the ratio, $m = \frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}}$.

In a prototypical experiment to determine the spatial MTF, Schade [1956] determined the threshold modulation ratio for a number of spatial frequencies. He found that the threshold modulation sensitivity falls off with increasing spatial frequency as can be expected for any optical system. However experiments indicate a response attenuation at low spatial frequencies, especially at higher luminance levels. The eye's optics and the rest of the visual system act as a band pass filter tuned to a center frequency of about 5 cycles/degree.

Campbell & Robson [1968] found that the detectability of a square-wave grating could be predicted in terms of the detectability of its Fourier components. Thus, at least at threshold, linear analysis could be applied to the visual system. Their findings also indicated however that each narrow band of spatial frequencies is dealt with by an independent channel. The overall MTF being the envelope of the sensitivities of the separate channels. The multi-channel model implies that the separate channels signal the components of an approximate Fourier analysis of the input stimulus done by the visual system.

The major conclusion from this section is that the higher visual functions can be analysed, at least to begin with, by means of the same Fourier analysis developed in §2.3. Thus for our purposes an analysis in terms of spatial and temporal frequencies will have a wider application than just the optical stages of the perceptual process.

2.4.4 Spatial Discrimination.

As far as spatial vision is concerned, the *optical* apparatus of the eye is a linear filtering system [Westheimer, 1972]. It transforms each luminous point in object space into light distributed on the retina according to a point-spread function (§2.3.2). Conversely, the distance weighted intensity spreads of all objects sum at each retinal point, that is, each retinal sensor picks up a convolution of the light contributed by all (in practice, nearby) objects.

The further, neural, processing of the visual system has been discussed in the preceding section. It is much more complicated.

Characteristics of human spatial vision:

1 Visual acuity of the retina drops off rapidly with eccentricity of the retinal image from the fovea.

2 Beyond a certain luminance level visual acuity is independent of illumination.

3 Horizontal and vertical gratings are resolved better than oblique ones.

4 The human visual system is capable of judging relative position with remarkable accuracy. Thresholds for these tasks are often 3-5 arc sec. These low thresholds are 5-10 times finer than either the cut off spatial frequency, as determined by the spatial filtering properties of the pupil, or the inter-cone (retinal sensor) spacing.

2.4.5 Brightness and Colour Discrimination.

Seeing depends on the ability to distinguish differences in light intensity and colour. In developing the spatial and temporal metrics (the main thrust of this study) we shall be ignoring questions which relate specifically to colour discrimination. The loss of colour with distance because of scattering (things turn blue with distance) could be incorporated into the spatial metric with ease. However practical tests would add to the complexity of the initial experiments without adding much to the conviction of the results. Thus colour discrimination was not considered essential to the initial development.

Questions of contrast thresholds, on the other hand, are directly relevant to the development of the spatial metric. Spatial resolution is related to contrast ratio. The retinal sensors respond to the volume under the surface of point-spread functions. The detection of a small object with a wide point-spread function is equivalent to a small contrast detection.

2.4.6 Motion Vision.

Real movement refers to the experience of motion when an object is continuously displaced across the visual field [Anstis, 1978]. Apparent motion is said to occur whenever the displacement of an object is discontinuous or intermittent and motion is still perceived. Since any individual frame in a motion picture is actually stationary this is said to convey apparent motion.

Movement perception is not a failure to resolve space and time, but rather an active integration of the two. Movement is seen if two spots are alternately exposed with a separation in space which can easily be resolved. Even if the separation in time is too short to allow perception of sequence, motion can still be seen.

While real and apparent movement are equivalent in some sense in the visual system, the question arises whether the same mechanism is used to perceive both. The answer [Bradick, 1974] seems to be that apparent movement perception is mediated by two processes: one, a ‘‘short-range’’ process, also handles real movement, while the other, a ‘‘long-range’’ process, does not.

The short range process operates with video displays, and is important for our study. This process can be shown to precede shape recognition. For example, random ‘‘sand-paper’’ patterns presented successively to the same eye, and which differ only in that a square sub-area is displaced, will result in the apparent movement of the square. This moving square being invisible in the individual patterns. This effect is seen only for displacements less than about ¼ degree.

On the other hand, movement can also be seen when the square is made up of two different patterns both with a square outline, i.e., the square must first be recognized as a shape in each case before the movement can be seen. This would correspond to the long-range apparent motion effect. This view is not completely accepted by all workers in the field [Kolers, 1984].

From the point of view of image synthesis the important point is that there seems to be a mechanism for detecting motion independent of the recognition of shape. This mechanism operates when series of images along a path are presented rapidly, as in video or film [Adelson & Bergen, 1985]. We may therefore hope to be able to sacrifice spatial fidelity in order to get movement at the correct speed for a particular object on a computer display.

2.4.7 Artificial Displays and Viewing Conditions.

There are many kinds of artificial displays, even excluding exotic devices such as the head mounted displays mentioned earlier there are too many to even begin to discuss here [see, for example, Salmon & Slater, 1987]. They have one common feature however: limited bandwidth.

This bandwidth limitation affects both spatial and temporal detail. Aliasing (see §2.3.4) effects occur if the source contains higher frequencies than the display can handle. If the update rate of a display is too low then we can't hope to induce apparent motion effects. That is we will not have what is known in computer graphics as a 'real-time' display.

If we restrict ourselves to the more standard TV type displays, we have the following types:

0.5 - 1 Gigabits per second — high definition displays.

In high definition video displays we have 1000-1200 lines and up to 80 fields per second. In this case the resolution is high enough for distortions in output to be below visible thresholds. The problem is mainly one of producing the vast amount of output data. Both the spatial and the dynamic metric can be used for this purpose.

30 - 100 Megabits per second — broadcast TV.

Broadcast TV resolution is generally high enough to share most of the qualities of high definition displays but anti-aliasing is vital.

64 Kilobits per second - 2 Megabits per second — long distance video.

Movement cannot be shown in 'real-time'. The spatial metric would be useful to reduce detail, while the dynamic metric can be used to pick out moving areas which are important visually.

9.6 - 64 Kilobits per second — Remote terminals and telephone lines.

At this rate no movement is possible. The spatial metric can be used to increase the size of primitives by decreasing the detail required.

The limits to human spatial- and temporal-frequency sensitivity have been called a *window of visibility* [Watson, Ahumada & Farrell, 1986]. Watson *et al.* provide a useful synthesis of results from vision research specifically for time-sampled displays and computer imagery. There are limits to human spatial- and temporal-frequency sensitivity and these limits are relatively independent of each other. We can therefore place a limiting box over the

frequencies (ξ, η, ν) of the image $f(x, y, t)$. Those frequencies outside this range will be filtered out. If we take one spatial frequency for simplicity then the window passes all frequencies

$$\{\xi, \nu \mid \xi \leq \xi_l \ \& \ \nu \leq \nu_l\} \tag{2.17}$$

where ξ_l and ν_l are the spatial and temporal limits respectively.

> A more precise expression of this hypothesis is that the spatio-temporal distribution of contrast in the image is filtered at some stage in the visual system. The limits of the passband of this filter are ξ_l and ν_l. If, after passing through the filter, the two stimuli are identical, then an observer relying on the output of this filter will be incapable of distinguishing between the two.
> [Watson, Ahumada & Farrell, 1986, p. 302]

This window predicts the critical sample rates required in space and time to render motion accurately on artificial displays. For computer generated imagery it is suggested that various spatial frequency bands in such synthesized images be treated separately and to display only those whose velocity does not produce aliasing at the display device sampling rate.

Summary

- Computer generated animation depends on features of human visual perception to make the image synthesis problem tractable.
- It may be possible to exploit further features of human visual processing in order to get more realistic animation sequences cheaply.
- Uncertainty relations govern the accuracy with which position and spatial frequency can be resolved together.
- Fourier and convolution analysis can be applied to higher level perceptual functions, not just to the optical imaging which takes place in the eye.
- Movement is not perceived by freezing motion and looking for correspondences between static images. It appears that optic flow is directly sensed in a space-time domain.
- Spatio-temporal low pass filtering occurs at some stage in the visual system. This defines a *window of visibility* which limits spatial- and temporal-sensitivity.

§2.5 Information Channels: From Computer Model to Viewer's Mind.

The metrics introduced in the previous chapter are meant to be applied to animating scenes taken from nature. On the one hand the rich detail of such scenes makes them difficult to represent in a machine, while on the other hand the lighting effects are simpler than in artificial environments.

In this section we will discuss the information requirements of the spatial and temporal metric. As we have seen the human visual processing system does not draw such a sharp distinction, it is always a spatio-temporal system. Therefore the two metrics will never be independent of one another.

Although we have reviewed human vision qualitatively we still do not have quantitative answers to the following questions:

- *Brightness Thresholds:* What are just-noticeable-differences for various test patches over different regions of a scene? This is needed to decide cut-off, quantization and permitted noise levels.
- *Resolution Trade-offs:* What should spatial and temporal resolution be?
- *Motion:* How do viewers interpret errors in rendering motion?

We will not have have these figures either because they are still current research topics. However our implementation is really experimental. Therefore many questions can be answered simply by starting with what is available (i.e., the qualitative relationships which obtain). In a sense the success or failure of the various metrics will provide experimental values for some of the numbers which are lacking

2.5.1 Information Transmitted to the Viewer.

The characteristics of visual perception in artificial and natural conditions (both circumstances being of interest to us) is the subject of current research. But for the present we can make do with the qualitative aspects of such theory. The actual implementation will require *ad hoc* adjustment of parameters (but not of the theory). For the present we only want to know what the parameters are and what their interrelationships might be. Lacking a fully operational system, the actual parameter values will be irrelevant.

Information conveyed to the viewer goes through several processing steps in the combined

theory. Describing each of these steps means gleaning the information from the appropriate theory. The processing goes from a dynamic three-dimensional description to a limited resolution two-dimensional display sequence. From this the viewer is expected to reconstruct in their own mind a close approximation to the author's intended environment.

Some of the steps and corresponding theoretical disciplines are listed below. Of course this table is *too tidy*: For example, human visual perception, listed as the sixth step informs the way nature is described in step one, it is intimately connected with rendering (step 4).

No.	Nature of Step	Applicable Theory	Section
0.	Formulation of author's intention in some computer animation language	not considered here.	
1.	Simulating the Appearance of Natural Surfaces	Morphology. Modelling.	2.2
2.	Spatial & Motion Relations of Objects (Light-scene-observer geometry).	Geometry. Kinematics.	2.2.2 & 2.3
3.	Atmospheric Effects (scattering & absorption).	Atmospheric Optics.	2.2.3
4.	Synthetic Eye in Synthesized Nature (Rendering to simulate human vision).	Sampling. (see also 6 below).	2.3.4 & 2.4.2
5.	Artificial Viewing Conditions (VDU or cinema screen).	Display Engineering. Experimental (?).	2.4.7
6.	Human Visual Perception	Psychophysics & Low, Intermediate Level AI.	2.4
7.	Eureka! (Decoding the symbols).	Theory of Art. Psychology. High Level AI.	2.1

Table 2.1. Steps in the transmission of information to the viewer.

The steps one up to four are simulations on a computer and are thus also subject to the limitations imposed by computer hardware and software resources.

The actions from two to six can perhaps be viewed as a series of filters. Some actions are really forms of sampling, but since sampling is only accurate for band pass filtered images, it can be assumed here that sampling is a form of filtering. We should distinguish between

three kinds of filtering:

a. Filtering implicit in a realistic model of the natural environment (e.g. atmospheric effects).

b. Filtering due to display device and other viewing conditions (e.g. low pass nature of the raster display).

c. Filtering due to trade-offs in the human visual system (e.g. trade off between spatial and temporal detail).

and of course:

d. ‘‘Filtering’’ due to coarseness of data structures and lack of processing time (i.e., lack of detail on surfaces and lack smoothness in motion).

We could also regard the steps outlined above as stages in a connected series of information channels each with its own encoding.

2.5.2 The Spatial Metric.

The term ‘‘perspective effects’’ has long been used by artists to mean the way the appearances of things change with increasing distance from the viewer. It meant both the loss of spatial detail seen in objects, and the colour changes over longer distances.

The spatial metric is primarily dependent on distance. It measures perspective effects in the broad sense defined above. Although it will not here be *applied* to colour effects (see §2.4.5 & §2.5.3). The primary ‘cause’ of loss of detail is the inverse square law which describes the relative diminution of areas with distance. A secondary cause of loss of detail is *atmospheric perspective* which results because light travelling through the air is scattered and absorbed.

Under normal conditions the geometrical perspective effect predominates in the medium range of distances. It is of course easily modelled. The longer range is always dominated by atmospheric effects. Since, when light is transmitted in the atmosphere, the resolution drops to nothing and the object merges with the background long before the geometrically predicted vanishing point (at infinite distance!) is reached.

Kaufman [Kaufman, 1981] has produced a model of the way the human eye responds to

atmospheric effects when viewing the boundary between two contrasting fields from above (so the path radiance, L_P, of §2.2.3 is neglected). He makes use of a modulation transfer function (MTF) of the atmosphere and derives an MTF of the eye from a model of vision by I. Overington.

The spatial metric can be extended to take into account the curvatures of the surfaces of the objects being modelled. That is, sharply curved surfaces could be rendered in more detail. This is equivalent to having a higher sampling rate for surfaces with high spatial frequencies. Even if this is not done it is clear that as we decrease the level of detail being rendered, because the object being rendered is further away, the aliasing effects can occur unless the surface is low-pass filtered.

2.5.3 Colour Effects

The human visual system samples the optical wavelengths with three detectors. This means of course a great simplification for computer representation of surface spectral reflectance and light source spectral distributions. It should also be possible to apply the idea of a metric in the modelling of the differential transmission of the various wavelengths of light in the atmosphere. However, that is beyond the scope of this study (see also §2.4.5).

2.5.4 The Temporal Metric.

As pointed out previously the observer in nature is confronted by a dense optic array originating in the many nested levels of structure in the environment. Some of the elements of the optic array originate from objects with their own independent motion. Superimposed on these independent proper movements are the effects of the observer's own motion: these combined changes in the optic array constitute the optic flow field.

In an analogous way to the spatial metric, the dynamic metric can measure the speed with which the projected images move. This speed can govern update rates. The temporal metric can be extended to measure the various levels of optic flow effects. The optic flow is smoothly varying where the optic array originates in an object whose surface is reasonably regular (relative to viewing distance). Over these areas the higher order optic flow effects could be taken into account, this is analogous to incorporating surface curvature measures into the spatial metric.

2.5.5 Trade-offs Between the Spatial and Temporal Metrics.

When objects move their frequencies get altered according to Equation (2.16). However the passband of the *window of visibility*, Equation (2.17) remains unchanged. Therefore some detail must be lost. This detail may of course appear in an aliased form if it is not filtered out; that is, for particular velocities there are limits to the spatial-frequencies which contain useful information and which can be displayed without aliasing. If we could have some description of the objects in a scene in terms of the 2-D velocities of their image components then we would have identified the limits to the detail needed from such objects. The temporal metric will provide such a measure via optic flow analysis.

In this chapter the computational cost of implementing the ideas has not been discussed. It can be considerable in the case of some of the pre-filtering steps discussed in this sub-section (see §7.4.3).

Another way of approaching the trade-off is to consider the conditions under which apparent motion is sensed. We have seen apparent motion in rapidly changing images does not depend on feature recognition (§2.4.6) and therefore the sensation of movement will not be impaired if the moving image is blurred.

Conversely we could try to exploit the longer range motion perception which does depend on feature recognition. In this case we render images too slowly for the short range process but in such a way that the long range process operates. Such motion will appear jerky, but this may be acceptable in some cases.

§2.6 Summary and Conclusion.

We have argued in this chapter that realistic pictures do not necessarily depend on realistic and accurate physics. In the first place, the natural world as we perceive it cannot usefully be described by the laws of physics. In the second place, the various imaging systems which lie between such a natural world and its imitation on a computer display have their own limitations and possibilities. These characteristics have very little to do with the physics of nature.

Finally, our perception of these displays is an active process which strives to create order and sense. For example, real motion and full colour are seen where there are only static images and three-coloured phosphors. Those examples are familiar but there are other

characteristics of perception which might also be exploited. For example, the limits in sensitivity to spatial and temporal frequencies, and the perception of optic flow fields rather than sequences of static images.

The need for a theoretical basis for synthesizing realistic pictures remains even if physics by itself proves inadequate. Such a theory can use laws of ecological optics as its starting point. This theory can then proceed to allow for computer modelled three-dimensional worlds and artificial display techniques. In the end it will also have to cover the way these displays are finally perceived by the viewer.

This is what we have attempted to do in this chapter. We have ended up with qualitative definitions of the spatial and temporal priority metrics. These are refinements of what is so inadequately called ‘‘faking it’’.

Some properties of the two metrics are summarized in the following diagram:

	Spatial Metric	Temporal Metric .
Main variable:	Relative distance.	Relative angular velocity.
Effects:	Blurring of detail.	Update rate.
Refinements:	Atmospheric effects, Surface curvature.	Optic flow field, hierarchy of motion effects.

Table 2.2. Summary of the properties of the spatial and temporal metrics.

A *combined effect* of the temporal and spatial metric is found in the loss of spatial detail due to motion.

Thus far we have discussed providing a theoretical foundation for computer animation without mentioning computer science! The next chapter will remedy this. Once that is done we can formalize the qualitative definitions of the spatial and temporal metrics given above.

Chapter III

Object Oriented Representation in Animation.

Any attempt to deal with the complexity of computer animation should have a well founded and appropriate underlying abstraction. This chapter introduces and critically examines the object oriented paradigm as it applies to computer graphics and animation. Regarding parts of an animation as independent actors communicating via messages has an intuitive appeal. This initial appeal, elaborated to become object oriented animation, does seem to stand up to the closer scrutiny.

However, it was discovered that object oriented languages need to be extended if we want to describe the way physical objects (and imaginary ones) are made up out of parts. This new abstraction was added to Smalltalk.

This chapter includes two surveys of the relevant literature: §3.1.6 examines object oriented computer animation and §3.2.1 surveys work on animating articulated figures.

§3.1 Structuring Complex Programs and Data.

Complexity in 3-D computer animation arises from many sources: the representation of geometric detail, motion, interactions between actors, the user interface, and the complexity inherent in large systems. We want to deal with complex natural scenes. Animation vastly compounds the problem since it adds changes and interrelations over time.

We have introduced the spatial and temporal priority metrics in the previous chapters. We are looking for ways to structure our graphics and animation programs which will simplify the conceptual, programming and interaction tasks. The object oriented paradigm seems very useful for this.

The priority metric measures the level of detail required from each object in the animated environment. It demands that different hierarchical representations be used together. That is, different underlying representations should appear the same from the ‘outside’. In this chapter we shall show how the object oriented paradigm provides a most appropriate abstraction to describe the elements of the environment and their animated interactions.

The rest of this section (§3.1) introduces object oriented programming; it contains a survey of object oriented programming in animation (§3.1.6) and concludes (§3.1.7) with a comparison between the object oriented and functional approaches to animation. The basic requirements for modelling animated figures form the subject of §3.2. A brief survey of

relevant literature is followed by an exposition of the role of part-whole and coordinate hierarchies in representing physical objects.

In the following section the object oriented programming languages used in the experimental implementations, Smalltalk and C++, are compared (§3.3). We take as a working example how a new class of numbers, quaternions, can be introduced. This also serves as a discussion of how localized coordinate systems are implemented.

§3.4 shows how object oriented languages ought to be extended if we want to model real physical objects and still retain all the conceptual and programming advantages of object oriented programming. We analyse our use of part hierarchies in Smalltalk (§3.5) and follow it with a discussion of the broader implications of part whole hierarchies for object oriented programming (§3.6). The conclusion (§3.7) also mentions possible extensions to allow multiple views of the same object.

3.1.1 The Object Oriented Paradigm.

‘‘Object orientation’’ has been used rather loosely in animation to mean nothing more than using 3-D object models instead of doing 2-D key frame animation. This is not what is meant by the term here. We adopt the more conventional ‘‘definition’’ [Cardelli & Wegner, 1985; Rensch, 1982; Stefik & Bobrow, 1985] where object orientation is some combination of:

> *Data abstraction* (named interfaces and hidden local state) plus *object types* (or classes) plus *type inheritance* (attributes inherited from superclasses). Processing is done by objects sending and replying to *messages*.

Languages need not conform to all these characteristics to be called ‘‘object oriented’’.

Object oriented languages descend from Simula and are exemplified by Smalltalk [Goldberg & Robson, 1983]. Some established languages have also been given object oriented features (e.g. C++ [Stroustrup, 1986]). In Hewitt’s actor formalism [Agha, 1986] greater emphasis is placed on concurrency and message passing. The NeWS windowing system has shown how naturally PostScript supports object oriented programming for user interfaces [Adobe Systems, 1985; NeWS, 1987; Densmore & Rosenthal, 1987].

3.1.2 Abstract Data Types.

In object oriented languages the computing process is factored into objects. Each object is comprised of data elements and procedure elements. Objects are typically instances of some class, or, equivalently, objects belong to a type.

Procedural abstraction means that procedures can be invoked by naming them without regard to their internals, that is, without regard to how the particular part of the algorithm was implemented. This forms the basis of structured, modularized, procedure oriented programming.

Data abstraction is used for similar reasons in object oriented programming. The state and implementation of an object is hidden from other objects. Instead an object possesses a protocol of messages which form its only interface with the outside. In order to use an object one need only know the protocol, or equivalently, its class or type [Cardelli & Wegner, 1985].

3.1.3 Classes and Inheritance.

In most object oriented languages objects belong to classes (these classes being objects in their own right). Objects which are instances of the same class are similar in that they share the same interface and have the same structure. Class and inheritance is based on an analogy with both taxonomy and genetics.

Delegation is used in actor languages in preference to the notion of class [Agha, 1986]. This means that an object's message protocol includes those messages which can be delegated to prototypes or exemplars [Borning, 1986; Lieberman, 1986; LaLonde, Thomas & Pugh, 1986]. For example, if we were modelling horses, the Platonic ideal horse would be a prototype and a particular nag in the field would delegate the responses to some of its messages to that ideal prototype.

Classes in object oriented languages form a hierarchy. Subclasses are specializations of their superclasses, and they inherit all the characteristics of the superclasses [Cardelli, 1984]. Simple abstract classes characterize the higher levels of the hierarchy while more complex behaviours, in concretely useful classes, are found at lower levels. Even actor languages which do not have a notion of class employ inheritance to organize information.

The simple hierarchy of inheritance relationships can be extended to a network of relationships. This is referred to as multiple inheritance. In some systems a class need not inherit all

the features of a particular super class. This is even closer to the situation in biology where traits are distributed amongst individuals in a gene pool and can be inherited separately.

The class hierarchy can be taken to express an ‘‘IS-A’’ relation [Brachman, 1983]. Thus an integer IS-A number. More specifically, if the inheritance hierarchy is anything more than an implementation tool then it exemplifies the ‘‘IS-A-KIND-OF’’ relation: integer IS-A-KIND-OF number.

Programming in an object oriented language is a question of designing and implementing classes (or their equivalent in prototype based languages). A large problem is split into a number of hierarchies of classes.

3.1.4 Message Passing and Polymorphism.

Polymorphism can be used in a number of senses; rather perversely the meaning is usually that a single form can be applied to a variety of underlying types. For example: conventional typed programming languages allow the parameters of functions to have only one type. If this idea was strictly applied then addition would require a different function for each type of number and generalized routines, like head of an arbitrary list, would be impossible. Polymorphic languages allow for the same functions to accept many different parameter types. This idea can be inverted in a way which is more appropriate to object oriented programming: polymorphic types are types whose functions can be applied to many different types.

In object oriented languages the same messages can be sent to a number of different classes and the messages can have any type of argument. The messages need not be used to provide concurrent communication but might behave largely like procedure calls. However, unlike procedure calls, messages sending allows polymorphism without requiring the constant checking of parameter types [Ingalls, 1986]. The existence of class hierarchies must entail a certain polymorphism if related but distinct classes are to understand the same messages [*inclusion polymorphism* — Cardelli & Wegner, 1985].

The conceptual power of inheritance hierarchies derives, at least partly, from the way in which they allow automatic but controlled polymorphism for all subclasses. One largely knows the behaviour of an object if one knows the behaviour of its superclass. (The other major advantage of class hierarchies is of course that they allow code sharing).

Having polymorphic messages makes it possible to program by extending the language with new types. For example, it was very easy to add a new class of number, namely quaternions, to Smalltalk. These quaternions understand exactly the same messages, for multiplication, addition, square root, and so forth, as other numbers. Smalltalk can be said to exhibit ‘‘true’’ polymorphism because all objects are uniformly represented and *can* exhibit uniform behaviour without coercion or explicit type checking.

C++ does not use message lookup for procedure calls and is statically typed, but it does allow operator overloading, i.e., the same operator can have a number of different, but predetermined, types of arguments. Quaternions can be implemented just as elegantly as with Smalltalk.

3.1.5 Concurrency.

Events in nature happen at the same time. Simulating such an environment requires concurrent execution of the objects representing elements of this environment, at least in principle. Luckily ‘‘time’’ when applied to computer animation actually means discrete steps synchronized every twentieth of a (simulated) second. Thus one could simply service all objects sequentially in each time slot. However explicit support for concurrency can be useful and it is provided in Smalltalk. See Appendix D for a further discussion of this topic.

3.1.6 Actors and Animation.

Actors in a different guise are met in object oriented animation systems. Early examples are DIRECTOR [Kahn, 1976] and ASAS [Reynolds, 1982]. A succession of animation systems based to a greater or lesser extent on the actor formalism have been developed by Magnenat-Thalmann & Thalmann [1985].

Message passing actors have proved to be very appropriate for modelling 3-D animation. Object oriented animation has its origins with Smalltalk and this aspect of Smalltalk was influenced by Logo [Kay, 1977]. Logo is not object oriented: in Kay’s terms Logo is a data-procedure language. Whereas in Smalltalk the data and procedures are replaced by the single idea of ‘‘activities’’ which belong to families. New families are created by combining and enriching properties which are inherited as traits. This message-activity system is inherently parallel. The kinds of animation produced nevertheless still have a distinct Logo-like flavour.

A similar strong influence of Logo is apparent in Kahn's [Kahn, 1976; Magnenat-Thalmann & Thalmann, 1985] Director language. Like Kay he emphasizes that a computer language should reflect both the structure of its applications and the intuitions of its users. For animation this means that each entity should be a ''little person'' who communicates with others by means of messages. An animation as a whole is then produced by a number of parallel cooperating processes.

Kahn's animation system is a practical approximation to this ideal. There is a Universe which holds the actors. Each actor remembers its own actions and the Universe (the scheduler) merely sends a 'tick' message to them. At each tick an actor performs its actions and interactions for that time increment. The display messages are sent to a screen actor and these messages can also be remembered to make a movie.

It is significant that two problems addressed in the fourth section of this chapter, that is, part-whole relations and the different ways of regarding an object to be composed of parts, are mentioned as future research goals by the two previous authors. Kahn mentions the need for better primitives for dealing with composite objects and for constructing objects out of parts. Kay advocates the development of an ''observer language'' by which he means a language which allows objects to be regarded from different viewpoints with respect to what they are said to be composed of. He gives the example of a dog which ''can be viewed abstractly (as an animal), analytically (as being composed of organs, cells and molecules), pragmatically (as a vehicle by a child) ... ''

Both Kay and Kahn produced rather simple two-dimensional images. In ASAS [Reynolds, 1982] we get much more realistic three-dimensional graphics.

3.1.7 Functional Animation and Object Oriented Animation.

Functional programming provides a useful abstraction for computer graphics [Arya, 1986; Henderson, 1982; Salmon & Slater 1987]. A few points of contrast and similarity with object oriented graphics will be given but a detailed comparison is not a concern of this research programme. Both fields are new and developing rapidly so such a comparison would probably be premature.

Functional programming derives its power from giving functions first class status. Functions can be combined and manipulated just like any other object. Data structures are defined by means of constructor functions which make abstract data objects. Access is only via the

operations defined on the data objects. The usefulness of data abstraction has already been mentioned. In functional programming polymorphism and concurrency are also emphasized.

Pure functional programs are declarative. They can be reasoned about and manipulated with formal tools. One can consider programs as static objects. The meaning of an expression does not change as computation proceeds. This property, know as referential transparency, also allows functional languages great power in using parallel evaluation [Darlington, 1984].

It is much more difficult to reason about the behaviour of procedural languages in a formal way, and virtually impossible in languages like Smalltalk which depend on dynamic type checking.

Real objects persist while their configurations and attributes change over time (§2.2). Animation, as the mimicking of three-dimensional physical objects, depends on a notion of *state*. We have seen that this meshes rather well with the concept of actors and objects in object oriented programming. In functional graphics the emphasis is shifted to dealing with a sequence of *different* objects related by a sequence of transformations. This model of computation is found in key frame animation, which is mainly used for two-dimensional pictures. (Slater also discusses the fact that difficulties arise when using functional languages for programming interaction and when using attributes [Salmon & Slater, 1987, pp. 290-291]).

Reasoning about concurrent functional programs with history dependence, (this includes interactive programs) faces a fundamental theoretical difficulty: such programs are non-deterministic [Abramsky, 1984]. This could lessen one of the theoretical advantages of functional animation over object oriented animation.

This does not prevent functional programming from being used in practice in time dependent situations. Generally it is possible to ‘‘abstract away’’ the notion of time, and replace it with some idea of sequences over infinite lists [Arya, 1986]. To discover the relation between the figures in the list one refers to the functions which constructed them. The sequence of transformations of an actor over time can be established by the user on input, but generating the transformations from some three-dimensional model might prove more difficult.

So it seems that the virtue of functional programming often becomes its vice in modelling animation. The conceptual advantages of functional programs remain limited precisely because such programs describe a dynamically changing world as a (conceptually) frozen system of infinite sequences.

Functional programming is evolving, and the final conclusions regarding functional *versus* object oriented approaches to computer animation cannot yet be drawn. The extent to which a (possibly impure [e.g., Halstead, 1985]) functional approach can be elaborated for three-dimensional animation needs further investigation. On the other hand, object oriented animation already seems well suited to modelling changing objects executing concurrently.

§3.2 Animating Jointed Figures.

There are two aspects to modelling animated figures: producing computer representations which allow movement and controlling that movement. The first problem is addressed in this study since we are concerned with the appearance of moving figures at various levels of detail. The problem of controlling figure movements so that they appear realistic is an area of active research to which answers are only now appearing. In developing our chosen representation we shall be very concerned with providing an underlying mechanism which allows constraints on movement to be handled elegantly. We shall not be concerned with motion control directly nor with the related problem of modelling motion subject to various constraints. In spite of the circumscribed area of interest, a significant result, which is the need to extend object oriented languages to include a part-whole hierarchy, will be presented in §3.4.

At the simplest level we can regard people and animals as being composed of limbs connected by revolute joints. The challenge of animation lies mostly in making movement realistic [Badler, 1987]. A general abstraction is developed in the next section for incorporating the notion of ‘‘an object made up of active parts which are related via constraints’’. The purpose of this section was to provide the motivation for that abstraction.

3.2.1 Modelling Articulated Figures for Computer Animation.

This is a difficult problem which has attracted many researchers. IEEE Computer Graphics and Applications has devoted two special issues to the problem, both edited by Norman Badler of the University of Pennsylvania: November 1982 and June 1987. Apart from

material directly addressing computer animation, research in robotics and the design of legged vehicles is often also relevant. Of course a major inspiration is the success of traditional hand-drawn animation, many of whose principles can be applied to 3-D computer animation [Lasseter, 1987].

An early survey of the field is Badler & Smoliar [1979]. Various issues concerning the representation of human movement were explored. A large part of the paper is devoted to describing a human movement simulator which interprets Labanotation. Labanotation (which is also know as ‘‘Kinetography Laban’’ in Europe) is a notation for describing human movement. It is mainly used for dance but can be used for human movement generally [Preston-Dunlop, 1969]. The basic architecture of the simulator is of a network of special purpose processors — one for each joint.

Computer animators note the need for abstraction as a way of dealing with their rather difficult problem. Zeltzer presents the argument for creating an integrated view of computer animation and in passing gives a survey of the field [Zeltzer, 1985]. Zeltzer describes different kinds of abstraction useful for character animation. His terminology presupposes a constant awareness that motion is inherent in animation and for the non-animator the terms are more easily understood by adding the word ‘‘motion’’ before the word ‘‘abstraction’’ in every case.

- Structural abstraction describes the kinematic properties of the figure, i.e. the hierarchy of jointed limbs and their possible motions.
- Procedural abstraction describes the movements in terms of the desired results rather than the particular kinematic structure.
- Functional abstraction describes movements in terms parameterized skills or basic movements. For example: walking or grasping, which can be fast or slow.

Zeltzer also points out how a human figure can be modelled as a tree structure of joints and parts. The parts are embedded in a generalization lattice of attributes, this lattice being supplied by some sort of multiple class inheritance hierarchy (in the object oriented sense). He emphasizes that the complex modelled environment of an animated object has to be structured in some way which allows rapid testing for the proximity of objects. The description of objects in terms of a hierarchy of parts which also reflect levels of detail should go a long way towards meeting this need.

ThingLab [Borning, 1979, 1981] is a system for simulating physical objects (e.g., geometric shapes, bridges, electrical circuits, documents, calculators). In ThingLab, objects consist of parts. Multiple class inheritance hierarchies, and *part-whole hierarchies* are used to describe the objects and their interrelations. Parts are referred to symbolically by means of *paths* that name the nodes to be visited in proceeding down the part hierarchy.

In ThingLab the superclasses of an object are a part of the object: an object contains an instance of its superclass (in the type theoretic sense) as sub-part (in the sense of part versus whole objects). The class which describes such a superclass part is a subclass of the normal part description class. Apart from this notion of multiple superclasses ThingLab also employs prototypes to provide initialized instances of objects.

The major contribution of ThingLab is a system for representing and satisfying constraints which exist between the parts. However, when it comes to providing a tool for modelling complex objects confusion can arise. Our work extracts, analyses, and refines the concept of a part hierarchy first encountered in ThingLab. In particular we draw a clearer distinction between class hierarchies and part hierarchies. We dispense with prototypes, not that classes are preferable to prototypes. It is just that Smalltalk already has classes.

3.2.2 What is meant by a Part Hierarchy and Why is it Important?

Things are often described in terms of parts and wholes; the way the division into parts is made depends on the purpose of the analysis. A part is a part by virtue of its being included in a larger whole. A part can become a whole in itself, which can then be split into further parts. In this way we build up a hierarchy of parts and wholes, which we have called the *part hierarchy*. Rather than attempt a formal description of part-whole relations [Smith, 1982] we shall present a series of illustrative examples.

We distinguish between a mere *collection,* or additive whole, or heap, (e.g., a bag of marbles, a pile of electronic components) and a more *structured whole* (e.g., an animal, a wired-up electronic circuit). To the former we apply set theory, to the later a part hierarchy.

It is also useful to distinguish between extensive *parts* — components, fragments, constituents, pieces — and non-extensive *attributes* — features, aspects, moments. For example: A table is made up out of a flat top and four identical legs placed in the corners, these are the parts. A table also has a colour, which is an attribute rather than a part. We shall be more concerned with parts than attributes.

Part-whole analysis is crucial to science and technology. Pirsig [1974] provides a very good example of the hierarchical description of the assemblies and subassemblies used in engineering:

> A motorcycle may be divided for purposes of classical rational analysis by means of its component assemblies and by means of its functions.
>
> If divided by means of its component assemblies, its most basic division is into a power assembly and a running assembly.
>
> The power assembly may be divided into the engine and the power-delivery system. The engine will be taken up first.
>
> The engine consists of a housing containing a power train, a fuel-air system, an ignition system, a feedback system and a lubrication system.
>
> The power train consists of cylinders, pistons, connecting rods, a crankshaft and a flywheel.
>
> The ignition system consists of an alternator, a rectifier, a battery, a high-voltage coil and spark plugs.
>
> That's a motorcycle divided according to its components. To know what the components are for, a division according to functions is necessary ...

Parts are also met in the those branches of computation where physical objects are represented, for example, *model-based computer vision* and *computer graphics*.

The example which will be used to test and illustrate our implementation is the same one which is used for the spatial detail experiment (Chapter 5): a stick figure. It should be noted that this very stick figure has importance not only in animation but is also found in a Marr's computer vision research [Marr & Nishihara, 1978] (see Figure 5.1). The particular stick figure we shall discuss, called ''joe'', has numerous parts arranged hierarchically. For example, joe's legs have feet which have toes, and toes consist of phalanges.

Model-based vision draws upon the work on knowledge representation in artificial intelligence [e.g., Brooks, 1981]. *Frame-based* representation [Fikes & Kehler, 1985] has similarities to the object oriented approach. Frames describe parts and attributes by means of *slots*.

One of the standard texts on computer graphics [Foley & van Dam, 1982] devotes a chapter to ''Modelling and the Object Hierarchy''. The proposed new graphics standard PHIGS (Programmer's Hierarchical Interactive Graphics System) [1986] organizes objects in a structure hierarchy. Both *structure hierarchy* and *object hierarchy,* as used above, are synonyms for our part hierarchy.

PHIGS also has the novel concept of *inheritance* on a part hierarchy where attributes of the

whole are inherited by the parts. E.g., the legs of the table could inherit the colour of the whole. The requirement is not quite as general as it at first appears. This ‘‘inheritance’’ is only used when the whole structure is traversed from the root down in order to display it. Thus the wholes are always accessed before the parts and the attributes can therefore be stacked.

We adopt the policy that information is stored in the part hierarchy at its corresponding logical level: Information about the whole is not stored in the parts, information about the parts which is not modified by the whole remains with the parts. Ideally the whole knows the parts but the parts do not know of the whole.

3.2.3 Coordinate Transformations.

Coordinate transformations play a vital role in computer graphics and animation. Scene composition, rendering and motion would be impossible without them.

Each object is modelled in its own coordinate space and is placed in its correct position relative to other objects in the environment by means of its own coordinate transformation. The camera is similarly positioned in the modelled environment. To render three-dimensional models of objects on the display they have to be transformed into the camera coordinate space and then projected by a perspective transformation. The movement of rigid objects is achieved by altering their coordinate transformations.

Each part of the hierarchical representation of an object has its own changing local coordinate system. In animation and rendering these coordinate systems have to be related to one another and to the world coordinate system.

The most general motion of a rigid body in the world coordinate system can be described as the combination of a translation of a fixed point in the object combined with a rotation about that point. (In a more general formulation this is Chasles’ theorem [Goldstein, 1980]). A freely moving rigid body has six degrees of freedom. The most general motion if one point is fixed is just rotation. Rotations are thus the most important transformation when implementing a system of localized coordinates for an animal’s limbs (or robot’s for that matter).

The standard formalism for 3-D computer graphics has tended to be homogeneous coordinates [Newman & Sproull, 1979]. Transformations can be expressed in a single 4*4 matrix which includes terms for translation, rotation and perspective transformations. Hamilton’s

quaternions, although rather neglected since the turn of the century [Goldstein, 1980: ‘‘musty mathematics’’], make a computationally efficient formalism which is also easy to understand. Compared to homogeneous transformation matrices quaternions have fewer redundant terms. An introduction to Quaternions is given in the next section (§3.3) where they are used as a running example in a comparison of object oriented programming in C++ and Smalltalk.

Each limb of the modelled figure has a local transformation quaternion (actually unit quaternion) which specifies its rotation with respect to the coordinate system of the limb to which it is attached. It also has a quaternion (actually a vector) which specifies its position relative to the origin of that coordinate system. This vector does not change for most limbs, only those which form the root of the hierarchy of coordinates reflect the changing translation of the actor as a whole. The synthetic camera requires a similar pair of quaternions. Perspective projection is achieved by bringing all objects into the camera coordinate system and then dividing by the distance along the viewing direction (invariably the z-axis coordinate).

§3.3 Smalltalk and C++.

Having discussed a number of theoretical issues concerning object oriented languages, we can now give a brief taste and comparison of the object oriented languages used in this research project: Smalltalk and C++.

The initial implementation and all the figure animation work was done in Smalltalk-80. The landscape generation and optic flow animation was done in C++. The choice of Smalltalk and C++ is motivated in § 3.3.1 and §3.3.2. We illustrate and compare their use while at the same time presenting the implementation of quaternions to describe coordinates and their transformations (§3.3.3).

3.3.1 Why Smalltalk?

For an initial implementation, like the stick figure of Chapter 5, one wants to try new ideas out quickly with enough realism to draw conclusions about feasibility. Smalltalk [Goldberg & Robson, 1983] is an object oriented language that makes a very attractive prototyping tool. A great deal of effort has gone into designing its programmer and user interface. When the application is a graphical simulation there is seems to be little to beat it. In order to include an object hierarchy and still retain the spirit of modularity, object oriented languages

should be extended to support part-whole relationships as discussed in §3.4.

The following are key features of Smalltalk:

- Graphical user interface.
  The user interface already provides the basic data types for graphical bitmap display and mouse interaction.

- Powerful development tools.
  Programming in Smalltalk is designed to be an incremental activity. The whole library of existing programs can be examined with ease. Powerful interactive debugging is provided. Since binding is dynamic there is no need to link new methods.

- Smalltalk is easily modifiable.
  All system classes (including, for example, the compiler) are accessible and can be modified or, preferably, extended by adding sub-classes.

A disadvantage of Smalltalk is that a number of new concepts and approaches to programming have to be learned. Smalltalk programming consists in defining data types or classes, or rather refining existing classes. Although the language itself is small, one has to have a good grasp of existing system classes. Once these classes are understood however, they assist greatly in writing new programs. We shall give details of the use of an object oriented language for animation in Chapter 5.

Other restrictions of existing Smalltalk implementations, such as restricted object memory, lack of colour primitives, slow speed and poor support for parallel processes have been, or are being, alleviated [Miranda, 1987].

3.3.2 Why C++?

There are two main reasons for using C++ once the choice has been fixed on an object oriented language: computational speed and compatibility with C (that is old C). What is sacrificed to get this, comparing C++ with Smalltalk, is dynamic binding and true polymorphism. A somewhat less serious loss with current implementations is that debugging becomes much more difficult.

Quite apart from its value as an object oriented language C++ provides facilities which are lacking from (old) C, e.g. type checking for function arguments, facilities for avoiding

macros such as inline functions, and typed constants.

But we are interested in C++ because it is a object oriented language: It allows one to define new types and subtypes. One can enforce information hiding (it provides better checking of this than Smalltalk) by declaring messages and data to be private to a class. Operators and function names can be overloaded, which allows a clean syntax for new types. ‘‘Virtual’’ functions allow one a limited form of dynamic binding: the binding of certain declared recipients of messages can be looked up at run-time, provided they share a common superclass. Operators include the usual ‘+ - ∗ /’ as well as C specialities like ‘>>=’ and even the subscription operator ‘[]’ and function call operator ‘()’.

C++ does not provide a garbage collection mechanism for dynamic storage. One can however declare constructors and clean-up functions which are called automatically when a variable comes into or goes out of scope.

Generally C++ has provided a fast object oriented language which can co-exist with systems designed for C. The fact that it is a preprocessor which increases the compilation-link cycle time and the lack of a source level debugger does mean that development is slower than with Smalltalk. The modularity of the object oriented language has made the task of developing a fair sized experimental system (8000 lines of code for the landscape synthesis suite) quite manageable.

3.3.3 Quaternions.

Hamilton’s quaternions were introduced to as a way of representing coordinate transformations (§3.2.3). Quaternions consist of a scalar part and a three-dimensional vector part [Hamilton, 1969, Pervin & Webb, 1982]. Those with the same unit vector part are isomorphic to complex numbers. Quaternion multiplication combines scalar and vector multiplication and is non-commutative in general. The famous formula discovered by Hamilton in 1843 shows the scalar result of multiplying the unit vectors:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{i}\,\mathbf{j}\,\mathbf{k} = -1$$

Quaternions represent rotations in terms of the axis of rotation and the angle about that axis. The effect of applying a quaternion is far easier to visualize than the more common Euler angles. Quaternions represent both the operands (vectors) and operators (rotations and translations) uniformly. Rotations can be combined by multiplying the quaternion

representations. Quaternions are more efficient computationally than rotation matrices because they don't have the same number of redundant terms [Shoemake, 1985].

Quaternions are thus chosen since they are *powerful* and provide a representation of 3-D rotation which is *easy to understand*. They are not too general for 3-D transformations and so are computationally *efficient*. Quaternions provide a uniform representation of operators and operands; a vector can simply be regarded (and implemented) as a quaternion with a zero scalar term.

3.3.4 Adding Quaternions to Smalltalk and C++.

The inherent polymorphism of messages (or overloading of operators) in Smalltalk allows easy and elegant implementation. The normal arithmetic messages can be implemented for quaternions; combined with a few coercion messages that is really all that is required to add quaternions as a subclass of numbers (Figure 5.4 shows multiplication). Quaternions then become fully integrated in an extended system-wide concept of Number.

A minor complication is having to represent translation and rotation as separate transformations. Once again one can simply define a new class which incorporates both and the rest of the system need never know of the true implementation.

For greater efficiency, unit quaternions, which are used for rotation transformations, are given special treatment. This is quite easy in Smalltalk and is transparent to the user. It is analogous to the way small integers are treated in the standard system. Unit quaternions are declared as a subclass of quaternions. The general messages are then handled by the superclass but specialized messages and more efficient implementations are dealt with by the subclass. For example, the inverse of a unit quaternion can be found without recourse to division and so this message is re-implemented in the subclass.

Much the same comments apply to C++ since it also allows operator overloading, subclassing and the possibility of overriding the methods defined by the superclass in the subclass.

In the case of quaternions the lack of dynamic binding does not matter greatly. Dynamic binding obviates the need for re-linking when the underlying representation of a type is changed. It is in the nature of a number system like quaternions that the implementation and messages understood does not change very often, although new optimizations may occasionally be introduced. Dynamic binding also allows a particular place holder (variable name) to

contain a number of different types of objects. The binding is made at run time when the variable has been instantiated. Quaternions can reasonably be interchanged only with scalars and vectors and for such a limited set of possibilities coercion rules can be drawn up which the C++ compiler will then invoke automatically. It has generally been the case that the types of objects used in arithmetic can be determined at compile time.

The use of automatic coercion does have some hidden snags. Unless one is careful a lot of conversion can happen from unit quaternions to quaternions (this is cheap) and then back again (which is expensive since it involves division and square root calculation).

§3.4 Representing Physical Objects: The part hierarchy.

From §3.2 it can be seen that there are two salient features of an animated figure which we must capture: (a) it is composed of parts (limbs) which depend on each other, and (b), these limbs can can move subject to various constraints. The notion of objects being constructed from parts, which in their turn may consist of smaller parts, is employed in very many other situations too (for example, in engineering design we speak of assemblies and subassemblies). Since this is such a fundamental property we would like it incorporated in our object oriented abstraction which, as I have argued, has much to recommend itself.

But, Smalltalk and many other object oriented languages fail to provide the facility to describe objects in terms of their parts. Or more accurately, when we want to model objects consisting of parts in Smalltalk, and many other object oriented languages, we are confronted with a dilemma: either sacrifice the data encapsulation properties of the language or utterly flatten the hierarchy whole which consist of parts.

Composite objects are provided in Loops [Stefik & Bobrow, 1985]. The emphasis seems to be on providing a uniform method for instantiation. For this purpose a simple type system is developed where classes are specified for each part. Using these classes the parts can then be correctly generated when a new instance of the whole is required. We developed a similar system (§3.5.1), but it tends to rather inflexible, and more elaborate parameterized typing might be necessary.

Once extended to include a part-whole hierarchy, object oriented languages, with their simulation (Simula) pedigree, become ideal choices for animation of natural environments.

3.4.1 The Dilemma Posed by Parts in Object Oriented Languages.

Data abstraction is a fundamental aspect of object oriented languages (§3.1.2). Data abstraction can be summarized as meaning explicit interface protocols and a hidden local state. When an object is assembled from its parts these parts are no longer independent. A part belongs to the local state of the whole and the interface is mediated by the owner.

If this requirement is strictly interpreted the existence of the parts should become invisible to the users of the whole. The whole protocol which a part understands, some of which it may implement perfectly adequately, will have to be re-implemented as the protocol of the whole. The net result is that the part hierarchy is replaced by a single monolithic whole as far as the external world is concerned.

On the other hand, if we did not include parts in the local state of the whole, external users could explicitly request the part and then modify it. The resulting changes could violate the integrity of the whole. The response of a part would be also be independent of its position in the whole.

This is the dilemma. We would like the parts to remain visible, while at the same time access is mediated by the owner on the next higher level.

An Example of the Dilemma in Smalltalk.

Parts can be identified with a particular kind of instance variable, that can be accessed via messages to read and assign the parts. In standard Smalltalk the parts can be accessed by putting together the message selectors of the various parts.

In our example of a stick figure (see Figures 5.2 and 5.3) we define the following classes:

> Person with instance variables: chest, leftLeg, etc.
> Foot has instance variables: bigToe, secondToe etc..
> then:
> joe ← Person new. "Create an instance of Person called ‘joe’"
> joe leftLeg. "Return contents of the left leg instance variable."
> joe leftLeg lowerLeg foot. "Return the left foot."
> joe leftLeg lowerLeg foot bigToe wiggle. "Wiggle left big toe."
> "(provided toes understand ‘wiggle’)"

This standard syntax has the disadvantage that the actual instance variable objects themselves are handed out and the sender of the message can modify them at will. This subverts the idea that objects can hide and control their local state.

The alternative is to disallow the direct part access messages. But then the top level object needs to implement the whole message protocol for all its parts. For example, to wiggle the left big toe the class Person would need a message like 'leftLegLowerLegFootBigToeWiggle' in its instance message protocol. This collapsed message utterly flattens the part hierarchy and so removes the conceptual advantages of factoring that knowledge in an intuitive manner.

One could attempt to intermix direct access to instance variables in ''safe'' cases, and use collapsed messages (e.g., leftLegLowerLegFoot) when access must be controlled, in an *ad hoc* manner. This is really worse, because the external world needs to know when to send collapsed messages and when not. If 'joe' in our example above, gets his foot encased in plaster then a collapsed message would be needed instead of direct access and all callers would have to know this.

3.4.1.1 Class Hierarchies Do Not Provide Part Hierarchies.

Neither single nor multiple class inheritance seem to have any bearing on the part-whole dilemma mentioned above. Classes deal with specializations of objects. In single inheritance we have a partial ordering which provides the same structure as can be expressed by sets and subsets. Even with multiple inheritance only one set of instance variables can be inherited for each superclass. Thus, even if the class ''Table'' inherited legs four times over along different paths, it would still only have the characteristics of one leg.

If multiple inheritance were extended so that the instance variable data were somehow kept separate, the parts would still be defined at the same level as the whole. That is, the same object would contain both the complete part protocols and the protocols corresponding to their structured interaction.

3.4.1.2 Parts in Smalltalk.

The standard Smalltalk system provides for objects to have dependants. The normal operation is that parts are given backpointers to their owners. The owner can then be informed of

changes. However:

- Parts have to know which changes are significant to the owner. (i.e., high level knowledge at lower levels).
- If parts forward all changes indiscriminately to the owner object then the owner must again contain the equivalent of the protocols of all the parts.
- It creates circular structures, with attendant maintenance problems.

This system of storing knowledge of the whole in a part which is then handed out, is useful where the part hierarchy is shallow and where it is simpler to hand out parts to the external world. An example of such a ‘‘part’’ is the *model* in the Smalltalk model-view-controller mechanism.

3.4.1.3 Prototypes and Delegation.

As will be seen in §3.4.2, our *implementation* of a part hierarchy makes use of message forwarding. Message forwarding is a powerful general notion which can also be used to implement delegation, which has long been used in actor languages in preference to the notion of class (§3.1.3).

It is in this sense that multiple inheritance was implemented by means of parts in ThingLab (§3.1.6). This creates a rather blurred distinction between a part hierarchy and a class hierarchy which we want to avoid. The notion of class is retained, as is the possibility of multiple class inheritance.

3.4.2 A Mechanism for Modelling Objects with Parts.

We have given a concrete example of incorporating parts using standard Smalltalk (§3.4.1). This provides the motivation for making a small change to the standard syntax (§3.4.2.1) which does allow a full part hierarchy to coexist with data encapsulation. The implementation of the extended language (§3.4.2.2) is achieved by a rather simple modification to the standard system.

For the rest of this section we shall be concerned with Smalltalk, but many of the remarks will apply to other object oriented languages.

3.4.2.1 A Syntax and Semantics for Manipulating Parts in Smalltalk.

We want to recognize that parts are objects in their own right — active first class objects. But *without violating the principle of encapsulation.* A whole is allowed to hide its parts, even pretend it has parts which do not in fact exist as such. The whole has additional properties arising from the interaction of the parts, which should not be stored in the parts.

Our solution uses the notion of censored access. The Whole forwards messages, possibly censored, to parts, and receives answers, also possibly censored, from parts. This necessitates a new kind of message selector: a *compound message selector* and a new concept: *message forwarding* of possibly censored messages.

Let us call the message selectors of standard Smalltalk simple selectors, whether they be unary, binary or keyword selectors. A compound selector is then a <path>+‘‘.’’+<simple-selector>. The path is a series of one or more part names also separated by full stops (‘‘.’’). Part names always begin with a lower case letter. The first part name in a path (its head if it were a list) is called the prefix. A path provides a way of referring to a part further down the part hierarchy.

Our example (§3.4.1) then becomes:

```
joe leftLeg.    "Return the left leg instance variable."
joe leftLeg.foot.bigToe.    "Return the left toe."
joe leftLeg.foot.bigToe.wiggle.    "Wiggle left big toe."
```

The semantics associated with the construct is one of message forwarding. Either a class understands a full compound message selector or it does not. If it does then the method corresponding to the compound selector is executed and the result returned, this may involve sending a censored version of the message to the part.

If the compound message is not part of the protocol of the object then the prefix is stripped off. The rest of the (possibly compound message) is forwarded to the instance variable named by the prefix. In that case the answer to the forwarded message is also the answer to the compound message as a whole. This definition is clearly recursive.

If the Smalltalk convention of *private* messages for information hiding is accepted then parts can always be accessed by means of automatically generated private messages. The message is then forwarded, not to an instance variable, but to the *answer* returned by sending a part access message. This means that the parts need not necessarily be instance variables.

They can be generated as and when needed. This could provide multiple views of the same object without storing all versions explicitly. This further extends the principle of information hiding.

3.4.2.2 Implementing the Part Hierarchy. (see also appendix A).

To implement the part hierarchy the following changes are needed:

1) Change the compiler to allow compound messages.

2) Extend the message selector class (Symbol) to provide access to the various parts of the compound messages.

3) Implement the mechanism for forwarding messages to parts.

4) (Optional) Add a class initialization method to add private messages automatically to access all parts and instantiate new parts depending on a predetermined dictionary associating classes with parts.

The extended Smalltalk with multiple inheritance [Borning & Ingalls, 1982] is widely distributed. This provides (1) and most of (2) above, while (4) is an elementary exercise.

It is (3) which is the heart of the system. It uses a powerful general mechanism: message forwarding. When a compound message is initially sent it is only understood if the object needs control over that aspect of its part structure which the message accesses. If the compound message is not understood then the part named by the message prefix can safely handle the message. Code is then compiled just to forward the message. If the same message is sent again the code already exists and the message will be understood.

The forwarding of messages can be overridden in a completely natural way if it is later decided that an object *does* need control over that aspect of its parts. The controlling method is simply added with the compound selector which names the part, and it replaces the forwarder.

§3.5 Experience with Using the Part Hierarchy.

The importance of a part hierarchy in computer graphics and vision has been mentioned in the the previous sections, as was the example of a stick figure. In this section we analyse how our extension to Smalltalk worked in practice. The part hierarchy described the structured figure rather well (§3.5.1) but it is not designed for a collection of stick figures (§3.5.2).

3.5.1 Results from Modelling a Single Stick Figure.

A moving stick figure and camera was implemented using the part hierarchy defined above (see Chapter 5). The following points arose from this experiment:

1) Parts are compatible with the existing multiple inheritance implementation.

2) For deep hierarchies the message selectors become rather lengthy.

3) Some, more or less elaborate, form of typing is required for the parts of an object (mainly to instantiate parts automatically).

4) The class inheritance hierarchy can interact in unexpected ways with the modelling hierarchy.

The second point means we might require some more syntactic sugar by way of abbreviations. In the case of parts which are common to a number of subclasses it is, at present, possible to define an abbreviation explicitly in the superclass. For example, all limbs of the stick figure have a proximal joint which implements the local coordinates. A common message to the proximal joint is to alter its orientation, thus the superclass of limbs implemented the abbreviation ‘‘orientation:’’ for the compound message ‘‘proximalJoint.orientation:’’.

The third point, concerning typing, uncovers an active area of current research, which is beyond the scope of our investigation. The need for an instance variable type arises when a new object is created and its parts have to be instantiated. In order to instantiate the parts their classes have to be known. This can be coded as an initialization message to the new instance, but a more general solution is to associate a default class with each instance variable. When a new object is then created, the creation message is also forwarded to these classes in order to create the parts.

The fourth point concerns the interaction between the class inheritance hierarchy and the

modelling hierarchy. Message forwarders are treated as normal methods and are inherited by subclasses. Consider message forwarders created in the class of the receiver of the compound message and not in the (super) class where the instance variable is actually defined. If the superclass is modified to intercept the compound message it will have no effect in those subclasses which have already acquired message forwarders. This would cause unpredictable results which depend on the past execution history. This problem is aggravated if message forwarders are made invisible to the user. The solutions are obvious and easy to implement.

3.5.2 Elements of a Collection are not Parts.

In our experimental implementation of stick figures the one situation where the part hierarchy was at a distinct disadvantage was with collections of stick figures. This is perhaps not too surprising since wholes are meant to be more structured than sets. The collection itself has an associated coordinate system in which the figures are embedded, it may even have a graphical appearance. Thus the collection is a kind of ‘‘Actor’’, and in some ways a whole just like a single stick figure.

The problem with using parts is two fold:

- The representation of a variable number of parts is rather difficult.
- The requirement that parts have explicit names, otherwise such an advantage, is here a liability. Objects in a collection are nameless, or at least their names change.

Representing a variable number of parts can be achieved in a number of more or less messy ways: a class for each cardinal number, allowance for empty part slots, or having indexed anonymous parts. All these options were attempted but had to be rejected.

If classes are dispensed with and prototypes used instead, some of these objections might be met. Each particular collection with a fixed number of elements could perhaps have its own part access messages. However, an appropriate solution is provided by multiple inheritance of both a *Collection* to provide access to the objects and a basic class of figure *(Actor)* to convey the appearance and position of the collection. This was the strategy adopted. The individuals in the collection were accessed with the normal messages used on collections.

§3.6 Implications of the Part Hierarchy for Object Oriented Languages.

At this stage of the chapter we have established the general importance of part hierarchies in natural philosophy and engineering. We have shown how a simple extension of Smalltalk can provide, fairly elegantly, a part hierarchy mechanism. This mechanism has been illustrated and criticized by means of examples drawn from three-dimensional modelling. We are now going to characterize the results of this effort.

3.6.1 Part Hierarchies and Class Inheritance Hierarchies.

When it comes to a single inheritance language, the question is really how orthogonal the concepts of part hierarchy and class hierarchy are. That the *implementations* can have mutual dependencies has already been seen.

The class hierarchy can be taken to express an ‘‘IS-A-KIND-OF’’ relation: integer IS-A-KIND-OF number (§3.1.3). The part hierarchy on the other hand expresses an ‘‘IS-A-PART-OF’’ relation, or more accurately since the part does not know its owner it expresses the reversed ‘‘HAS-A-PART’’ relation.

If we adopt this view then the two concepts are orthogonal. The class hierarchy provides generalization and specialization, this has nothing to do with structured building up of objects from parts, except that general objects may sometimes have fewer parts than their specialized subclasses.

It has already been seen that delegation provides an inheritance mechanism. Message forwarding, which we use to access parts, can be used for delegation. Therefore message forwarding can be used to implement multiple inheritance [Borning, 1986]. If we retain the notion of class (or maintain a distinction between prototypes and other objects), then we can regard this as an implementation issue and not a fundamental relation.

Multiple inheritance is preferred when objects are unstructured collections. When objects have a fixed structure, or when parts are subservient to the emergent properties of the whole, then a part hierarchy is better.

A part hierarchy is thus a valuable extension to both single and multiple inheritance object oriented languages.

3.6.2 Part Hierarchies and Encapsulation.

At first sight it might appear that part hierarchies violate the principle of data encapsulation and information hiding, because they provide access to the parts of an object. In our view the opposite is actually the case. Part hierarchies provide better control over access to the parts than is found in many object oriented languages.

It is true that the internal structure of the object is made visible. But this is strictly mediated by censored messages. These messages allow full control and even allow the whole to pretend it has parts, by providing messages, which are not in fact explicitly implemented.

The need to see structure of an object is similar to the need for ‘‘grey-boxes’’ rather than ‘‘black-boxes’’ in engineering. Depending on the level at which we are designing we need more or less of the structure of our wholes to be visible.

3.6.2.1 Phantom Parts.

A whole can pretend to have parts which are not actually stored as such [Borning 1979, ‘‘virtual parts’’]. Since access to the parts is only via censored messages the responses to these messages can be generated on the fly, rather than stored. A rectangle can be stored in terms of a top-left and bottom-right corner, but it can equally well pretend to have a centre which can be read or modified.

In animation the ability to have parts which are dynamically implemented allows great freedom in representing anomalous structures (such as the grin of the Cheshire cat which gets bigger and bigger while the cat disappears).

It is apparent that doing this would have been impossible if we had sacrificed data encapsulation. These different names (like ‘‘phantom’’, or ‘‘imaginary’’, or ‘‘virtual’’) for such parts refer purely to an implementation issue: to the outside of the object the distinction does not exist.

§3.7 Conclusion.

This chapter has been concerned with object oriented programming and its use in animating natural environments. We have argued that figure animation makes two basic demands on a system: (a) the system should allow figures to be built up as a hierarchy of parts and (b) provide for constrained motion of these parts relative to one another. It was emphasized in the

literature survey that this only provides the basic building blocks for a complete animation system.

Motion is provided by means of changing coordinate transformations. Quaternions provide this facility in intuitively appealing and at the same time computationally efficient way. The introduction of quaternions as a new type of number can be done rather elegantly in object oriented languages, because programming such languages is centered around the addition of user defined types or classes.

The basic requirements for part hierarchies have been incorporated in object oriented languages in a way which strengthens the data abstraction properties of these languages. Such part hierarchies are most useful in simulation, where we a concerned with structured physical objects.

Part hierarchies provide facilities not provided by class hierarchies. This is because class hierarchies provide less structure than part hierarchies, they provide the structure of sets, and sets contain no notion of relative position or multiplicity. Part hierarchies are less useful for describing sets and collections and multiple inheritance may be needed to deal with these. Part hierarchies can be regarded as being an orthogonal notion to class inheritance hierarchies.

Chapter IV

The Formulation of the Spatial and Temporal Priority Metrics.

We first develop a definition of *detail* (§4.1). We then derive formulations of the spatial and temporal priority metrics which can be applied in practice to synthesize images which have adaptive detail. The simple spatial metric depends on distance, it can be extended to allow for atmospheric effects. The simplest form of the temporal metric allows adaptive updating of moving images. The more complex versions allow the adaptive approximation of 3-D motion by 2-D image movement and distortion. There can also be a trade-off between spatial detail and movement accuracy.

The definition of the abstract data structure which interacts with the metric is given (§4.4). The use of fractals is discussed as an illustration.

§4.1 Introduction: What is Detail?

This chapter provides a mathematical formulation of the spatial and temporal priority metrics. These metrics were introduced in Chapter 1 as measures of the importance of features in the modelled environment to the viewer. The vague but general term ‘‘importance of a feature to the viewer’’ is now converted to ‘‘amount of detail contributed by the feature to the image’’. This is more precise, but it is also more restricted.

The tools and background for the formulation we are going to present were given in Chapter 2. The first section presented an aesthetic view which equated realism with a subjective human sensation of conviction of truth, rather than an objective criterion of physical correctness — *verisimilitude* rather than *veracity*. This view was then expanded with an analysis of perception of nature and artificial display devices.

Chapter 3 was concerned with the issue of modelling animated physical objects in a computer. The discussion was more concerned with general issues than specific data representations. Physical objects were seen to consist of assemblies of parts. Changing objects maintain their identity while changing their state. These features were captured very well by object oriented programming or actor/message semantics. In this chapter we will make use of another feature, common to both object oriented and functional programming: abstract data types. This is a very practical way of specifying the essential behaviour of data types.

These various sources will now be drawn together. However there is a difficulty which cannot be avoided nor can it ever be completely overcome: a precise definition cannot capture

all the senses of an intuition. In view of this difficulty we will accept a definition on the basis of its practical utility. The definitions of the spatial and temporal metrics derived in this chapter will be tested by the experimental applications of the subsequent chapters.

It would be very difficult to give a complete integrated mathematical theory of the way we see natural scenes which have been rendered on artificial displays. Aesthetics is not quantified and many things about human perception are still unknown. The analysis of visual requirements which follows will therefore be limited to low level vision, sampling theory and geometry†. This will still provide a number of interesting and applicable results.

By importance of an object we will understand ‘‘the amount of relevant detail’’. This can be taken as an ‘‘axiom’’ provided the terms used are further defined. The axiom is motivated in Chapter 2 and restricted by need for a mathematical formulation. The ‘‘axiom’’ requires two definitions however: A definition of *detail* and of *relevant*. The rest of this section is concerned with defining *detail*. *Relevance* is a more qualitative than quantitative concept and it will be discussed in terms of various criteria for relevance. It will be dealt with in the sections concerned with the metrics themselves. For the time being we can note (another interpretation of the drunkard’s world view!) that any detail which is visible can possibly be considered relevant.

There are at least four interrelated ways in which we can talk about detail which is important to us, and these will be covered in the next few subsections:

- Detail refers to the spatial scale which we are observing. The smaller the scale the higher the level of detail (§4.1.1).
- Detail can refer to the spatial and temporal frequencies present in the image. The higher the frequency the more detail (§4.1.2).
- Detail can also be taken to refer to the information content of the moving picture (§4.1.3).

---

† I am irresistibly reminded of the joke about a drunk who was found crawling about under a lamp-post by a policeman. Once the drunk had explained that he was looking for his house keys which he had lost the policeman helped in the search. Eventually, because nothing could be seen, the drunk is asked whether he is sure he lost his keys under the lamp-post. ‘‘Oh no’’, says the drunk, ‘‘I lost them somewhere in the dark, but I can only find them by using the light.’’ [I don’t recall the source of this joke]. The mathematical formulation of the priority metric has not gone quite so far off course!

- Detail can also be taken to mean the visibility of contrast in the picture (§4.1.4).

Detail is used to reconstruct the object which produced the image. The more detail the more accurately we can reconstruct the source of the image. Detail can be destroyed by noise added to the image by the various processing steps which the image undergoes from source representation to display (§4.1.5).

4.1.1 Spatial Scale.

Detail can be measured as the apparent size of features in an image. The best way of sampling features of scale σ is by means of a Gaussian with standard deviation σ (§2.4.2). The level of detail present in an image is limited by the width of the sampling filter. We can define the detail level or scale of an image feature to be the smallest σ for which feature still appears in the image when the image is convolved with $G(\mathbf{r}) = \frac{1}{2}\pi\sigma^2 exp(\mathbf{r}/2\sigma)$.

Natural images contain information in nested levels of detail (§2.2.1). There is no single scale which could contain all the relevant information about the scene. If we already have an image we can create other images a lower levels of detail by convolving the image with different, wider, Gaussian kernels. The behaviour of features of such an image is illustrated in figure 4.1 [''scale space filtering'', Witkin 1983 & 1986].

In order to synthesize an image with such definable levels of detail it would be very useful if the the various levels of image detail could be directly computed from the three-dimensional representation.

It could be argued that interesting features of a scene, such as occluding edges, are the result of physical processes and appear on a number of scales independently in the image [''spatial coincidence assumption'', Marr & Hildreth 1980]. When applied to image synthesis we can say that *important* features are those which are spatially localized but which exist even at large scales in the model.

4.1.2 Spatial and Temporal Frequency.

From the discussion of Fourier analysis in §2.3 it is clear that relations in the space and time domain have equivalent relations in the spatial and temporal frequency domains. Which we use is a matter of convenience. Clearly the spatial domain argument presented in the previous subsection can be expressed in the spatial frequency domain as well. Smaller spatial scales correspond to higher spatial frequencies.

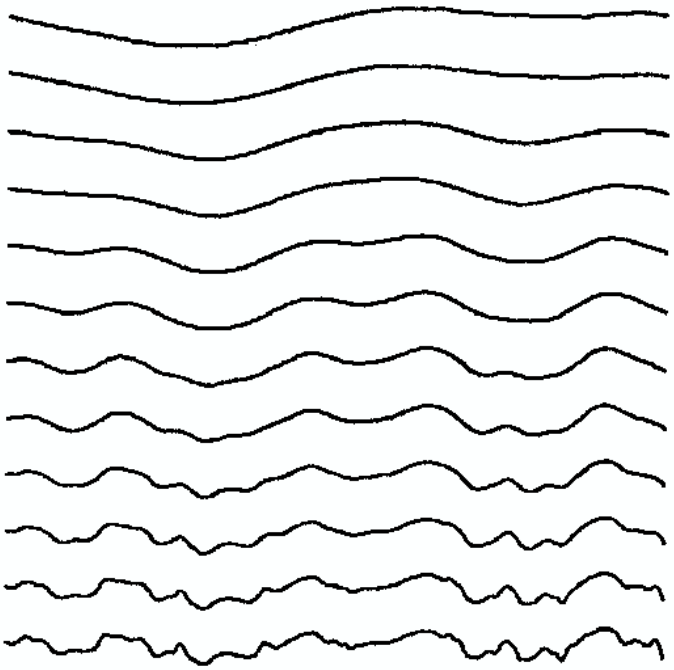


Figure 4.1. Scale Space Filtering. This diagram shows the effect of decreasing the detail resolution (from [Witkin, 1983]).

The frequency domain makes it easier to see the trade-off between spatial and temporal detail. The Fourier analysis of images can be extended to the three-dimensions of 2-D space and time (§2.3.5). Motion shifts higher spatial frequencies more than low spatial frequencies. Large features are therefore more visible at high speeds if the same band-pass filters are subsequently applied (§2.4.7).

4.1.3 Information Content of a Moving Picture.

Information theory was developed in the field of electrical signal transmission by Shannon [Pearson, 1975; Yu, 1976]. It is mainly concerned with communication. The information content of a message depends on how unexpected it was. Messages with a high probability carry little information. The viewer centered metric can be cast as a communication problem as follows:

> The combination of (raster) display and the finite bandwidth of the human visual system constitute a finite capacity communication channel. What is the information capacity of this channel, that is, what is the upper bound on the information which can be transmitted through the channel?

This upper bound on the information which we would want from our three-dimensional computer representation is not much use. For normal viewing conditions the pixel size of the

raster display is just not resolvable. The channel capacity is thus determined by our display resolution and luminance levels. This maximum capacity, in Shannon's terms, would be reached for a completely uncorrelated image. Clearly this is not the upper limit in practice because the human visual system would never be able to decode such an image.

Real images contain multiple redundancies at much larger scales than that of the image resolution, in other words, there is a high correlation between pieces of an image. Because information theory depends on knowing the ensemble of values from which the input signals are drawn this theory cannot easily be applied to images produced from natural scenes [Pearson, 1975]. In order to use Shannon's theory it would be necessary to replace the original image with a new image encoding [Yu, 1976]. It is not clear how such an encoding would be arrived at.

We will therefore not be making further use of Shannon information theory in this dissertation.

4.1.4 Contrast Visibility.

Unless there is some contrast between a feature of interest and its background then that feature will not be visible. Contrast visibility measures an extreme case: we ignore colour and grey levels. It is useful in establishing bounds on visibility. It also has the virtue of being amenable to mathematical analysis [e.g., Gordon & Johnson, 1984].

The observed brightness B_o of an ideal black object at a distance L is solely due to the light scattered into view between the object and the observer (see also §2.2.3). The solid angle of the object at the observer is $d\omega$. Consider a thin layer of air of thickness dx at a distance x. An elementary volume along the view path is then: $dv = x^2 \cdot dx \cdot d\omega$. This volume element contributes a certain scattered luminous intensity $dI = dv \cdot \sigma_\varepsilon \cdot A$. Where A includes the influence of the total illumination and of the relative scattering function of dv, and σ_ε is the extinction coefficient. Actually A could depend on x because of varying illumination or because of the varying reflectance properties of the ground along the path of sight [Gorraiz & Horvath, 1983]. A has values of 1.0 in the sun to 0.4 in the shade, while A at an albedo of ~ 0.9 is $2 \times A$ at an albedo of 0.1 (Soil & plant reflectance ≈ 5% and snow ≈ 90%). Both the scattering function and the extinction coefficient are assumed constant.

The light scattered by the volume element is reduced by a factor depending both on atmospheric extinction ($exp(-\sigma_\varepsilon \cdot x)$) and on the inverse square law ($1/x^2$). Hence the

illuminance contribution of *dv* is

$$dE = A \cdot \sigma_\varepsilon \cdot \frac{e^{-\sigma_\varepsilon x}}{x^2} \cdot dv \tag{4.1}$$

The brightness of the volume element is (by definition) $dB = dE/d\omega$, and from Equation 4.1 this becomes:

$$dB = A \cdot \sigma_\varepsilon \cdot e^{-\sigma_\varepsilon x} \cdot dx \tag{4.2}$$

Integrating Equation 4.2 we get:

$$B_o = \int_0^L A\, \sigma_\varepsilon e^{-\sigma_\varepsilon x}\, dx \tag{4.3}$$

Equation 4.3 gives the brightness of an ideal black target. The brightness of the horizon is defined as $B_h = \lim_{L \to \infty} B_o$. Thus the contrast of of the black object seen against the horizon is:

$$C = \frac{B_o - B_h}{B_h} \tag{4.4}$$

If we assume that A is constant then Equation 4.4 reduces to $C = -exp(-\sigma_\varepsilon L)$. If we further take a typical value for the contrast threshold of the eye in daylight of 0.02 then we can deduce the visibility of the black target to be: $V_b = 3.9/\sigma_\varepsilon$. Horizontal atmospheric visibility ranges from a few metres in fog to ~ 270 km. for a purely molecular atmosphere.

The minimum requirement for visibility of a feature is that the luminance difference with which it is encoded lies just above the viewer's contrast threshold. Therefore if we can show that the contrast of a feature is below threshold it can be eliminated from further consideration. Further use of contrast thresholds is hampered by the fact that no simple relation exists between contrast thresholds and grey level discrimination [Pearson, 1975]. However we can use the form of the above equations for reduction of illumination to extend the usefulness of the spatial priority metric (§4.2.2).

A further extension of the spatial priority metric can be made by combining the atmospheric transfer equations with the equations of the eye (as was done by Kaufman [1981], see §2.5.2). This should be the ultimate aim of the synthesis we are making.

4.1.5 Noise and Detail.

Producing a picture is a process by which information concerning a three-dimensional environment is collapsed into a two-dimensional picture of limited resolution. Naturally information is lost. The viewer reconstructs a representation of the three-dimensional scene from the information available in the moving picture. This reconstruction is complicated by the presence of noise. This noise may have no connection with the scene (e.g. random noise) or it could have had its origin in features of the scene which are being misrepresented (e.g. aliasing effects). On the whole the eye is remarkably tolerant of noise, provided the noise energy is randomly distributed. This is the perceptual basis for stochastic sampling which can be seen as a way of converting the coherent errors of aliasing into random noise [Cook, 1986].

To produce good pictures we need not attempt to exclude all noise, the hazard only really lies with correlated noise. Such correlated noise can result from attempting to render unresolvable detail. We are going to define a detail metric. Its function is to limit extraneous detail. However, as long as the extra detail does not cause unwanted artifacts in the picture (like jagged edges or scintillation of moving objects) it causes no reduction in picture quality, although it probably reduces the efficiency of computer rendering routines.

4.1.6 Conclusion.

We have examined a number of definitions of detail. The most promising for practical implementations seems to be that of spatial scale (§4.1.1), especially if we avoid aliasing effects in the way described in §4.1.5. The Fourier description (§4.1.2) will underpin our theoretical analysis. We shall not be making much use of information theory and the ideas from §4.1.4 on contrast visibility, integrated with an account of human vision, will only be used in extensions to the spatial priority metric.

We shall now derive the spatial priority metric (§4.2) and the temporal priority metric (§4.3). In §4.4 we show how well these metrics are suited to the abstractions of object oriented programming. The last section (§4.5) reviews the results of this chapter.

§4.2 The Spatial Metric.

The *static spatial metric* measures the extent to which extraneous detail can be dispensed with, or equivalently, the extent to which high spatial frequencies will be attenuated. In a sense spatial filtering is applied directly to object representations. In this section we shall see that the static priority of an object is inversely proportional to the width of its spatial frequency spectrum as modified by the various imaging operations (§2.5.2).

For example: the atmospheric modulation transfer function (MTF — §2.3.3.2) is combined with the eye MTF of the supposed viewer [Kaufman, 1981]). Finally, allowance is made for any limitations in the imaging apparatus, such as it being a raster display.

This is therefore more than the usual anti-aliasing in two ways:

(a) anti-aliasing tends to account only for the MTF of the raster display, and

(b) we want to reduce spatial detail in the 3-D representations and not just in the projected image, so as to get the full computational benefit from these approximations.

The priority metric should measure the importance of the object to the viewer and not its size on the image. Geometrically, it is the solid angle subtended at the synthetic camera which is important (§2.1.1). That solid angle, or equivalently projection on the unit sphere, is dependent on the true Euclidian distance and this distance is used to calculate spatial priority. As we have seen in §2.1.1 for small viewing angles there is not much difference between the perpendicular distance to an object from the image plane and the radial distance. We can therefore *approximate* the radial distance by the perpendicular distance in such cases.

The priority dependence on distance can also be explained by considering the spatial frequency domain of the Fourier transform (§2.3.3 & [Bracewell, 1978]). Let $f(\xi,\upsilon)$ be an object and $e(x,y)$ the observed effective image, let the corresponding Fourier transforms be $F(\nu, \omega)$ and $E(u,v)$. According to the previous argument we measure x & y as angular displacements. $x = 2arctan(\xi/2\rho)$ where ξ is one of the spatial coordinates measured in meters and ρ is the distance to the image. If $\rho \gg 1$ then we can approximate x by ξ/ρ. So:

$$e(x, y) = f(\rho x, \rho y) \tag{4.5}$$

where ρ is the distance to the image $f(xi, \upsilon)$ and $\rho \gg 1$. The scaling in space produces an equivalent scaling of the Fourier transform with:

$$E(u,v) = \frac{F\left(\frac{u}{\rho}, \frac{v}{\rho} \right)}{\rho^2} \tag{4.6}$$

This shows how the receding figure has its frequency spectrum spread out over a larger and larger width of frequencies. When low-pass filters subsequently process this image less detail will be passed since detail is concentrated in the shifted higher frequencies.

Apart from the purely geometrical matter of perspective projection image formation is the result of convolution. Many low-pass spatial filtering processes in cascade occur in any image formation system (§2.3). The result of successive convolution is normally to spread a function out: the variance of the convolution of two functions is equal to the sum of the variances of the functions. In fact, as the number of functions convolved together increases indefinitely the result approaches the smooth Gaussian form (Central-limit theorem) provided the transforms are ‘‘humped’’ at the origin.

This generally results in the image being smeared out and losing more and more of its interesting features which are in the high spatial frequencies associated with edges. To demonstrate this blurring would require a grey level display.

The low-pass filters can be (crudely) approximated on black/white displays by choosing not to display features smaller than some threshold. The cutoff being determined by the apparent area of the feature in the image. In frequency terms we cut the image if a characteristic spectral width [Bracewell, 1978] exceeds a limit derived from our priority metric. If we replace the spectrum of our real (even) images with a box which has the same height as the central ordinate of the spectrum and the same volume, then its width is related inversely to the area of the image, if we assume the image is of uniform brightness. That is, the width

of the spectrum is:

$$\frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} E(u,v)\,du\,dv}{E(0,0)} = \frac{e(0,0)}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e(x,y)\,dx\,dy} \qquad (4.7)$$

$$= \frac{1}{image\ area} \quad \text{(if the image is uniform)}$$

4.2.1 The Simple Version of Spatial Priority.

The simplest measure of priority of a feature is to measure its lowest spatial frequency in the image. From the preceding discussion this is proportional to its size and inversely proportional to the distance of the object from the synthetic camera.

The precise meaning of size depends on the extent to which human spatial frequency detectors are orientation specific. If there is averaging over all orientations of spatial frequency then the square root of the area would be a good measure of priority. If the receptors are highly tuned to a specific orientation then using the object's largest dimension is a better measure of size. In practice this has not mattered much since the priority needs only to be a minimum measure of visibility not an accurate one. The priority of an object is determined when its modelling transformation is applied.

The metric is applied by calculating the distance to the object at each frame. In the simple case the object is not displayed if its priority is less than the distance (§5.1.2).

4.2.2 Extensions to the Spatial Priority Metric.

The presence of the atmosphere causes loss of higher spatial frequencies. The effects are also stronger in red wavelengths of light. The effect can be approximated by applying a small exponential weighting to the distance used in the priority calculation (see §4.1.4).

The sharp cut off of features below the resolution limit can cause scintillation in moving objects. On a grey level display these effects can be minimized by using anti-aliasing techniques. The simplest of which is to proceed to sub-pixel resolution and fade the object. This fading of small objects is a form of adaptive supersampling. Supersampling, i.e., increasing in the sampling frequency, is simply a way of increasing the frequency at which aliasing

effects occur (§2.3.4) it does not eliminate aliasing.

The effect of spatial priority cut-off on extended objects is to cause them to be sampled more and more crudely. In this case supersampling only delays the onset of aliasing and does not prevent it. There are probably three ways to avoid aliasing in this case:

- Filter the image of object with a low pass filter before rendering. This approach is useful when the object in question can be represented (implicitly) at different resolution levels.
- Use stochastic sampling [Cook, 1986] to replace the aliasing with random noise, which is much more acceptable visually.
- If neither of these approaches are possible the single priority figure for an object must be replaced by a function which depends on the local curvature. This will then result in adaptive sampling of the surface.

§4.3 The Temporal Metric.

The temporal metric is much more elaborate than the spatial metric, it encompasses several different orders of effect. The simplest view of the temporal metric is the directly analogy with the spatial metric: Objects moving quickly with respect to the observer need to be redrawn more often than those at relative rest. Because two-dimensional motion of the display screen is much simpler than three-dimensional movement in space the next feature which can be measured is the extent to which a particular three-dimensional motion may be approximated by an image transformation. Finally, because of the possible trade-offs between spatial and temporal detail, there is the question about the extent to which spatial detail can be sacrificed in fast moving images.

4.3.1 The Simple Temporal Metric.

The relative speed with which an object is moving with respect to the unit sphere surrounding the observer determines the extent to which its image changes from frame to frame. If there is no movement it need not be rendered again. This is the familiar concept of frame-to-frame coherence. Although this idea is easy to state and the formulation is trivial it is often difficult to apply in practice because of computational difficulties.

When objects in an animation are modelled as actors (in the sense of representing autonomous threads of control) then the temporal priority can be thought of as being the processing priority of the actor computational process. The priority of a real-time computer process is usually unchanging. However, consider a relatively slow moving object. If we don't update its position in one frame then by the time the next frame is rendered the error in its position will have increased. Thus the temporal priority of an actor slowly increases at a rate proportional to its relative speed, until the actor is rendered.

The temporal priority of an object is therefore best formulated as the relative distance moved with respect to the camera since the last update of the image of the object.

4.3.2 Two-Dimensional Approximation to Three-Dimensional Motion.

The generation of images which represent the stages between two configurations of a scene is known as in-betweening. Generally the automatic generation of in-between images requires a full simulation of the three-dimensional geometry of the scene. In only a few cases can we replace the full simulation of three-dimensional motion with changes defined in the two-dimensional image.

Some examples of 2-D transformations which accurately reflect 3-D movements are:

- Rotation of the object (or camera) about the view direction is simply a rotation in the image plane.
- Movement of an object in a great circle about the camera (or camera pan or tilt) results in a simple translation of the image (for small viewing angles).
- Small movement directly towards the camera results in magnification of the image.

The general transformation of an image includes a combination of these effects as well as image shear. There is no simple linear interpolation which can in general transform one projected image into another without doing visible violence to the accuracy with which three-dimensional motion is being rendered [Burtnyk & Wein, 1976].

The optic flow field (§2.2.2) does provide an estimate of the extent to which a three-dimensional motion can be approximated by two-dimensional movement. We shall consider those areas over which the optic flow field is a smooth function of position.

4.3.3 Decomposing the Optic Flow Field.

We shall first establish a few elementary results. The general references were given in §2.2.2, the specific notation used here is largely that of Buxton [1984]. Consider an ideal pin-hole camera. A set of three-dimensional observer coordinates are chosen with the origin at the pin-hole and the z-axis pointing along the direction of view. The re-inverted image plane is fixed at $z = l$ with normal **k** (**k** being a unit vector in the z-axis direction). We shall indicate an object point with the vector $\mathbf{R} = (X, Y, Z)$, and an image point with the vector $\mathbf{r} = (u, v, l)$. (See Figure 4.2). The familiar perspective projection equation is then:

$$\frac{\mathbf{r}}{l} = \frac{\mathbf{R}}{Z} \tag{4.8}$$

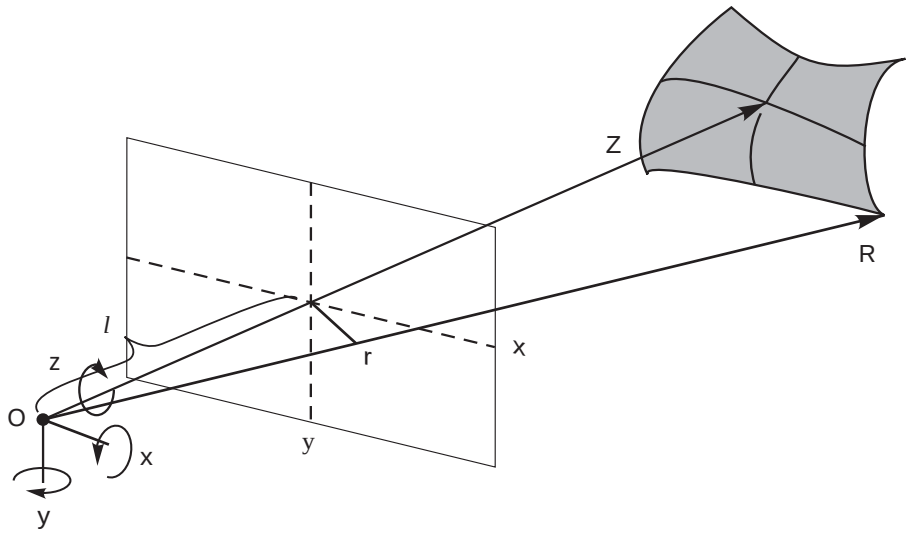


Figure 4.2. Perspective projection of a re-inverted pinhole camera. The centre of projection is at the origin, O. Z is the optic axis. The image plane is on the same side as the object viewed.

If **R** is a function of time then:

$$\frac{d\mathbf{r}}{dt} = \dot{\mathbf{r}} = \frac{l\dot{\mathbf{R}}}{Z} - \frac{l\mathbf{R}\dot{Z}}{Z^2} \tag{4.9}$$

but $\dot{\mathbf{R}} = \boldsymbol{\Omega}\times\mathbf{R} + \mathbf{V}$, where $\boldsymbol{\Omega}$ is the angular velocity of the (rigid) object about the origin and **V** is its velocity due to translation, and $\dot{Z} = \mathbf{k}\cdot\boldsymbol{\Omega}\times\mathbf{R} + \mathbf{k}\cdot\mathbf{V}$ is the projection of $\dot{\mathbf{R}}$ on the Z-axis. Now, eliminating **R** with Equation 4.8, we can write the equation of the optic flow

field:

$$\dot{\mathbf{r}} = \frac{l\,\mathbf{V} - (\mathbf{k}\cdot\mathbf{V})\mathbf{r}}{Z} + \boldsymbol{\Omega}\times\mathbf{r} - [\mathbf{k}\cdot\boldsymbol{\Omega}\times\mathbf{r}]\,\frac{\mathbf{r}}{l} \tag{4.10}$$

One feature of this result is that it can be split into a (first) translational part which depends on Z, the depth, and a rotational part which is independent of depth. Information on 3-D structure is all contained in the translational part.

If we write $\mathbf{r}_0$ for the image of the point towards which we are moving then:

$$\mathbf{r}_0 = \frac{l\,\mathbf{V}}{\mathbf{k}\cdot\mathbf{V}} \tag{4.11}$$

Substituting Equation 4.11 in 4.10 and taking the case of pure translation (i.e., $\boldsymbol{\Omega} = 0$) we get:

$$\dot{\mathbf{r}} = -(\mathbf{r} - \mathbf{r}_0)\frac{\mathbf{k}\cdot\mathbf{V}}{Z} \tag{4.12}$$

If $\mathbf{V}$ is fixed then image points (texture elements) move away from $\mathbf{r}_0$, which is called the *focus of expansion* for that reason. The speed of the texture elements is proportional to their distance on the image from the focus of expansion and to their depth (see Figure 2.3).

The optic flow field $\dot{\mathbf{r}}$ (Equation. 4.10) is a smooth function over $\mathbf{r}$ and t only if $\mathbf{V}$, $\boldsymbol{\Omega}$ and Z are smoothly varying. If we assume we are dealing with freely moving rigid bodies then the flow field will be segmented into smoothly varying patches. The boundaries of these patches occurring where there are depth discontinuities or where one object occludes another, or moves faster than another.

Up to now we have used a general vector notation for optic flow. For the purposes of computation we fix on a rectangular coordinate system, and write $x_n = (x_1, x_2, x_3)$. The vector product $(\mathbf{a}\times\mathbf{b})_i = \varepsilon_{imn}\, a_m\, b_n$, where we use the summation convention that repeated indices are summed over all possible values and where ε_{ijk} is the permutation symbol ($\varepsilon_{ijk} = 1$ for even permutations, -1 for odd permutations, and 0 if any indices are equal). The indices i, j & k range over 1 & 2 only and the other indices, m & n, range over 1, 2 & 3. The Kronecker delta, δ_{ij} is 1 if $i = j$ and 0 if $i \neq j$.

We can then rewrite equation (4.10) as:

$$\dot{\mathbf{r}} = v_i = \frac{l\,V_i}{X_3} - \frac{V_3\,x_i}{X_3} + \varepsilon_{imn}\Omega_m x_n - \frac{x_i}{l}\varepsilon_{3mn}\Omega_m x_n \tag{4.13}$$

In the analysis which follows we shall need the derivatives of the optic flow field. The partial derivatives (i.e., covariant derivatives in a rectangular coordinate system) with respect to x and y of the optic flow field are the following second order tensor:

$$\frac{\partial v_i}{\partial x_j} = \frac{-l\,V_i + V_3\,x_i}{X_3^2}\,\frac{\partial X_3}{\partial x_j} - \varepsilon_{ijm}\,\Omega_m + \frac{x_i}{l}\,\varepsilon_{3jm}\,\Omega_m \tag{4.14}$$
$$- \delta_{ij}\left(\frac{V_3}{X_3} + \frac{\varepsilon_{3mn}\,\Omega_m\,x_n}{l}\right)$$

To derive this equation we have made use of the following results:

$$\frac{\partial}{\partial x_j}\left(\varepsilon_{imn}\,\Omega_m\,x_n\right) = -\varepsilon_{ijm}\Omega_m$$

$$\frac{\partial}{\partial x_j}\left(x_i\varepsilon_{3mn}\Omega_m x_n\right) = -x_i\varepsilon_{3jm}\Omega_m + \delta_{ij}\,\varepsilon_{3mn}\,\Omega_m\,x_n$$

Finally, the second derivative of the flow field w.r.t. the image coordinates is:

$$\frac{\partial^2 v_i}{\partial x_k\,\partial x_j} = \left(\frac{l\,V_i + V_3\,x_i}{X_3^2}\right)\left(\frac{2}{X_3}\,\frac{\partial X_3}{\partial x_j}\,\frac{\partial X_3}{\partial x_k} - \frac{\partial^2 X_3}{\partial x_k\,\partial x_j}\right) \tag{4.15}$$
$$+ \delta_{ij}\left(\frac{V_3}{X_3^2}\,\frac{\partial X_3}{\partial x_k} + \frac{\varepsilon_{3km}\,\Omega_m}{l}\right) + \delta_{ik}\left(\frac{V_3}{X_3^2}\,\frac{\partial X_3}{\partial x_j} + \frac{\varepsilon_{3jm}\,\Omega_m}{l}\right)$$

In this sub-section we have derived the Optic Flow Equation (4.10 or 4.13), with its associated focus of expansion (4.11). For smooth flows we then found the 1st partial derivative (4.14) and the 2nd partial derivative (4.15) with respect to the image coordinates. These equations will be used to define 0, 1st and 2nd order optic flows.

4.3.3.1 First Order Effects of the Optic Flow Field.

The derivatives only apply to the flow field to the extent that the underlying surface is smooth and rigid (although recently some work on bending deformations has appeared: [Koenderink & van Doorn, 1986]). If we apply these equations to a plane moving in three-dimensions we should be able test the applicability of the flow field decomposition to

animation.

If we want to apply optic flow in image synthesis it is useful to express the 1st partial derivatives w.r.t. image coordinates (4.14) as a composition of three separate terms:

A. An Isotropic term — represents expansion/contraction of elementary patches of the flow field.

B. A Traceless Symmetric term — embodies shear.

C. An Antisymmetric term — contains the rotation component.

These terms would then measure the rate at which transformations applied to differential planar facets change. The basic two-dimensional transformations mentioned above (i.e. excluding translation) can be expressed by 2×2 matrices.

The tensor $\frac{\partial x_i}{\partial x_j}$ from equation 4.14 can be written out as a 2×2 matrix:

$$\begin{pmatrix} \frac{\partial \dot{x}}{\partial x} & \frac{\partial \dot{x}}{\partial y} \\ \frac{\partial \dot{y}}{\partial x} & \frac{\partial \dot{y}}{\partial y} \end{pmatrix} = \tag{4.16}$$

$$\begin{pmatrix} \frac{-lV_x + xV_z}{Z^2}\frac{\partial Z}{\partial x} - \frac{V_z}{Z} + \frac{2x}{l}\Omega_y - \frac{y}{l}\Omega_x & \frac{-lV_x + xV_z}{Z^2}\frac{\partial Z}{\partial y} - \frac{x}{l}\Omega_x - \Omega_z \\ \frac{-lV_y + yV_z}{Z^2}\frac{\partial Z}{\partial x} + \frac{y}{l}\Omega_y + \Omega_z & \frac{-lV_y + yV_z}{Z^2}\frac{\partial Z}{\partial y} - \frac{V_z}{Z} + \frac{x}{l}\Omega_y - \frac{2y}{l}\Omega_x \end{pmatrix}$$

These can be written as changes to scaling, shear and rotation by splitting the matrix up as follows:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 2a & b+c \\ b+c & 2d \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & b-c \\ c-b & 0 \end{pmatrix} \tag{4.17}$$

symmetric antisymmetric

$$= \frac{1}{2}\begin{pmatrix} a+d & 0 \\ 0 & a+d \end{pmatrix} + \frac{1}{2}\begin{pmatrix} a-d & b+c \\ b+c & d-a \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & b-c \\ c-b & 0 \end{pmatrix} \tag{4.18}$$

scaling shear spin

In order to apply these transformations we still need to derive the basic image transformation to which these corrections can be applied. This basic transformation depends on the type of surface being rendered (e.g., plane, quadric etc.). These considerations will be left to the actual implementation when such details are known (§7.2.2).

4.3.3.2 Power Series Expansion of the Optic Flow Field.

Any smooth surface patch can be described as a Taylor series expansion in terms of distance in a given direction. If we take the direction to be the line of sight, i.e., the Z-axis, then the distance is the depth Z and we can write [cf. Koenderink, 1986; Waxman & Ullman, 1985]:

$$Z = Z_0 + X \left.\frac{\partial Z}{\partial X}\right|_0 + Y \left.\frac{\partial Z}{\partial Y}\right|_0 + \frac{X^2}{2} \left.\frac{\partial^2 Z}{\partial X^2}\right|_0 + \frac{Y^2}{2} \left.\frac{\partial^2 Z}{\partial Y^2}\right|_0 + X\,Y \left.\frac{\partial^2 Z}{\partial X \partial Y}\right|_0 + \ldots \tag{4.19a}$$

The zero order term (Z_0) determines the position of the patch. The two first order terms specify the orientation (slant and tilt). The second order terms give the curvature of the patch.

Because of the perspective projection equation (eqn 4.8) we can also write:

$$Z = Z_0 + \frac{x\,Z}{l} \left.\frac{\partial Z}{\partial X}\right|_0 + \frac{y\,Z}{l} \left.\frac{\partial Z}{\partial Y}\right|_0 + \frac{x^2 Z^2}{2\,l^2} \left.\frac{\partial^2 Z}{\partial X^2}\right|_0 + \frac{y^2 Z^2}{2\,l^2} \left.\frac{\partial^2 Z}{\partial Y^2}\right|_0 + \frac{x\,y\,Z^2}{l^2} \left.\frac{\partial^2 Z}{\partial X \partial Y}\right|_0 + \ldots \tag{4.19b}$$

If we now substitute the first few terms of Equation 4.19a for all Z in Equation 4.19b, and collect terms in x & y up to a power of two we get:

$$Z \approx Z_0 + \frac{x\,Z_0}{l} \left.\frac{\partial Z}{\partial X}\right|_0 + \frac{y\,Z_0}{l} \left.\frac{\partial Z}{\partial Y}\right|_0 + \frac{x^2 Z_0}{l^2} \left[\left.\left(\frac{\partial Z}{\partial X}\right)^2\right|_0 + \frac{Z_0}{2} \left.\frac{\partial^2 Z}{\partial X^2}\right|_0 \right] + \frac{y^2 Z_0}{l^2} \left[\left.\left(\frac{\partial Z}{\partial Y}\right)^2\right|_0 + \frac{Z_0}{2} \left.\frac{\partial^2 Z}{\partial Y^2}\right|_0 \right] \tag{4.19c}$$

$$+ \frac{x\,y\,Z_0}{l^2}\left(2 \left.\frac{\partial Z}{\partial X}\right|_0 \left.\frac{\partial Z}{\partial Y}\right|_0 + Z_0 \left.\frac{\partial^2 Z}{\partial X \partial Y}\right|_0 \right)$$

However expansion can equally well be given in terms of the image coordinates, so then we have:

$$Z \approx Z_0 + x \left.\frac{\partial Z}{\partial x}\right|_0 + y \left.\frac{\partial Z}{\partial y}\right|_0 + \frac{x^2}{2} \left.\frac{\partial^2 Z}{\partial x^2}\right|_0 + \frac{y^2}{2} \left.\frac{\partial^2 Z}{\partial y^2}\right|_0 + x\,y \left.\frac{\partial^2 Z}{\partial x \partial y}\right|_0 \quad (4.19d)$$

Then equating powers of x & y of Equation 4.19d with those in Equation 4.19c we get the following relations:

$$\begin{aligned}
\frac{l}{Z_0} \left.\frac{\partial Z}{\partial x}\right|_0 &= \left.\frac{\partial Z}{\partial X}\right|_0 \\
\frac{l}{Z_0} \left.\frac{\partial Z}{\partial y}\right|_0 &= \left.\frac{\partial Z}{\partial Y}\right|_0 \\
\frac{l^2}{Z_0} \left.\frac{\partial^2 Z}{\partial x^2}\right|_0 &= 2 \left.\left(\frac{\partial Z}{\partial X} \right)^2\right|_0 + Z_0 \left.\frac{\partial^2 Z}{\partial X^2}\right|_0 \\
\frac{l^2}{Z_0} \left.\frac{\partial^2 Z}{\partial y^2}\right|_0 &= 2 \left.\left(\frac{\partial Z}{\partial Y} \right)^2\right|_0 + Z_0 \left.\frac{\partial^2 Z}{\partial Y^2}\right|_0 \\
\frac{l^2}{Z_0} \left.\frac{\partial^2 Z}{\partial x \partial y}\right|_0 &= 2 \left.\frac{\partial Z}{\partial X}\right|_0 \left.\frac{\partial Z}{\partial Y}\right|_0 + Z_0 \left.\frac{\partial^2 Z}{\partial X \partial Y}\right|_0
\end{aligned} \quad (4.20)$$

These relations allow us to express image transformations in terms of geometrical properties of surfaces. The various Taylor series terms (distance, orientation, curvature, etc.) can now be used to specify local optic flow (or motion parallax) effects — 0, 1st, 2nd, ..., order optic flow effects. This is important in that it expresses the optic flow measure in object coordinates.

The derivatives derived above for the optic flow field (Equations 4.13, 4.14 & 4.15) can now be used for a Taylor series expansion of the optic flow field about a given direction. Moreover, by substituting the relations of Equation 4.20, these can be expressed in terms of the object geometry and motion. For planar objects these relations are not needed because we can use the equation of the plane $(\mathbf{R}-\mathbf{R}_0)\cdot\mathbf{N} = 0$ together with the perspective projection equation (Equation 4.8) to eliminate Z from the optic flow equations [Waxman & Wohn,

1985] — see §7.2.2.

Zero order optic flow is uniform motion in the image plane. The zero order flow has two terms (v_x, v_y) given by Equations 4.10 or 4.13. It accurately accounts for translation parallel to the image plane and rotation about axes in the image plane.

First order optic flow effects are the dilation, spin and shear transformations of Equation 4.16. It is particularly useful for graphics since such affine transformations can be implemented efficiently. If second order effects are small then these transformations account for the optic flow of a surface.

Second order optic flow (Equation 4.15) provides the gradients of the 1st order description. Second order optic flow exactly describes the flow of planar patches. It can be regarded as a truncated Taylor series approximation to the flow produced by a quadric surface patch. For such a patch it has a small but finite radius of convergence.

We have seen that 3-D movement of a surface can be analysed as movement of the features in the corresponding Taylor's expansion as projected onto the image plane. In the case of a planar object the only non-zero terms are the distance (zero order) and orientation of the plane (1st order). Such planar patches do not induce optic flow effects above 2nd order (see §7.2.2).

For more complex surfaces one can either go on to higher order terms or, more practically, build these surfaces with facets which are much simpler. Alternatively we can note that the deformations of first order optic flow (Equation 4.16) apply to small patches and/or short time spans. We can then use the temporal metric to indicate the extent of validity of the approximations. These approximations can then be used for short stretches until the estimated error becomes too large. The full equations for projected 3-D motion can then be applied to correct any errors and a new cycle of approximation can then be initiated.

4.3.4 Influence of the Temporal Metric on Spatial Detail.

We have already seen from Fourier analysis that motion can shift high spatial frequencies (§2.3.5) to the extent that they become invisible §2.4.7). We have also mentioned that there seem to be two kinds of processes operating in the perception of visual apparent motion, one at high frame rates and another at lower frame rates, but that there is no unanimity about this amongst researchers in the field (§2.4.6).

The short range process is said to operate at high frame rates. It is also a truism of animation that realistic motion can compensate for crudely drawn pictures, indicating perhaps that the higher level cognitive process is also at work here. Unfortunately such effects are hard to quantify. This is a case in which experimental determination of parameters might be more useful than mathematical analysis.

A simple experiment along these lines could be comparing the effects a quick rendering method for fast moving objects and slower, but more detailed, rendering method for slower moving objects. One important property of such a fast rendering method has to be that it does not cause small details to disappear altogether. The sudden appearance and disappearance of even very small detail is very obvious, indicating that low resolution objects should be low-pass filtered.

4.3.5 Conclusion.

The spatial metric is used to reduce the levels of detail required from a representation to the minimum required for conviction. It depends primarily on distance, but it can also be weighted to allow for atmospheric effects and human visual characteristics.

The temporal metric measures the ways motion influences the relative visual importance of objects to the camera. The temporal metric aims at:

1) a reduction of the rate at which moving images have to be updated,

2) a reduction in the extent to which moving 2-D images have to be recomputed from the underlying 3-D representation.

3) a reduction in the spatial detail of fast moving images.

Only the relative motion of an object with respect to the observer influences temporal priority. This is another expression of our general principle of viewer-centredness. Thus it is more profitable to speak, not of the motion of the camera, but rather of the optic flow field of the environment about the camera.

The optic flow field can be split naturally into a hierarchy of effects, 0, 1st, 2nd, etc., order flow. Using these various orders of optic flow we can approximate 3-D projected motion with 2-D image transformations in a predictable fashion. The magnitude of the neglected terms provide an estimate of the error of neglecting them and the higher order terms. In

Chapter 7 this concept will be more fully developed. The temporal metric will then be seen to be a measure of frame-to-frame coherence (§7.2.2). The temporal metric is not just a single number — it is hierarchy of terms.

§4.4 Data Types.

The definitions of the various detail metrics have now been derived. In this section we will state a few general principles which govern the application of the metrics to object models. We shall first examine how variable detail representations would interact with the metric (§4.4.1) and then present the general type of abstract interface such data structures will require.

4.4.1 The Prototype of a Variable Detail Data Structure: Fractals.

Fractals can be generated with increasing detail. This results in a process which is a reversal of the way scale space filtering is done. Fractals of increasing detail can be synthesized adding higher and high spatial frequencies to a picture (see Figure 4.3). Because of the way they are generated we can also guarantee that high frequencies are absent. Thus low-pass filtering is inherent.

Unfortunately the Fourier synthesis method requires that all the levels of detail be precomputed. Increasing detail is cheaper with other methods which are less accurate approximations to true fractals. The ideal data representation would be recursive. That would mean that at any particular spatial patch the amount of detail could be increased to the required level for the spatial priority of that location. As we have seen (§4.2.4) the generation of extra detail could stop once the typical dimensions of the patch were below the spatial priority indicated by the metric.

4.4.2 Data Types Required for Adaptive Detail Animation.

Three basic classes of object are required to produce an animated picture with adaptive detail:

- A Camera which can render shapes in perspective on the display.
- Actors which contain the basic motion and geometric information.
- An Appearance type which mediates between the underlying model and the Camera in

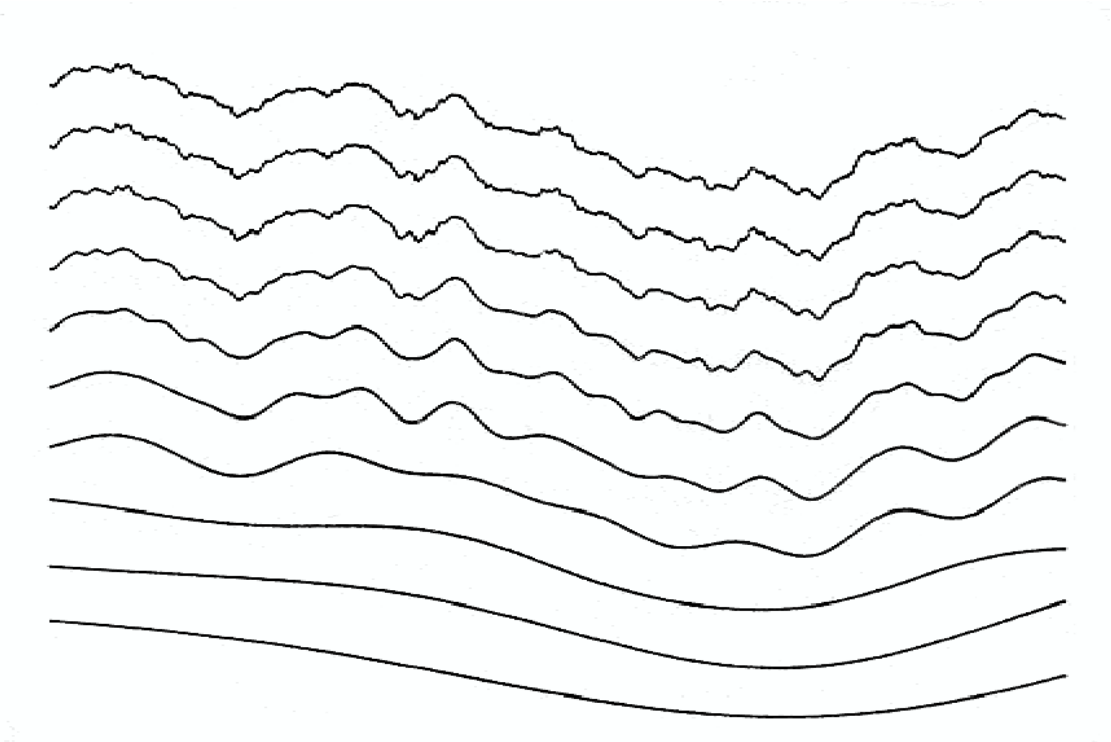


Figure 4.3. Increased detail by further generation of a fractal. Compare with the figure 4.1. (From [Voss, 1985]).

order to produce an adaptive rendering depending on the priority metrics.

Naturally many more classes would be involved, but these are the ones which directly participate in the application of the spatial and temporal metric. Of these types the Appearance plays a central role. It is a *part* of the Actor in the sense used in Chapter 3. However it is part which can be handed out to the Camera as a message parameter. If we did not have such a type then the Camera would have to interact with all the different Actor types by means of a standard primitive representation.

In the case of the static metric the Appearance is handed to the Camera once the Actor's position in the coordinate space of the Camera is known. The Appearance can incorporate information about the way the Actor looks which is tuned to the specific circumstances, including of course the spatial priority. An Actor can have many varieties of Appearance, depending on how it is to be rendered, e.g. wireframe or shaded.

In the dynamic case the Appearance is handed to the Camera once an Actor has performed all updating for the particular time step which has to be done whether the Actor is displayed or not. The Appearance thus signals completion of processing and can be used to ensure that

Actors have terminated before rendering proceeds. The Appearance will maintain a link with the Actor in this case since the Actor may need to know whether it has in fact been rendered or not.

The Appearances embody the spatial and temporal detail metrics. They are used to specify salient features about the Actor relative to the Camera before the Actor is actually rendered.

This discussion has served to give a qualitative outline of the specifications of the abstract data types needed in adaptive detail rendering. These ideas are further worked out in Chapters 5, 6 and 7. Finally in the conclusion, §8.1.3, the progression of ideas culminates in a general methodology. As a postscript Appendix D contains some proposals for using these abstractions in concurrent animation.

§4.5 Conclusion: Practical Priority Metrics for Animation.

The spatial and temporal metric measure the amount of relevant detail in an object (or Actor). The definition of detail in terms of spatial frequencies is favoured for analysis, and this allows us to discuss the various stages of processing filters or convolutions of the spectrum of an object.

The practical implementation of the metrics does not depend on this kind of Fourier analysis. Instead we derive simple measures and constructive definitions. The spatial metric is a weighted distance to the object, where distance can be either the radial distance from the viewpoint or the perpendicular distance from the image point. The spatial metric is *always* weighted, the weighting allows for Camera viewing angle, display characteristics, and can be extended to include atmospheric effects, etc.

The temporal metric is more complex. In its simplest form it can also be expressed as a single measure: radial speed of the object with respect to the synthetic camera. Once again the option exists to take the velocity as projected on the unit sphere (true radial velocity) or as projected on the image plane.

More complex forms of the temporal metric depend on optic flow decomposition. In the 1st order case we get a 2×2 transformation matrix which measure the rate of change of the two-dimensional image transformation. This measure is quite general: it applies to all surfaces, at least to some approximation, in exactly the same sense as a normal Taylor

expansion. However, its application depends on the kind of surface being modelled (§7.2.2).

To encapsulate the various ways in which the priority metrics interact with objects we define the interaction in terms of an abstract data type: the Appearance. The Appearance mediates between an Actor (or object) and the synthetic Camera. The Actors can have any underlying data representation and the Camera a number of different rendering options. The Appearance calculates the various priority measures based on the relative states of the Actor and Camera. The correct data can then be extracted and presented to the chosen rendering mechanism.

Summary of the Metrics.

1) Spatial priority metric:

   - Limits detail via a single measure depending on distance.
   - Extended to allow for atmospheric and perceptual MTF.

2) Temporal priority metric:

   - The measure for adaptive processing and updating is the relative distance moved since the last update, or, the order of optic flow transformation which has to be done.
   - In order to replace 3-D motion with 2-D transformations it measures various orders of optic flow effects (0, 1, 2, ...).

3) Trade-offs between spatial and temporal detail:
   E.g. less detail needed in fast, blurred objects.

4) Abstract Data Types:

   - The various actors and synthetic cameras are implemented as classes with a fixed interaction protocol.
   - Interaction between Actors and Cameras is mediated by an Appearance. This abstract data type encapsulates the priority metrics.

This concludes our formulation of the spatial and temporal priority metrics. The experimental tests of the metrics will now be presented.

Chapter V

The First Experiment:
Using the Static Metric on a Model with a Non-Uniform Hierarchy.

A stick figure, which has a non-uniform hierarchical structure, was chosen for the first implementation. This prototype was implemented in Smalltalk which had been extended to include a part-whole (assembly-subassembly) hierarchy. This experiment provided a test of some key aspects of the static (spatial) priority metric formulated in the previous chapter. The resulting costs and benefits have been analysed (§5.3).

This experiment also provided an opportunity for critically examining the object oriented programming paradigm as exemplified by Smalltalk, some results have already been discussed in Chapter 3.

§5.1 Discrete Detail and a Continuous Metric.

As a first test of the operation of the metric a simple implementation of a stick figure was used. Such a stick figure has a non-uniform hierarchy of parts making up wholes. The levels represented discrete unequal changes in detail. The device chosen for displaying the animated figures in this first implementation was a simple black/white display without grey levels.

Stick figures were chosen because they are easily drawn on a display which lacks grey-levels. They are more challenging than many other hierarchical models with respect to the priority metric because the levels of detail are not equally spaced nor are the figures infinitely divisible (as fractal representations are).

The intention is to demonstrate the adaptive detail display, and not primarily to create realistic pictures. The low-pass filters are crudely approximated by choosing not to display features smaller than some threshold. The cutoff was determined by the apparent area of the feature in the image. In frequency terms, we cut the image if a characteristic spectral width exceeds a limit derived from our priority metric. The relation between frequency and the apparent size of spatial detail has already been investigated theoretically in the previous chapter (§4.2).

The reader is no doubt aware that simply cutting out a feature when it reaches the resolution limit could cause distracting ‘‘boiling’’ or scintillation effects in an animated sequence where a part of a figure moves continually in and out of resolution. This can be avoided for

such figures by choosing part priorities so that a feature only disappears when it already has sub-pixel size. In this case the metric should in fact reduce scintillation because for sub-pixel sized objects some sampling schemes record alternate hits and misses on a minute object as it moves. By using the priority metric these occasional hits are prevented. Since the target display (a standard Smalltalk display) does not allow shading this is probably the best we can do. In §5.4.1 some consideration is given to the case where a shaded display is available.

5.1.1 The Hierarchical Representation of a Stick Figure.

When viewed at the coarsest resolution or from far away, the human body can be represented by a single upright cylinder [Marr & Nishihara, 1978, Clark, 1976]. Thus previously also Leonardo da Vinci [MS E 80v, (~1513) in MacCurdy, 1954, p349.]:

> Thus with a horse, it would lose the legs sooner than the head because the legs are thinner than the head, and it would lose the neck before the trunk for the same reason. It follows therefore that the part of the horse which the eye will be able last to discern will be the trunk, retaining still its oval form, but rather approximating to the shape of a cylinder, and it will lose its thickness sooner than its length ...

Making the shape a sharp edged cylinder is a further simplification which neglects the loss of high spatial frequencies which cause the ‘‘cylinder’’ to be rounded. At the next level of resolution, when the body is being viewed from somewhat closer by, we have a collection of attached cylinders representing the torso, head, arms and legs. At still closer approach, the hands and feet, fingers, toes, and features of the face would become visible. So to represent the body at each of these distances, successively more complex models are needed. These models can be organized in a hierarchical list (Figure 5.1).

We need not go through all features of an object to identify and represent each possible level of detail. This means, for example, that we need not necessarily organize the features of the face into a separate hierarchy. The features could be incorporated into one of the higher levels, such as the head as a whole, and we could depend on the rendering mechanism to ‘‘filter’’ out the extraneous detail.

Using an object oriented approach to modelling has the advantage that the parts which an object possesses need not actually be represented as such in the object (§3.6.3). The internal implementation of the object is hidden behind the object’s interface protocol. The only requirement is that the object respond correctly to messages which address the part. This allows great freedom for dealing with special cases. A particular object is free to possess

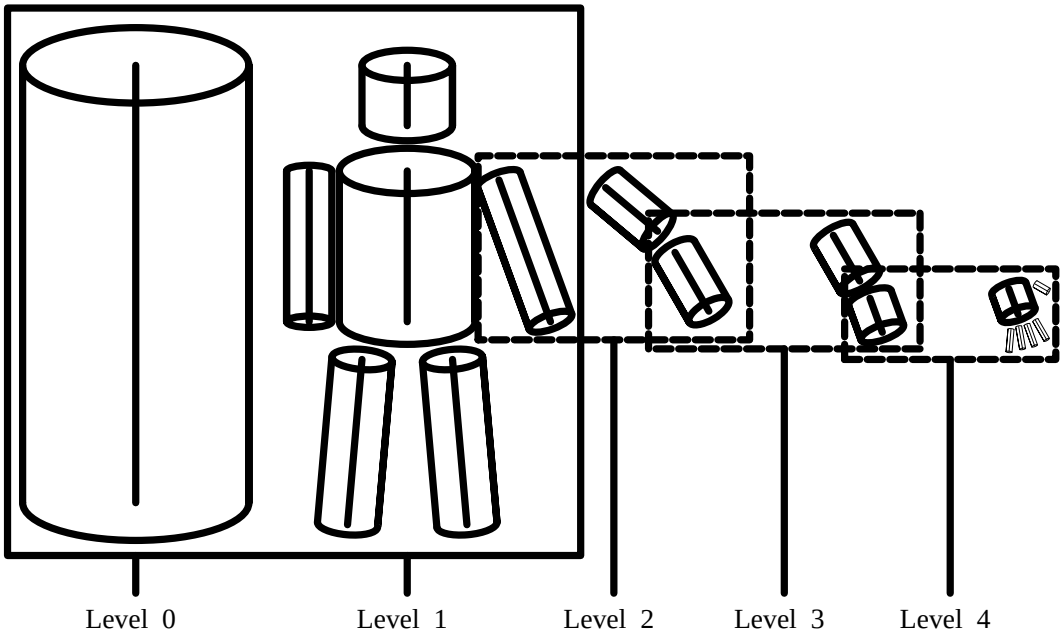


Figure 5.1. The hierarchy of levels of detail in representing a human figure. Level 0 is an unstructured ‘‘blob’’, while level 4, which is represented on the diagram only by an elaboration of the hand, is much more detailed (based on [Marr & Nishihara, 1978]).

phantom appendages which represent anomalous detail structures that appear occasionally.

Up to now we have been talking of a hierarchy of levels of detail. When an object is modelled we get a different kind of hierarchy: the *modelling hierarchy*. A modelling hierarchy records the way complex objects are built up out of simpler parts. When animating natural figures consisting of rigid limbs connected by joints it is usual to model a figure as a tree of dependent parts. This is in general not possible for artificial objects where mechanical feedback loops necessitate more complicated structures, but such objects do not concern us in this study.

It is vital too to note that the modelling hierarchy for a stick figure is very similar to the detail hierarchy. The human body can be modelled as a tree of parts, the root of which is the torso and the parts are the chest, head, upper arm, forearm, hands, thigh, lower leg, foot, etc (Figure 5.2). While this hierarchy is not the same as the visibility list, it is closely related in that parts at lower levels of the tree are also generally smaller.

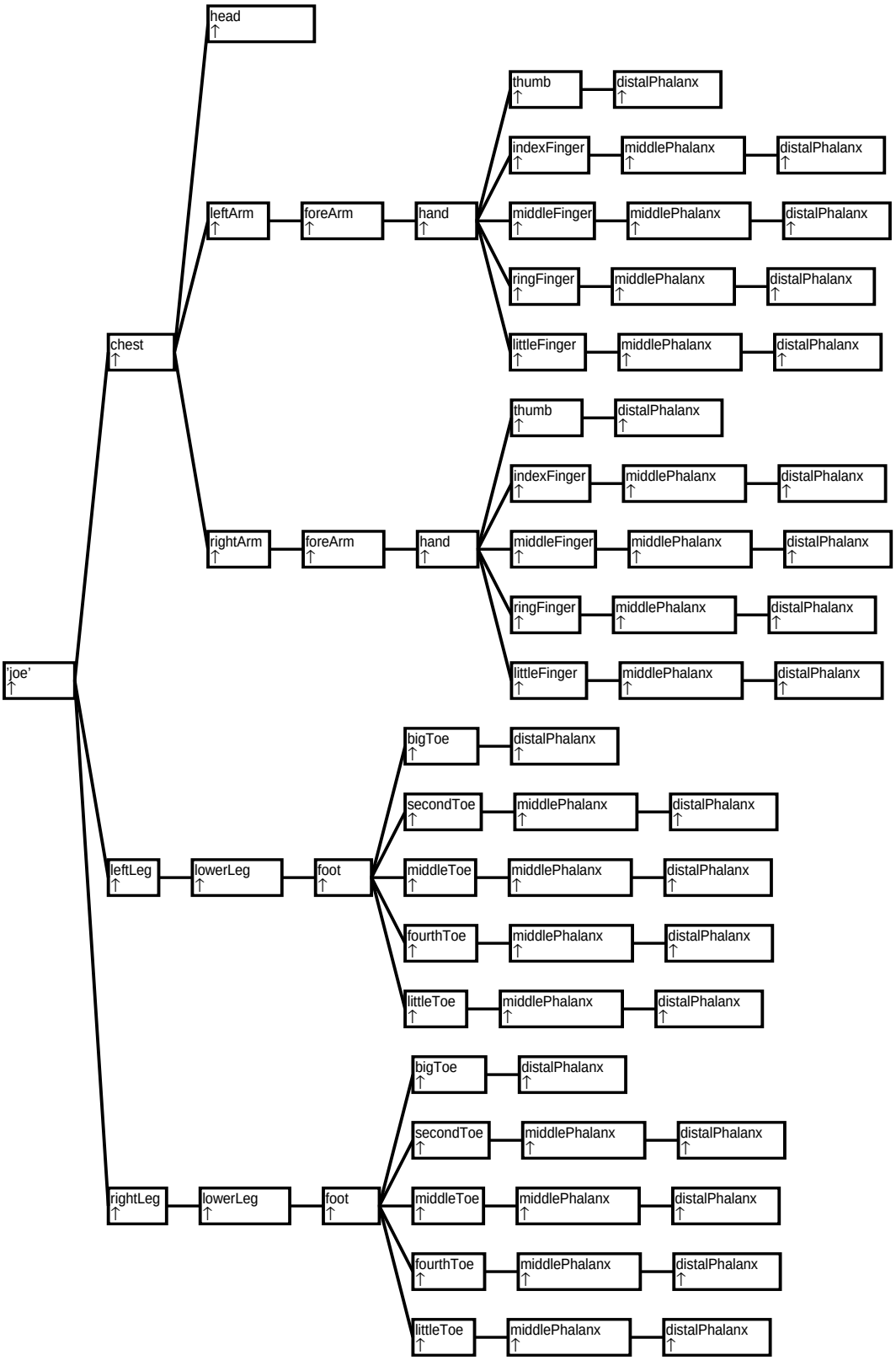


Figure 5.2. The part hierarchy for the stick figure. The boxes contain the name of the part and below it the class to which the part belongs. This is a *component* hierarchy, but it is very similar to the *detail* hierarchy in Figure 5.1.

This result applies to large class of living things. They can often be represented by generalized cylinder figures with extremities of ever decreasing size. Marr & Nishihara [1978] contains illustrations of a pipe-cleaner rabbit, giraffe, ostrich and so forth. It can be conjectured that being able to describe the figures in this way is a combination of ontogenesis (artifacts can always have big bits tacked on) and of size (insects do not seem to have such a simple relation between distance from the ‘‘torso’’ and limb size).

5.1.2 Using the Priority Metric with a Stick Figure.

We shall now examine the way the priority of an object is actually calculated and the way in which the calculated priority interacts with the hierarchical representation in order to vary the detail of the figure to be displayed.

The detail hierarchy deviates from the object hierarchy in the treatment of parts which lie at the lowest level of detail which is still visible. Parts thus have two guises: They are *internal nodes* when there are other, smaller, parts which can still be resolved. When a part is itself the smallest resolvable object in its sub-tree then it is said to assume its *leaf appearance*. Generally, and by default, the leaf appearance is the same as the internal node appearance, but it can differ. This differing leaf appearance is to allow for the effects of blurred nodes lower in the hierarchy, but it can only accurately be used when there is only one dependent node (unary sub-tree, as with the elbow joint or finger joints), but this is also where it is most needed.

The priority (P) of a part is the square-root of the maximal cross-sectional *area* which its ‘‘leaf’’ appearance can present. In practice the ‘‘leaf’’ appearance is empirically determined and is mostly the same as the internal node appearance. In frequency terms we are assuming that the frequency spectrum is circularly symmetric and its width can be represented by a single number (see also §4.2.1).

The precise definition of the priority is not that important at this stage provided it can consistently be applied and provided it ensures that sufficient detail is provided to the camera. Thus for stick figures a simple measure which depends only on the maximum dimension (the length) of a part could also be used.

Each part has a range (R), which is the maximum distance by which its extremes or the ends of its parts can move while it rotates in one place. This is essentially a spherical bound on its position. More precisely: if the sub-parts are at distances x_i from the proximal joint of

the part in question and if the ranges of the sub-parts are r_i where i is the index of the sub-part, then the range R of a part is the maximum of x_i+r_i or the length of the part itself if that is longer. The range is independent of the higher levels of the hierarchy of a part (which it need not know about in any case).

Distance (d) is measured from the camera and weighted for viewing angle (or magnification m) and display resolution. The rendering algorithm then only descends the hierarchical model up to the level where (object priority + range) becomes less than the magnification*distance.

In symbols:

$$\text{if } P + R > md \textbf{ then } \text{display part.}$$

For any particular object the extent to which its complexity drops off with distance depends on its structure. For the human figure used in the experiment the choice of priorities, determined by means of areas as outlined above, resulted in the number of parts dropping off more or less with the square root of the distance, for those distances over which the figure was visible.

§5.2 Implementing the System of Priorities.

The advantages of using Smalltalk for the initial implementation is discussed in Chapter 3 (§3.3.1). In order to include an object hierarchy and still retain the spirit of modularity, object-oriented languages should be extended to support part-whole relationships (see §3.4). As far as modelling an animated figure the above mentioned sections may be summarized as showing that once object oriented languages, with their simulation (Simula) pedigree, are extended to include a part-whole hierarchy they become ideal choices for animating natural environments.

Coordinate system transformations are needed to relate the systems of localized coordinates of the stick figure's limbs (§3.2.3). These manipulations were effected by an implementation of Hamilton's quaternions (§3.3.3).

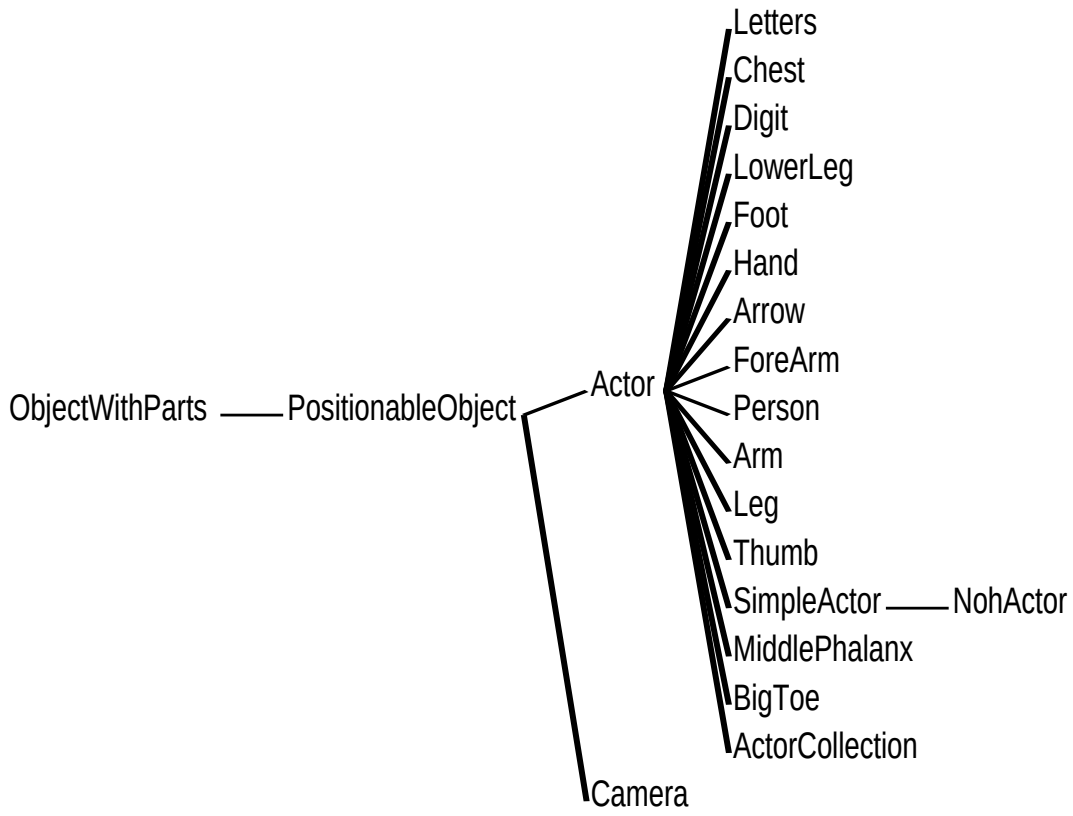


Figure 5.3. The subclass hierarchy of the class ObjectWithParts. It contains the classes used to model the stick figure. Contrast this with the part hierarchy of Figure 5.2.

5.2.1 Splitting the Problem into an Hierarchy of Classes.

The knowledge required to implement moving figures is factored over a hierarchy of classes. As usual the top level class in our sub-hierarchy for creating actors is the most general, abstract, and simplest. This is the class ‘‘ObjectWithParts’’ (Figure 5.3); it confers the ability on objects to consist of parts. Subclasses of ObjectWithParts will inherit its message protocol, which is the set of messages it can understand, along with any methods implemented for executing the messages. Processing is carried on by instances of classes, but no instances of this class are created since it is rather abstract.

The next subclass, ‘‘PositionableObject’’, which is also an abstract class, provides the six degrees of freedom of objects in space: position and orientation. The required transformations are implemented by instances of the class ‘‘RevoluteJoint’’ which in turn uses the class ‘‘Quaternion’’. Any subclass of PositionableObject will be able to implement a modelling hierarchy consisting of subassemblies and transformations.

Instances of the subclass ‘‘Camera’’ are used to render other actors. The class ‘‘Actor’’ confers the ability for an object to have a visible appearance but is itself too general to be concrete. The appearance of an Actor depends on the class of object in its appearance ‘slot’ or instance variable. For a stick figure this is an instance of the class ‘‘Stick’’, but it could be a more complicated appearance. Instances of Stick know how to interact with an instance of Camera for rendering and this all that would be required from more complicated appearances.

The subclasses of Actor are the actual parts of a figure. Because a ‘‘SimpleActor’’ is a terminal node in the object hierarchy it has no parts and its inherited capability for parts is disabled.

5.2.2 Putting Together a Figure.

Having defined the classes which make up the figure we still have to instantiate it. In general with object oriented programming this is a fairly simple operation: the class to which an object belongs (which is another object in its own right) is requested to return a new instance of the object. Most non-trivial objects also require a simple method for initializing the instance. But stick figures are rather complex assemblies of objects and their initialization would involve using numerous *ad hoc* routines for ensuring that all parts are present and correct.

For complicated objects composed of parts, numerous advantages accrue from using a formalism which provides language support to the notion of assembling complex objects from simpler parts. The classes defined above come ready equipped to collect the necessary parts and slot them into itself correctly. This is because all actors (parts) inherit from the abstract superclass ‘‘ObjectWithParts’’ the following features:

- Objects are assembled from parts.
- Parts are instances in their own right and can belong to any concrete subclass of ObjectWithParts.
- To instantiate a whole object the classes of the parts must be known. Defaults are stored as a dictionary of parts and classes.
- On becoming part of a whole an object ceases to be independent. It belongs to the whole’s hidden local state and the interface is mediated by its owner.

Thus when defining a new kind of Actor the system ensures that a default class is specified for each part. Then when a new instance of a whole is requested the system will have compiled code which can automatically instantiate an instance of the whole with all its parts.

But actors are not merely assemblies of parts, they live in three dimensional space and their limbs have lengths and orientations. These lengths have to be specified for each kind of figure. This knowledge is typically not stored at the level of the part but rather in its owning whole. Each part by default is of unit length and it overrides the lengths of its sub-parts. Thus, for example, the hand specifies the (relative) lengths of the fingers, while the body as a whole specifies the size of the head, arms and hands, and so forth.

Thus the system can instantiate whole objects with very little user effort. What it cannot do is provide for the proper physical placing of parts which are kinds of ‘‘PositionableObject’’. In general the programmer still has to define methods for setting the relative positions of limbs.

5.2.3. Movement.

Each part of the hierarchical representation may have its own changing coordinate system. In rendering these coordinate systems have to be related to one another by a traversal of the whole object. Essentially the task of the animator is to specify a visually convincing series of transformations of positions. That this is a large and complex topic, which is in many ways an art rather than a science has, already been mentioned in Chapter 3.

The purpose of this first experiment discussed in this chapter is to show that we have provided the basic, essential, foundation of an animation system and that the spatial metric can successfully interact with such a system. On the basis of an examination of the literature (see §3.1.6) we can perhaps make the stronger claim that such an actor based system makes a *very good* foundation for an animation system.

In this experiment we are content to have figures and cameras moving steadily in straight lines or circular arcs by direct manipulation of coordinate transformations. After all, we are trying to show the variation in levels of detail only.

The parts of the figure are each defined in its own local coordinate system, which oriented along the principal axis of the limb. The rendering algorithm of the camera starts by getting the names of the actors from the World and recursively descending the part hierarchy of each

actor. The coordinate transformations are concatenated with the inverse coordinate transformation of the camera.

Thus a change in position of a part affects all parts lower in the hierarchy. Traversal for rendering stops when the limits of resolution determined by the priority metric is reached. Figure movement is therefore effected by altering the coordinate transformation of a joint (see Figure 5.4) after which the figure is rendered again on the display.

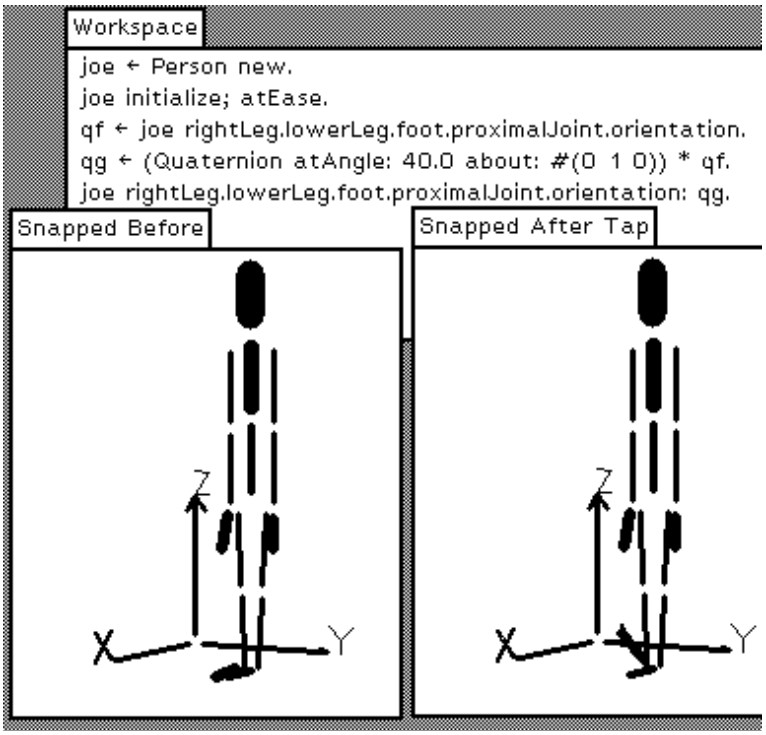


Figure 5.4. Accessing the part-whole hierarchy with a compound message. The new instance of Person is assigned to the variable ‘joe’. Joe is asked to forward messages to its right foot. The request is to replace the orientation Quaternion by one which has been rotated by 40° about a vector along the y-axis. Notice that the familiar multiplication message ‘‘*’’ is also understood by Quaternions.

The compound message syntax, though elegant from a modelling point of view is rather unwieldy for the user’s point of view. Abbreviations of messages by defining single word abbreviations for the more common compound selectors is an obvious first step. For example all subclasses of PositionableObject understand the message ‘orientation’ which is simply an abbreviation of ‘proximalJoint.orientation’ as used in the illustration.

A somewhat better user interface could include a system of direct manipulation where the user selects and positions an object on the display by using some kind of pointing device

(this is routinely done in Smalltalk based systems, see for example ThingLab [Borning, 1979]). The complicating factor here is that we have the normal depth ambiguity of a 2-D perspective projection of a 3-D scene. This ambiguity affects both selection and the specification of movement direction.

The figures have been modelled as a hierarchical collection of parts. These parts each possess a local coordinate frame which can be altered to show movement when the figure is rendered. Thus the basic requirements for an animation system exists and there is good reason to believe that it can be extended. The point of the experiment was to see how the spatial priority metric interacts with such an object.

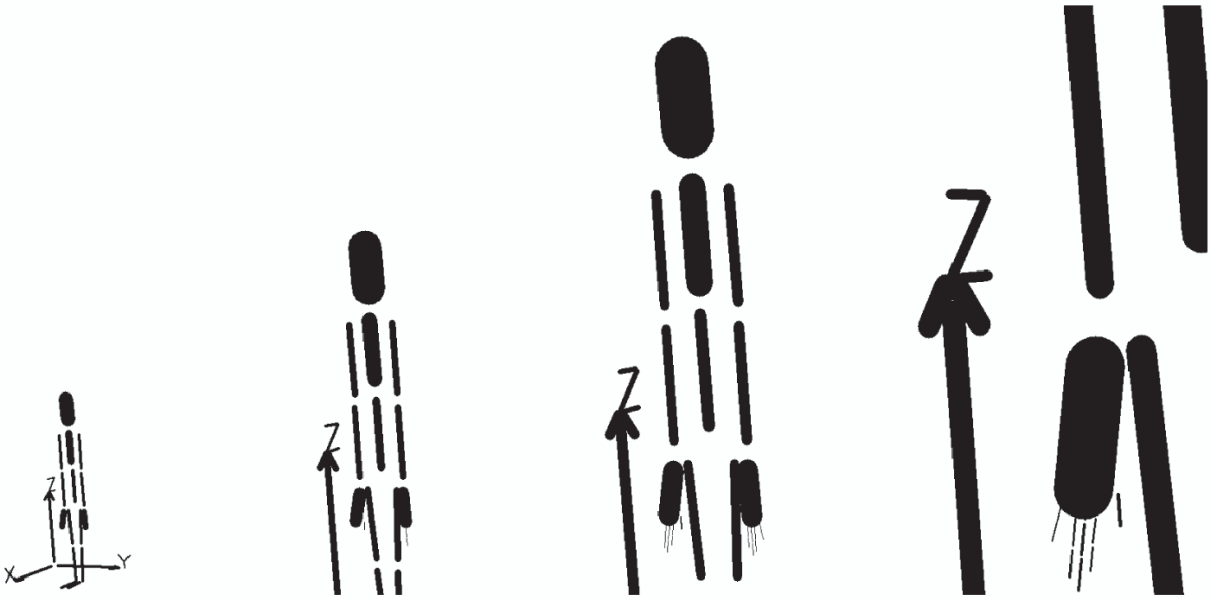


Figure 5.5. Stills from a sequence of the camera flying in to the right hand. The effects of the spatial detail metric have been exaggerated to show how new levels of detail are added: in this case the spindly fingers. The little finger appears last.

§5.3 Results from Running the Experimental Implementation.

The effects of the spatial priority metric are illustrated by the sequence of stick figures (Figure 5.5). The camera is flying in to the right hand. The effects of the priority metric have been exaggerated in order make its operation visible. The persistence of a feature can be seen to depend on both its length and its width. For the human figure used in the experiment the choice of priorities resulted in the number of parts dropping off more or less with the square root of the distance, for those distances over which the figure was visible.

Coordinate system hierarchies were used; each part of the figure being described in its own local coordinate system. Such a part and all its dependent parts is moved by changing its coordinate transformation. The hierarchy has a considerably richer structure than octree hierarchies, and motion was easily modelled.

We greatly simplified spatial frequency effects and used a simple rendering algorithm on a bit-mapped display. The scenes are animated but dynamic effects on the visual importance of detail were neglected. That is, only a spatial metric was applied.

5.3.1 Computational Benefits.

The depth complexity of natural scenes can be very large: on average an ‘X-ray’ from the viewer to the horizon will pierce many more potentially visible surfaces than the two or three surfaces in artificial scenes [Sutherland et al. 1974]. The terms normally used to analyse rendering complexity, such as number of faces in the environment and average face height, do not apply because we have tried to capture the nested complexity of nature in the hierarchical models.

We shall assume in this analysis that the environment is isotropic, that is, objects are uniformly distributed on the two-dimensional surface of the Earth with density K. Therefore the depth complexity of views which are not too vertical is independent of viewing direction.

Consider those objects in the simulated environment which lie beyond the furthest limit of resolution, they need only be considered as unstructured ‘‘blobs’’. The range, R, will ensure that this condition is satisfied for all internal configurations of the objects. Then for average R and priority, P, the average limit of resolution is $l = R + P$ (see §5.1.2). The area contained within the resolution limit is proportional to l^2 and so only $K(P + R)^2$ non trivial objects need be examined in any greater detail. Rendering complexity is decoupled from the

total number of objects and depends on the density of objects, *K*.

Within the average resolution limit the priority metric attempts to remove all sub-pixel sized parts from consideration. The object therefore cannot consist of many more parts than the number of pixels it covers. The complexity of the rendered scene depends on the resolution of the display which can now be adjusted to suit viewing conditions. We reduce the number of (parts of) objects to be examined in *clipping* and *rendering*, two very substantial costs when realism is sought.

The *principal computational benefit* is to limit the extent to which the complexity of the rendering problem depends on the complexity of the environment. Detail in an object depends on its distance from the synthetic camera, weighted for screen resolution etc.

The *second substantial benefit* is that a uniform metric allows different forms of hierarchical models to be integrated in an animated environment. It is an essential component in providing a uniform interface between the synthetic cameras and object representations.

5.3.2 Costs.

The major cost, apart from setting up the hierarchy, is associated with calculating the metric. Instead of just the distance along the axis of projection, the Euclidian distance to the part must be found, and the priority comparison made. However, the three squares and square root calculation is far less expensive than any but the simplest of rendering options.

We do not give relative timings here because our rendering was the simplest imaginable. The complexity arguments given above are convincing enough to indicate the promise of the approach. More detailed considerations of the costs and benefits of this approach will be found in the next chapter (Chapter 6). The rendering problem considered there is much more realistic.

If the cost of calculating the square root is considered excessive then both sides of the priority comparison, $P + R > md$, can be squared. Or the Euclidian distance metric could be simplified, but this was not considered worth exploring in this demonstration.

The cost of setting up the hierarchy has two aspects: the amount of effort required by the modeller and the effort required from the machine. In animation the effort from the machine is insignificant because the benefits will be accrued frame after frame.

We are then left with the effort required from the modeller. The resolution hierarchy can readily be superimposed on the modelling hierarchy which is already required. Therefore the additional effort by the modeller is small. The only exception arises when parts on the modelling hierarchy have only one dependant with no decrease in size. The two levels must then be essentially one object as far as the display hierarchy is concerned. For example, the arm is modelled as two parts which drop in and out of visibility together.

The cost during the running of the system is the calculation of the distance and the check against priority. This is small compared to the cost of 3-D shaded rendering. Setting up the detail hierarchy for many living things is not much more complicated than producing the normal object hierarchy which is required in any case.

§5.4 Conclusion.

By means of the experimental implementation discussed in this chapter we have investigated methods to use in building representations for animating complex environments such as those found in nature. An approach to modelling was presented and implemented which:

- can cope with the rich detail of natural figures.
- is computationally efficient.
- is appropriate to modelling animated objects.

The implementation showed that a metric can be used with stick figures on a bit mapped display. Computational costs seemed minimal on the basis of complexity arguments and the benefits grow according to the complexity of the environments.

We have so far investigated the benefits in terms of the reduction in rendering time. The approach is more general than that. In general the priority metric can be used to modify the computational cost of any property of the model which can be bundled or batched as the object becomes less important to the scene. This can apply also to movement modelling. For example: individual birds merge into a flock where we need not model wing movement and eventually the birds coalesce into a single body.

Chapter VI

The Second Experiment:
The Spatial Metric Applied to Continuous Detail Levels.

The usefulness of a spatial priority metric is investigated further by applying it to a landscape model. A fractal representation is chosen. This has, properly speaking, self similar detail at all levels of resolution. This model is approximately rendered by an algorithm which combines a number of recent approaches to the rendering of land surfaces with the static detail metric. The metric is used to limit both the space and the time requirements of the computation. The resulting algorithm is recursive with each level of recursion increasing the resolution of the picture.

§6.1 Introduction: Digital Representation of Landscapes.

It has already been pointed out that nature presents us with a startling array of nested levels of detail. This profusion of detail can be bewildering to the builder of a digital model of a physical landscape, or rather it would have been if we had attempted the task a couple of decades ago. Since then Mandelbrot's fractal geometry [Mandelbrot, 1982b] has provided a satisfying and compact mathematical description of many complex natural phenomena, including landscapes.

Fractals are mathematically ideal shapes, albeit rather chaotic compared to the traditional geometrical ideals. They are governed by rather precise statistical correlations between the various levels of detail. The properties of fractals were discussed in §1.3.4 and §2.2. To render these mathematical descriptions on the display of a computer we have to approximate them. We may even choose to simplify the statistical properties, but thereby run the risk of introducing artifacts into the pleasing natural appearance of the pictures [Mandelbrot, 1982a & 1983]. The algorithm adopted (§6.3) is based on a previously adopted compromise between mathematical rigour and computational efficiency. Methods of computing fractals and approaches to rendering landscapes are reviewed in §6.2.

The earth's surface can be represented by a (single valued) height function over the area being modelled. For such surfaces the processes of clipping, hidden surface removal and shading can be greatly simplified compared to the requirements for general three-dimensional shapes. A feature of our approach is that the fractal texture can be overlaid to provide additional texture to an existing less detailed model of the landscape defined over a

grid of data points. This is an essential requirement for creating realistic landscapes which fit in with a story. A number of cameras can range simultaneously over the landscape and facilities are provided for setting these cameras up interactively.

The point of this experiment to analyse the applicability of the spatial priority metric (§6.4). Adding a spatial priority metric greatly reduced the time to render the landscape satisfactorily for the most common viewing conditions. The application of the metric also reduced the number of intermediate results which had to be stored.

§6.2 Synthesizing Pictures of Landscapes.

In this section we provide a brief recap of recognized ways in which models of landscapes can be built and rendered with a computer.

Producing a computer model of a landscape is really a question capturing the endless detail in an efficient way. The fractal model of a landscape stores features as a procedure for generating them. The procedure is governed by very few parameters: typically a parameter which governs roughness, or perhaps a couple of parameters governing the way roughness depends on absolute height. Alternatively there are digital maps of the earth's surface which specify explicitly the height for all points of interest. In computer animation it is usually desirable to be able to specify the general shape of the landscape and allow the computer to fill in the details.

Although fractals require few parameters to initialize them they do require storage for previous and intermediate results. This is because it is an essential feature of fractals that there is correlation between points at all distance scales. This requirement for some sort of global state make it difficult to implement fractal generation recursively.

Having produced our landscape model we still have to render it on a display. The assumption that landscapes are represented by single valued height functions of position allows clipping and hidden surface removal to be greatly simplified.

6.2.1 Approximating Fractals on a Computer.

Fractals were introduced in Chapter 2 as an excellent description of rough natural landscapes. There exist a number of ways fractals can be approximated for rendition on a computer display. That they *have* to be approximated is evident from the fact that a

mathematical fractal contains self similar detail at *all* spatial frequencies. The very least that is needed is to limit the small levels of detail, in principle this is done through low pass filtering at the Nyquist limit of the display.

Fractals require the calculation of an stochastic function, $B_H(t)$. Methods of generating fractals include [Voss, 1985, Fournier et al., 1982, Haruyama & Barsky, 1984, Brelstaff, 1984, Miller, 1986, Mastin et al., 1987]:

1) *Sum of independent random pulses*. This method is analogous to the way in which Brownian motion is the sum independent random jumps. It is computationally very expensive. Termination conditions depend largely on visual acceptability so precise terms are difficult to estimate. Each step addition affects half the points being generated on average. Voss quotes 10000 steps for models which appear to contain some 1000000 points, thus the number of operations was $n^{5/3}$ in that particular case.

2) *Fast Fourier transform filtering.* This method allows the synthesis of random functions with any desired spectral density. Finding the FFT of n points takes O(n log n) operations. However all points have to be calculated at the same time to the same resolution which makes it unsuitable for our purposes.

3) *Random midpoint displacement.* This method is very quick, O(n) although extensions by a single point can take O(log n) operations. Unfortunately the fractals approximated by this method do not have the required correlations between points and this produces visible artificial faults which do not disappear with added detail. The recursive algorithm developed in the next section is based on this method, it is further discussed below.

4) *Successive random additions.* Like the midpoint displacement method, this method, due to Voss [1985], also only requires O(n) operations, though with a larger constant term. This method has the advantage that it can be made to approach the accuracy of the FFT method, but it cannot easily be implemented recursively because determining the position of a point requires information to be communicated from other, distant, points.

5) *Weierstrass-Mandelbrot fractal function.* This function is the infinite sum of geometrically changing levels of detail. Exponentially decreasing Gaussian random terms modulate sine waves of exponentially increasing frequencies. The phase of the sine

wave is uniformly random [Berry & Lewis, 1980]. Methods 3 and 4 are in fact special cases of this method, and like method 4 it requires too much global information to be easily implemented recursively.

Random Midpoint Displacement. [Fournier, Fussell & Carpenter, 1982]

This method proceeds recursively with each stage increasing the level of detail represented. Consider starting with a sample of N_i points at stage i and smallest detail of size δ_i. The next stage $i+1$ with detail of size $\delta_{i+1} = \delta_i/2$ requires $N_{i+1} = N_i \times 2$ points. Of these only N_i are new. They are generated by interpolating between the points in the previous stage. To the N_i new points of the stage $i+1$ a Gaussian random variation of zero mean and variance ${\delta_{i+1}}^2$ is added. H is the parameter which governs the roughness of the resulting surface, and $0 < H < 1$. For our height fields the fractional dimension $D = 3–H$.

With the random midpoint displacement method however we do not add a random element to every point at every stage, once a point is determined it remains fixed. When the final stage n is reached with its N_n points each will have had only one random addition. The calculation therefore requires N_n calculations and additions of random variables.

6.2.2 Hidden Surface Removal with Height Fields.

To render an object we have to display those surfaces which are visible from the chosen viewpoint and give them the correct shade for the existing light conditions. To decide which surfaces are visible is in general a sorting problem which is well know to be an O(n log n) problem for n elements [Sutherland et al., 1974]. However, excluding the odd cave and beetling cliff, landscapes are essentially single sheets with the height $h = f_h(x, y)$. Provided the height field can be made single valued, we can circumvent the need for sorting the facets of the landscape. Instead, the facets can be enumerated in a predetermined order which can ensure the closer facets are always presented after the further ones (or vice versa), this order is called an *occlusion compatible* ordering.

We can make a trivial, but quite general, observation about visibility on the unit sphere (Figure 6.1) which surrounds the viewer: only those objects which lie along the same ray from the centre can fall on the same point on the sphere. Therefore only such objects can occlude one another.

A landscape reduces the spherical symmetry to that of a cylinder, whether or not the observer is on the ground. Call the point on the earth's surface directly below the observer

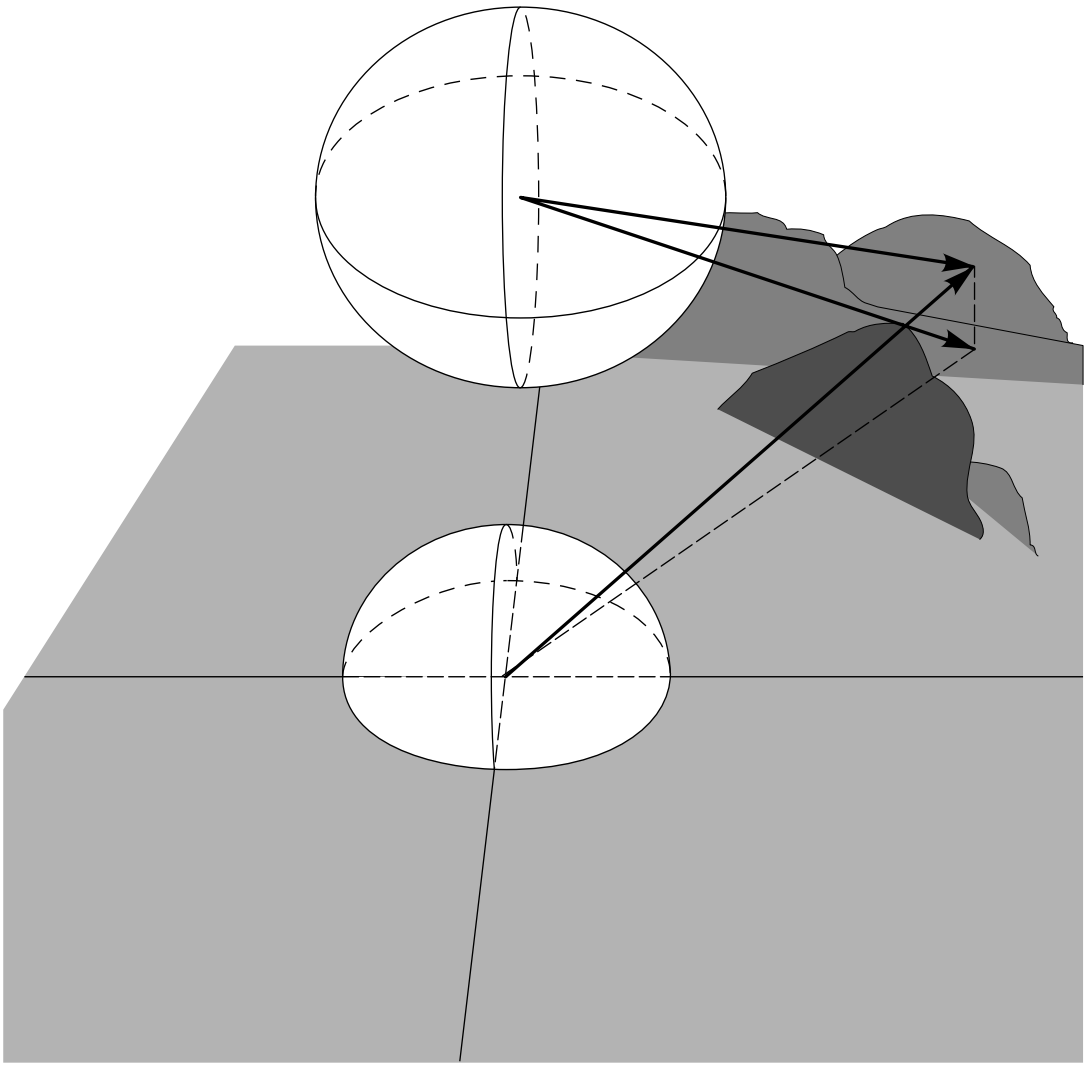


Figure 6.1. Visibility relations on the unit sphere. Those objects which lie along the same ray from the centre fall on the same point on the sphere, and only such objects can occlude one another. Relative height cannot alter this fact.

the station point. Then we can measure direction as the two-dimensional geographical direction about that point. Any position in space can then be characterized by a direction (θ), a distance along that direction (r) and a vertical height h. It is clear that only objects lying in the same direction (θ) from the observer can obscure one another. If one object (A) obscures another (B) then it must also be closer in the horizontal direction ($r_A < r_B$), height cannot influence this fact.

The representation of the station point in the picture is the vanishing point for all vertical lines. Radial directions in world coordinates about the station point thus project into the same straight lines as vertical displacements from those directions. This simply expresses

the same relation given above for world coordinates in terms of image coordinates.

If the earth's surface is traversed for rendering in such a way that for every direction we first visit the furthest points along that direction then we can simply render the surface on the display as it is encountered. The closer opaque areas will obscure the more distant points — the painter's algorithm.

When applied to height fields the painter's algorithm can involve a lot of unnecessary rendering of surfaces which are eventually obscured. This happens particularly when the landscape is viewed in an almost horizontal direction: the most common view direction for animals which can't fly. Fortunately the spatial priority metric is most efficient at limiting detail in such cases.

''Obscured features of the landscape lie below the horizon'' is another, equivalent way of expressing the relations derived above. This leads to another hidden surface algorithm which can be used with height fields: the floating horizon method. For this algorithm the height field is traversed from the nearest to the most distant points. A representation of the current horizon is maintained. New pieces of the landscape are rendered if they are above the present horizon in that horizontal direction. These calculations can be performed in screen coordinates by making use of the vertical vanishing point [Anderson, 1982]. This method requires more housekeeping but it allows anti-aliasing of facet edges.

6.2.3 Clipping Height Fields.

Coquillart & Gangnet [1984] described a way of precomputing which area of the height field is definitely invisible, which might be visible, and which is certainly visible. Again this can be done without needing to calculate the height fields in detail. All that is required is maximum or minimum values which the heights can reach.

We shall describe the way the whole area which is potentially visible can be deduced, this is the only one which we need. The way of finding which areas are certainly visible is analogous. Let C be a convex polygon on the plane of $z = 0$ (zero height). Initially this polygon will be the domain over which the height field is defined, typically a rectangle.

The synthetic camera is a rectangular window on the scene. With the viewpoint at the apex this forms the traditional viewing pyramid. There are thus at least four intersecting planes which limit the field of view: top, bottom, left and right. Objects on the viewer's side of the

window can also be excluded: the front clipping plane. (A distant clipping plane can also be defined but it was not used in this case.

Let P be such a plane, it has an equation (in world coordinates) of a $ax + by + cz + d = 0$, or in vector notation $\mathbf{n} \cdot \mathbf{x} - d = 0$ where $\mathbf{n}$ is the normal to the plane. If the normal is chosen to point into the visible area then all those points for which $\mathbf{n} \cdot \mathbf{x} - d < 0$ will certainly be invisible. It thus remains to deduce from this plane the domain of the height field which can only produce invisible points.

If the height field was uniformly at its maximum height (z_{max}) then the plane P would intersect it along the line $ax + by + cz_{max} = 0$. The half plane H^- where $ax + by + cz_{max} < 0$ would then be the invisible domain. If the height field were at its minimum value we would similarly substitute z_{min}. The real height function lies between these values and so the real intersection will lie between them as well. It then only remains to determine which of the half planes is the most conservative estimate. This depends on whether the normal to the plane P is horizontal or vertical or whether it has a net positive of negative vertical component. The details can be found in the cited paper.

One consequence which we will use is that if plane P has an upward normal ($c > 0$) then the maximum height is the limiting factor. For most views only the bottom clipping plane will have such a normal. If it were otherwise we would be staring at the sky with our synthetic camera. Such a bottom plane cannot intersect the surface at a point closer than the view point whatever happens, and that intersection will happen if the maximum height reached is the height of the viewer. This fact is used with fractal surfaces in the next section where the extrema are not exactly determinable.

6.2.4 Quadtrees, Quadtries and Quadcodes.

Quadtrees [Hunter & Steiglitz, 1979a; 1979b; Samet, 1984] are well known methods for recursively subdividing square images into quadrants. They can of course be used for any function defined over a two-dimensional domain, including height fields. A *quadtree* is a generalization of an extended binary tree with nodes which have either 4 children or none. A node with no children is a *leaf*. The leaves will represent points on the height field.

We are interested in surfaces defined by single valued functions over a finite, square, two-dimensional domain. The function is evaluated or approximated at finite points corresponding to the corners of a quadrant. A particular quadrant can be characterized by the length of

its sides, which we can represent as some integer power of 2 in any convenient unit of measurement.

Quadcodes are a way of addressing and representing the recursively subdivided quadrants. The order of this subdivision is not circular, it is left to right and top to bottom: north-west to north-east to south- west to south-east (Figure 6.2).

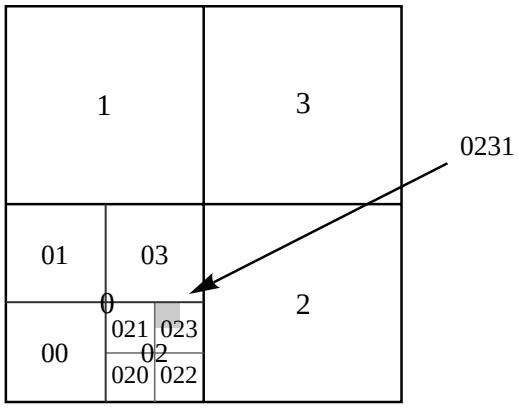


Figure 6.2. Recursive subdivision of a square into quadrants. The quadrants are labelled with their quadcodes.

A base four (quaternary) number can be assigned to each quadrant in the order given above and on Figure 6.2. As the quadrants are subdivided the further digits can be appended on the right (least significant position). The resulting code uniquely specifies the position and depth of the subdivision, and constitute a quadcode. Some properties of quadcodes were recently documented by Li & Loew, [1987a; 1987b], Samet refers to them as *locational codes*.

We write a quadcode of length *n* as:

$$Q = q_1 q_2 \cdots q_n \tag{6.1}$$

where $q_i \in \{0,1,2,3\}$ for $i = 1,2,...,n$.

When a sequence of quaternary digits are used as keys in this way to access data the structure is known as a *quadtrie* [‘‘information re*trie*val’’ Fredkin, 1960].

The resolution (R) of the function approximation is then the length of the quadrant at whose corners we evaluate it. The size of the domain (M) is also a power of 2.

$$R = 2^l \leq \mathrm{M} = 2^m \tag{6.2}$$

where l & m are integers, $l \leq m$.

The corners of the domain M are addressed by quadcodes of length 1. The corners of the quadrants at resolution R are addressed by quadcodes of length $m - l + 1$.

There is a simple relationship between fully enumerated quadcodes for some square area and the corresponding matrix with the same resolution. If we index the matrix with binary row index i and binary column index j then the quadcode takes its odd bits from i and its even bits from j.

We can write:

$$q_1 q_2 \cdots q_n = i_1 j_1 i_2 j_2 \ldots i_n j_n \tag{6.3}$$

We are concerned with enumerating point data in a *preorder traversal* of an implicit quadtree. We visit first the root, then node 0, then node 1, then node 2 and finally node 3. We can call such a traversal *monotonic* if we never decrease the depth of recursion as we proceed. As we proceed with the recursion an area is gradually traced out by the leaves, this area must obviously form a connected region. The boundary of this region can never be longer than the border of the original quadrant which is being subdivided.

This can easily shown inductively by considering the subdivision process for a quadrant. For each of the nodes 0,1,2,3 the boundary length is obviously less than or equal to the quadrant border length.

The point data we are concerned with is stored at the corners of a quadrant. During the recursive subdivision process some points are shared by disjoint subtrees. These are *perimeter* points, other points are called *interior* points. Consider the first subdivision step in subdividing a quadrant: clearly the centre point of the quadrant which is shared by all subquadrants can be directly passed to them without having to appear on a perimeter. The corners of the quadrant will have come from a higher level and need not concern further, except to pass it on to the lower levels. Exactly 4 points remain: the midpoints of the sides.

We assume that the quadrant has been assigned a quadcode. We can then label the midpoints with either the odd or even bits of the the quadcode depending on whether the midpoints are on a horizontal or a vertical edge. Provided we distinguish between outer and inner edges

we then have a compact and unique labeling for our perimeter points. This method is recursive. Taking as our base case the root quadrant, whose corners are given as input, it is clear that the following result follows by induction over the recursion depth:

Theorem 6.1: The perimeter of a preorder recursive subdivision of a quadrants can be stored in four binary trees with the same depth as the quadrant subdivision.

The four binary trees are called *horizons*, (by analogy with the floating horizon method). They are called the inner and outer, horizontal and vertical, horizons. The codes to access the elements on the horizons depend on the quadcode of the quadrant being subdivided.

The maximum number of values which have to be stored in a horizon depends on the depth of the quadrant subdivision. Starting with $l = 0$ we store 2^l values at each level of the binary tree.

Corollary 6.2: Each horizon contains $\sum_l 2^l$ values $= 2^n - 1$ where $l = 0, 1, ...n$.

Since n is the $\log_2$ of the number of subdivisions of an edge the number of values which have to be stored in an horizon is of the same order as the resolution along an edge in *object* space.

§6.3 Rendering Landscapes Efficiently on a Raster Display.

A number of threads in recent research in rendering landscapes (discussed in the previous section) can be combined with the spatial priority metric to reduce the computational complexity of the problem. These are:

- Using a predeterminable occlusion compatible ordering for traversing a landscape for rendering. This makes hidden surface removal very simple, though it may in fact increase the overall computation time. It was used because sophisticated hidden surface removal is not relevant to the purpose of this experiment. Backfacing facets were eliminated before rendering them.

- Clipping the domain of the height function to the maximal region which could be visible without reference to the actual function values. This is a simple technique to reduce the area over which the fractal has to be calculated. It was used both with and without the priority metric.

- Approximating self similar fractals by the method of random midpoint displacement. This technique produces adequate results when we interpolate between the points of the predefined grid data. The grid data is read from a file which was in fact generated by the Fourier method but could equally be real altitude data.

- Use of quadcodes and our result derived above for the subdivision perimeter to limit the storage requirements for a recursive subdivision method *only when combined* with the spatial detail metric.

We make use of these and add the spatial priority measure to produce a recursive algorithm (§6.3.1). The algorithm is implemented in an object oriented language: C++. This allows a clear separation of the camera and associated priority metric from the representation of the modelled landscape ((§6.3.2). See Figure 6.9 for a snapshot of the rendering algorithm in action.

6.3.1 Recursive Subdivision Algorithm for Synthesizing Landscapes.

The algorithm has five stages:

1) The polygonal domain of potentially visible points in the height field is determined.

2) The occlusion compatible ordering for recursively traversing the domain is determined.

3) Recursive subdivision commences and proceeds to the level determined by the *priority metric*.

4) Intermediate results are stored for later reuse where necessary.

5) The smallest resolvable square facets are passed to the shader in an order which is compatible with their being rendered immediately on the screen. Normals to the facets can be found without any multiplication.

Stage 1. Clipping.

The algorithm for stage one has been described in §6.2.3. The only potential problem lies in the fact that the maximum and minimum heights which are the random result of the random additions cannot be exactly predetermined. The expected variance of the overall process could be used. Or we can simply say that all heights below a certain level will be suppressed

(‘‘filled with water’’) which fixes the minimum height. As discussed above, using the viewer’s height instead of the maximum height will produce a correct result for all view directions that are met in practice. The actual height of the landscape underneath the viewer can be calculated and used as the minimal height of the viewer.

We require from this stage a function which can say if a point on the height field in world coordinates (x, y) is potentially visible or not. We also need the maximum and minimum values of the bounding polygon in the x and y directions, which gives us a bounding box. This clipping allows us to reject points as being invisible without having to find their position relative to the viewer (i.e. without projecting them).

Stage 2. Ordering.

Let the station point (the point on the world’s surface vertically below the view point) be the origin of a viewer centered coordinate system in the plane $z = 0$. The plane is divided into four quadrants (Figure 6.2) labelled 0 to 3, but where 0 is the south-west, 1 the north-west, 2 the south-east and 3 the north-east quadrants. As discussed in section §6.2.2 we want to deal with the points which lie along any given direction in a predetermined order. Let us say furthest first, that is, from the outside inwards (see Figure 6.3). Similar arguments would apply if we chose the opposite, outwards, order.

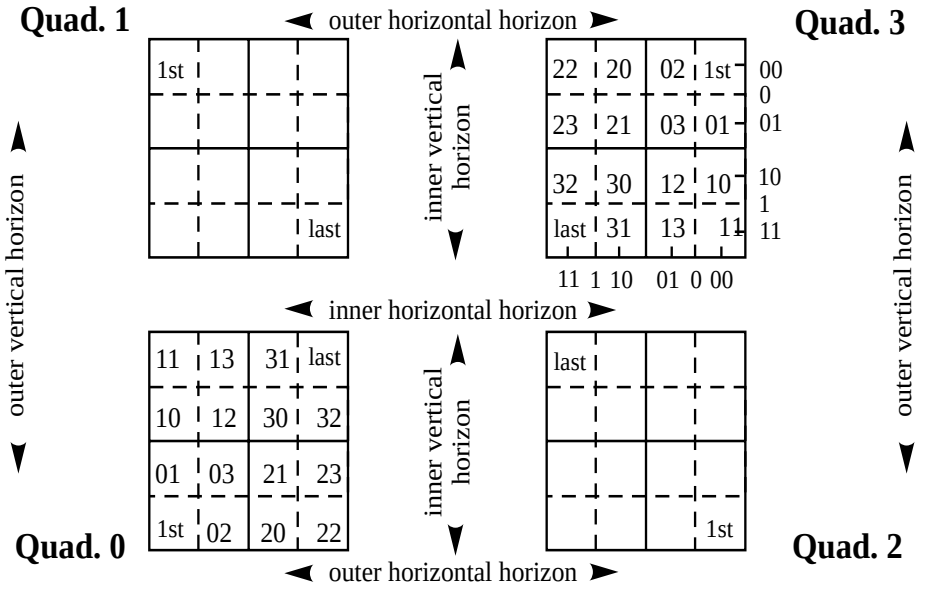


Figure 6.3. Occlusion compatible ordering with quadcodes. The figure indicates the ordering for progression from the outside inwards. The *horizons* are also indicated.

Not all the quadrants will be visible for a given viewpoint. We can determine which will be visible by classifying the station point with respect to the clipping polygon's bounding box.

We intend recursively subdividing the quadrants and want to know what order to proceed in. We number subdivided quadrants in the same order as the major quadrants. An occlusion compatible inwards order for quadrant *n* is then:

first: subquadrant *n*

second: subquadrant $1 \hat{} n$

third: subquadrant $2 \hat{} n$

fourth: subquadrant $\neg n \,\&\, 3$

Where $\neg$ denotes bitwise negation (ones complement), & is bitwise and, and $\hat{}$ is bitwise exclusive or. It is clear that the order presented here corresponds to the quadcode preorder traversal examined in §6.2.4.

Stage 3. Recursion.

For each potentially visible quadrant we choose the subquadrants in the order specified in the previous stage. The quadrants are recursively subdivided in a depth first fashion. According to the clipping polygon quadrants are either completely within the polygon of potentially visible points, partially inside it, or completely outside it. Subquadrants, which we can now call facets, are discarded when they are certainly invisible. The other facets are continually subdivided until the current resolution limit as determined by the priority metric is reached. Subfacets of facets which have been found to be completely within the clipping polygon need not be tested further against the clipping polygon.

The priority metric is based on the distance from the viewpoint to the facet corner closest to the origin (it is numbered $\neg n \,\&\, 3$ according to the scheme presented above). The distance is either the true Euclidean distance or the distance projected onto the camera axis pointing into the scene (i.e. the distance used for perspective projection). The choice is determined for each camera individually. The true distance is left as a square and is compared to the area of the grid underneath a facet. The projected distance is compared with the average dimensions of the grid.

The real Euclidean distance is more expensive to calculate. The reasons for using this measure (which is equivalent to projection on the unit sphere) have already been discussed (§2.1.1 and §4.2). Any weighting of the distance for atmospheric effects can only accurately be applied to the true distance, although for small viewing angles the two measures are virtually the same.

Once the resolution limit is reached the facet can be passed to the camera for shading and rendering. Thus recursion terminates either when the resolution limit is reached or when it is found that a facet is completely outside the visible area.

Stage 4. Storing Intermediate Results.

As the recursive subdivision proceeds the recursion reaches greater and greater depths, since we are proceeding from more distant points to closer points which need to be examined in more detail according to our spatial detail metric. This process need not be monotonic but since we are interested in maximal storage requirements we will deal with the maximum depth reached at any stage.

A disadvantage of recursive subdivision methods is that state information cannot easily be communicated form one subtree to another. This information is needed to ensure consistency between adjacent points, to avoid unnecessary recalculation and to communicate information on the gradients of adjacent facets for smooth shading purposes. However the spatial metric can be used to limit the storage required for these intermediate results.

From corollary 6.2 we have an upper limit on the perimeter storage required which depends on the depth to which the current quadrant is being subdivided. This is normally the root quadrant. However we can prune form our tree any quadrants for which we have completely enumerated the zeroth, first and second subquadrants. The effective quadrant becomes then subquadrant 3. In terms of quadcodes we can ignore a leading sequence of threes.

Because of the spatial priority metric the deepest recursion levels will in general be reached only on the closest subquadrants.

Stage 5. Rendering.

Hidden surface removal is particularly simple. Nothing need be done! The facets don’t arrive in exactly the depth order or furthest first. However they do arrive in such a way that any facet which can obscure another arrives after that other one. Before this last stage

proceeds a final clipping test in screen coordinates determines whether the projected facet does fall on the screen.

We can assume that the surface of the world is always viewed from above. This fact, together with the knowledge of which facet corner is furthest and which nearest to the viewer, means that back facing facets can be eliminated by a two-dimensional scalar product in integer screen coordinates. This test is inexpensive.

The facets are defined on a grid. The average of the normals to the four corners determines the facet normal. This average can be found without resorting to any multiplications. The shade of a facet is calculated by the cosine law. The scalar product of the facet normal and the light direction is found and normalized.

Scan conversion is done by a polygon fill operation with the selected shade. The facets are small enough to obviate the need for interpolating shades across the polygon. Although this can be done since information about adjoining normals can be communicated via the perimeter horizons.

It is also possible to divide the facets into a far and near triangle *a priori* without having to calculate distances. The same rendering algorithms apply except that the normal for shading purposes is now unique and need not be averaged.

6.3.2 An Object Oriented Implementation.

The various parts of the problem are divided into a number of classes. In addition some classes interface with the workstation display and interaction routines. The program as a whole was designed for interactive use (see Figure 6.4). The user can specify various parameters of a camera which is to view the world. A number of cameras can view the same environment concurrently.

The basic classes into which the problem was divided was:

World — This class ‘‘owns’’ the other classes and is responsible for starting up the interaction with the user.

Camera — Responsible for rendering and projections. Performs the basic calculations which the facet interprets as the spatial priority. (Algorithm stage 3, priority metric, and stage 5, rendering.)

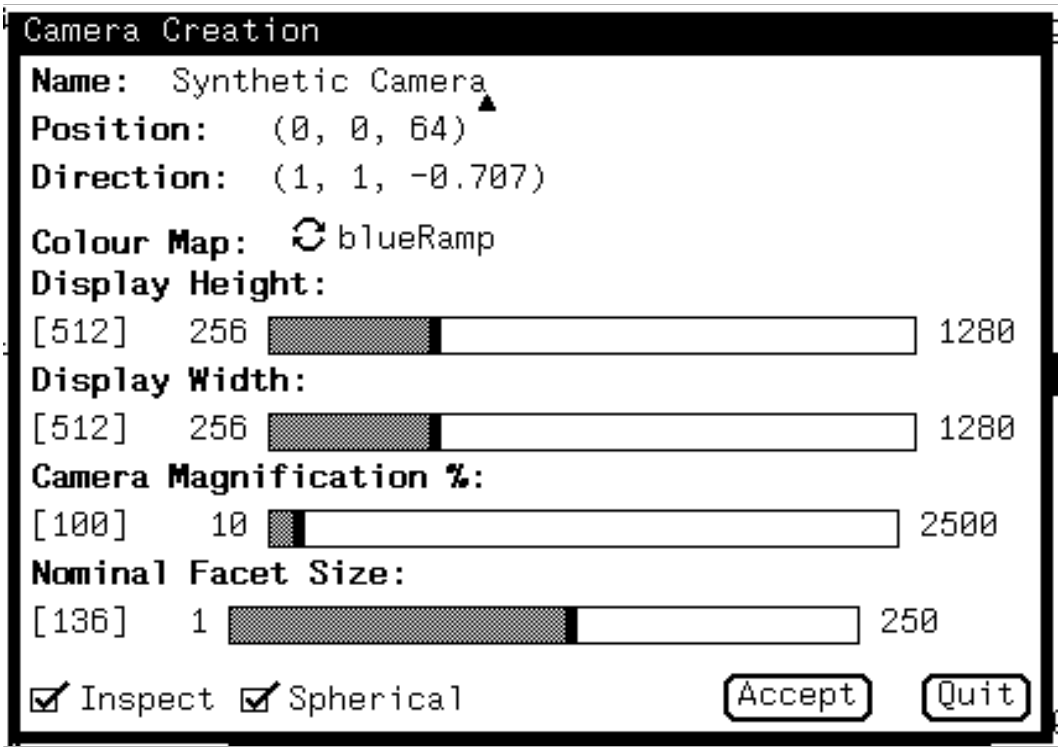


Figure 6.4. Interactive fractal generation. Setting up the program to produce a picture.

ClipTest — Responsible for initial clipping. (Stage 1 of algorithm).

Recursor — The ordering of quadrants (stage 2) and the actual recursion (stage 3) is handled by this class. The Recursor corresponds to the object model, except that in this case it is a procedural definition which can generate the data points or read them from a predefined data file.

Facet — Incorporates the information passed between Camera and Recursor. The facet corresponds to the Appearance class mentioned in other experiments. The priority metric test is done by a Facet, which decides when it has reached a small enough resolution to be rendered.

Horizon — The perimeters of the four quadrants are stored and managed by this class (stage 4).

Quaternion — This class provides coordinate transformations. The efficiency of perspective transformations depends crucially on the efficiency of rotations. See Chapter 3 for more details.

In addition there are two interface modules written in C (‘‘old’’ C) which provide an interface between the operating system and the landscape renderer (which is called static).

This implementation allows a number of Cameras to have different views of the same model. These views can differ in perspective and in rendering properties (resolution, shading, etc.). This multiplicity of views is a specialization of the notion of phantom parts introduced in §3.6.2.1. These different views are realized by means of the class Facet, an example of an ‘Appearance’. The general need for such a class was identified in this study. Each combination of Camera and model (Recursor) results in a stream of Facets being created only when the spatial priority metric indicates the necessity.

§6.4 The Results of using the Spatial Metric.

The recursive method outlined in the previous section was tested to discover benefits and limitations. The major benefit was in greatly reducing the time required to render typical views of landscapes §6.4.1. The limitation of the recursive method lay in the limited applicability of the method §6.4.2. Some coloured pictures illustrating various aspects of fractal generation are included at the end of the chapter (Figures 6.7 — 6.10).

6.4.1 Benefits of the Metric applied to Recursive Landscape Synthesis.

The spatial detail metric effectively limits the amount of detail required from a representation. For all viewpoints and view directions there is a limit on the total number of facets which have to be rendered.

Although the metric applies to all cases it is perhaps best illustrated by taking an oblique view of the landscape (as opposed to a vertically downwards view). This is of course the most common way that a landscape is viewed. Figure 6.5 illustrates the situation by means of a plan view of the facets to be rendered. The camera is situated somewhere above the origin (marked O) and is pointing slightly downwards. At each distance from the camera there is always approximately the same number of facets. On the screen they are almost the same size, but on the ground they cover bigger and bigger areas. Clearly if the metric was not in effect then we would have had a geometrically growing number of facets as distance increased. Instead of illustrating approximately 40 facets the diagram would have to contain more than 600. The exact dependence is complicated because the rate at which the number of facets changes with distance depends on the viewing conditions. With the spatial metric

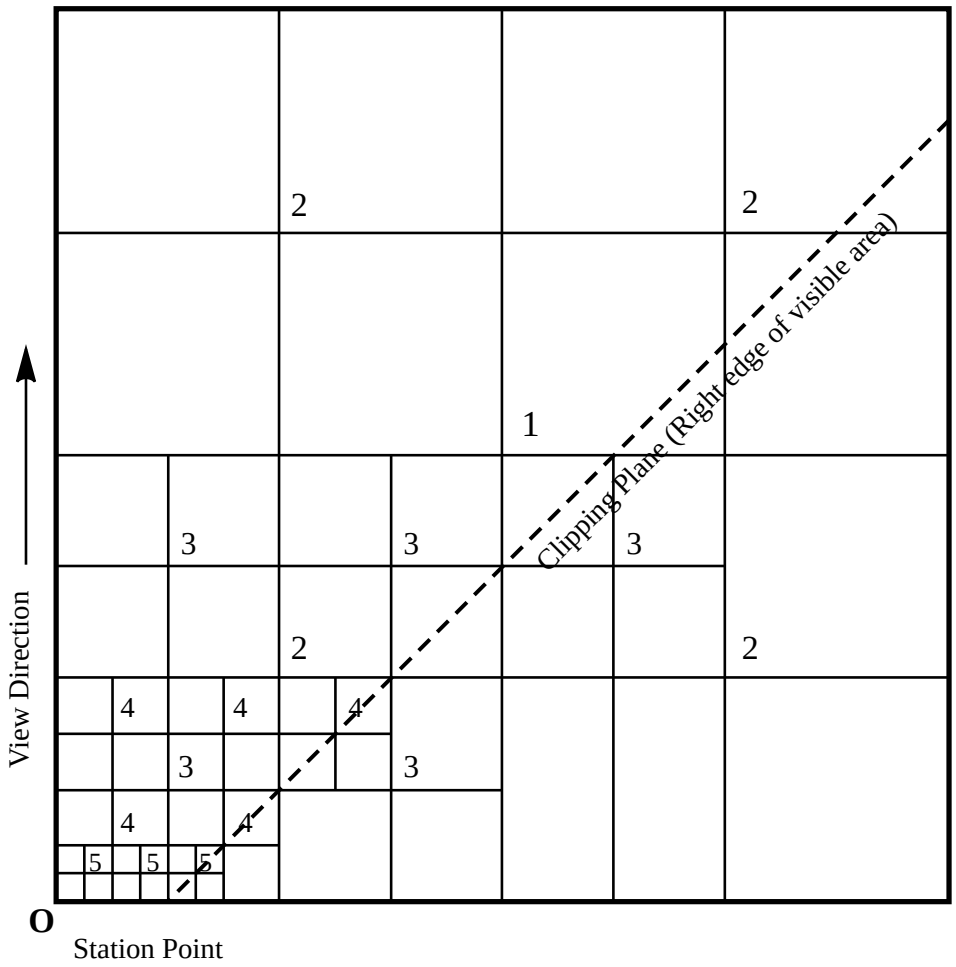


Figure 6.5. Facet plan view. Diagrammatic representation of one quadrant of a two quadrant view field. The camera is above point O looking slightly downwards. The squares indicate the sizes of the facets. The centre of each square is labelled by the level of recursion at which the point was generated. Notice how the number of facets at each distance remains fairly constant at about four.

the size of the facets increase in proportion to distance. So if the picture is a projection over a range of distances d then the saving is proportional to d^2.

This saving was first tested with a regular function with a single dominant spatial frequency and then with a series of fractals.

The function was the product of sine waves depending on x and y over the domain 0 to 2048 in each coordinate. The viewing direction was along the positive diagonal at an angle of about 20 degrees downwards (along the vector 1, 1, -0.5). The observer was at the origin at a height of 24 in the z coordinate. The priority metric was weighted to produce a nominal facet size of 16, with this metric the maximum recursion depth was 12 levels. The results

were compared with the case where the test against the metric was eliminated with recursion fixed at 12 levels for the whole visible picture.

Table 6.1 is typical of the results obtained:

	With Spatial Metric	Without Metric
Time (seconds):	32.74	19 343.10
Recursion calls:	22 585	overflowed
Facets Passed:	16 939	16 777 216
Facets Drawn:	9 557	10 841 987

Table 6.1 Typical results of profiling with a regular (sine) height field. The time is cpu time. The figure for ''facets passed'' refers to the number of facets not eliminated at the first clipping test inside the recursion. The figure for ''facets drawn'' refers to the number of facets which passed the final clipping test.

The benefit of the metric is clear. Even if the view is more vertical the metric limits spatial resolution automatically to that required for a given screen resolution.

Time

After these encouraging results a customized polygon fill was written since the standard routine was unrealistically slow. The timings were performed for a number of cases using a fractal landscape, the raw results are in Appendix B.

Firstly for a particular image size there is an inverse relation between facet size and execution time: the smaller the facets the longer execution takes. The case of 512×512 images the squared correlation coefficient for the model $t = 23.01 + 2113/f^2$ is 0.9963. The graph in Figure 6.6 plots $\sqrt{t}$ against $1/f$ to illustrate this relationship. On the same graph are the times for case where the recursion depth is constant for the whole image. These are clearly a lot slower. The shapes of the curves are similar because the constant depth recursion case also represents a variable resolution from one data point to the next, it just does not adapt within the image.

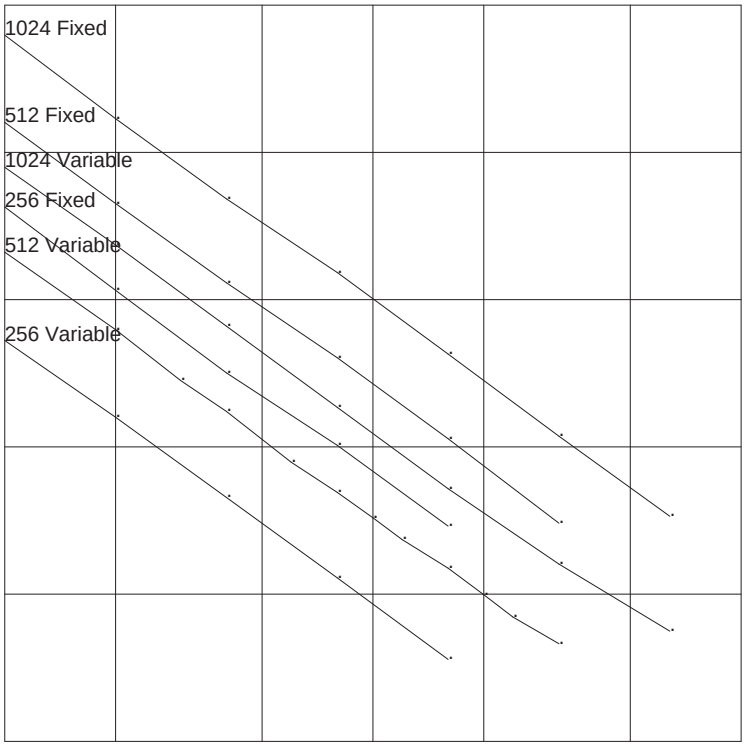


Figure 6.6 The inverse relation between facet size and execution time. The graph shows $\sqrt{t}$, where t is the execution time in seconds, against $1/f$, where f is the facet size.

The following table contains some of the times plotted on the graph:

Facet size	Recursion depth	Time (seconds) Adaptive detail	Constant resolution
32	7	4.6	30.3
16	8	15.0	113.7
8	9	49.1	402.6
4	10	174.1	1303.3
2	11	626.1	4487.9
1	12	2100.9	15978.6

Table 6.2 Summary of Timings from Appendix B for 512×512 Images.

The model given above for the relation of time to facet size ($t \propto f^{-2}$) had a very good correlation with the data. However from examining the logarithms of the facet sizes and the times a better model was found to be $t = -3.543 + 2106 \times f^{-1.764}$, i.e. the time dependency

for the particular case analysed is somewhat less than a square.

The times given above show that the adaptive detail case resulted in a seven fold increase in speed over a wide range of detail levels. This was for a simple fractal scene without any particular depth complexity. If scenes were more complex greater speed-up can be expected.

Storage requirements

The stack space required for storing the intermediate results for recursion is minimal and obviously depends on the log of the domain being examined. The other requirement is storage for the perimeter as discussed above.

Binary trees are rather inefficient compared to arrays as far as storage is concerned. However we have already seen that we have a unique index which can be derived from the quadcode of the quadrant. Thus the horizon (binary tree) can be stored as a series of dynamically created arrays. As we proceed down the recursion the size of the arrays increases by powers of two until we reach leaf nodes which can be rendered. Once the nodes are small enough to be rendered then as we have indicated already the effective quadrant which being subdivided gets smaller and smaller.

In the ideal case of a constant height field we can expect the size of these arrays to remain fairly constant as a result of the application of the spatial metric. The extremes to be considered vary from a vertical view direction where all facets have more or less the same spatial priority to a very oblique view as illustrated in Figure 6.5. Clearly in the case of a vertical view the depth of enumeration is limited by the display resolution, there is a direct correspondence between the number of facets required to tile the display and the number of quadrants into which the domain of the height field is subdivided. The spatial priority metric ensures that the same happens in the case of oblique views.

6.4.2 Costs and Limitations.

The cost of calculating the metric is minimal. It is less than 0.1% of the running time of the test program. The metric is either based on the perpendicular distance to the object which is already required for perspective projection or it is based on the square of the 3 coordinates (the square root is not needed) of the object. In the former case the comparison is with a typical dimension of the facet and in the latter it is with the square of that dimension.

More important are the limitations of the recursive method of object synthesis combined with adaptive detail resolution. The variations in sampling of the underlying function means that it can only be applied to random self similar structures. If the function is not self similar at all levels of detail then different sampling densities yield different regimes. For example: the test function of sine waves produced distinctly different shades at various distances which depended on the aliasing of detail sampling with function values (see Figure 6.10).

When applied to fractals at low resolutions there was some loss of texture gradients in the scene. In the absence of other depth cues this tended to flatten the perspective. The filtering effect of smooth shading would lessen the sharp edges of the facets at such low resolutions and might compensate to some extent.

§6.5 Conclusion.

The purpose of this chapter was to explore the interaction of the spatial priority metric with a model which had continuously variable levels of detail. For this purpose a recursive algorithm was developed to synthesize fractals.

Normally, recursive algorithms for generating fractals suffer from the difficulty of communicating between adjacent facets produced from nodes of the recursion tree which are not neighbours. This communication is needed to ensure consistency between recursions and to provide (some of) the correlation across detail levels required by fractals. The communication is also useful in that it eliminates redundant recalculation and can be used to implement smooth shading of the facets.

This inherent limitation was overcome by means of the auxiliary data structure called a *Horizon*. The storage requirements were show to be of the same order as the resolution along an edge in the object coordinates (Corollary 6.2). However, the whole purpose of the spatial metric is to apply visual resolution limits to the resolution of the objects being modelled. Therefore the resolution along object edges is also limited (§6.4.1).

The algorithm developed here is believed to be new, certainly the reduction of storage requirements from the application of the spatial metric appears to be novel.

The spatial metric proved very effective in limiting extraneous detail. For typical views of a landscape the computational benefits were immense. However it should be borne in mind

that recursive detail limitation is a very specialized method, and it is the purpose of this research project as a whole to provide a more general purpose result. As indicated in the previous chapters the strategy for achieving this lies in deriving a universal measure and then applying it uniformly to a large number of data representations. This universal application would not be possible without some form of data abstraction: this is provided by object oriented programming which is central to the methodology.

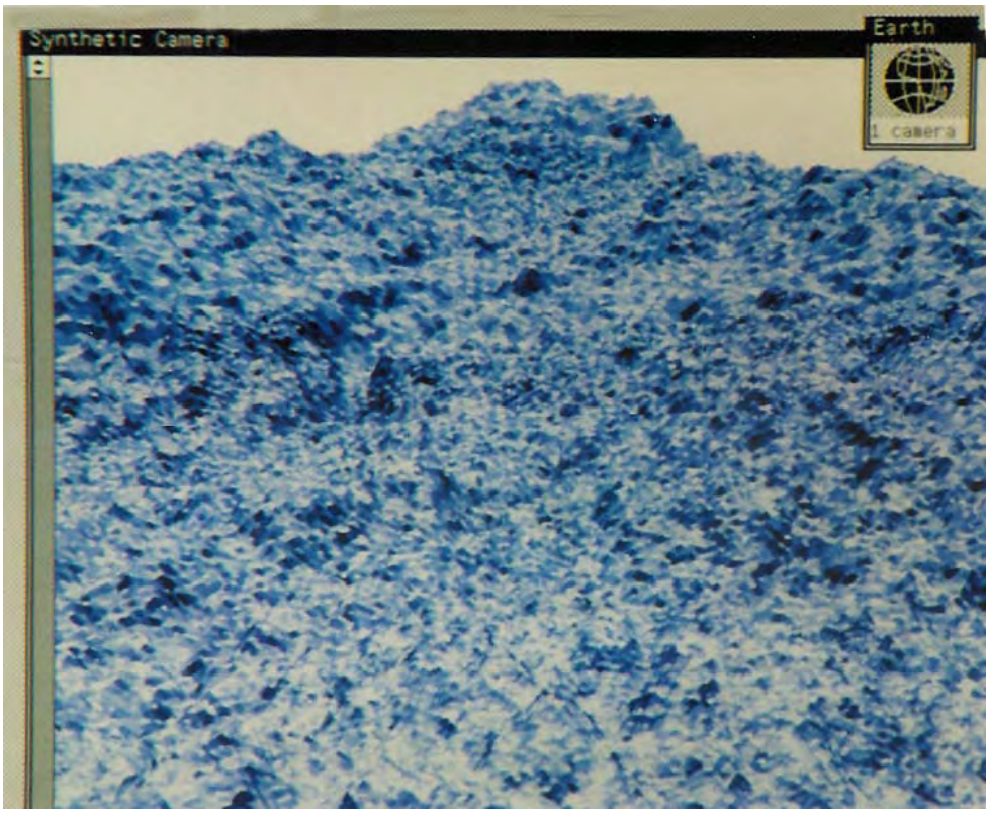


Figure 6.7. Fractal mountain at high resolution. The background was taken from a file of height fields (generated by the Fourier synthesis method). The foreground was filled in by means of the recursive subdivision method whenever there were not enough grid points. The foreground and background seem well matched (compare with Figure 6.10).

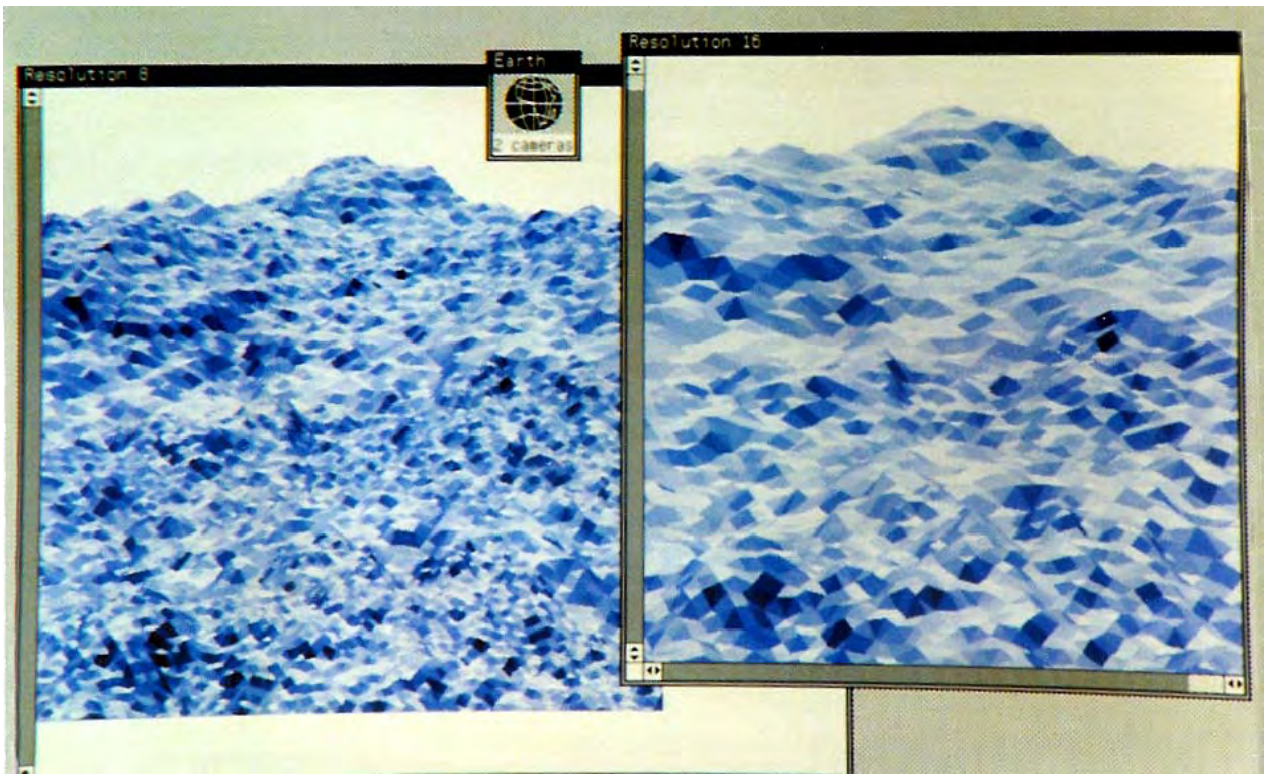


Figure 6.8. Fractal mountain viewed by two cameras at different resolutions. The nominal facet sizes in pixels are given in the frame lables (8 & 16). The same facet size was maintained in spite of increasing distance; both in the given (Fourier synthesized) landscape and in the generated parts. This same method was used in Figure 6.7, but at the higher resolution, as intended, the mechanism cannot be seen.

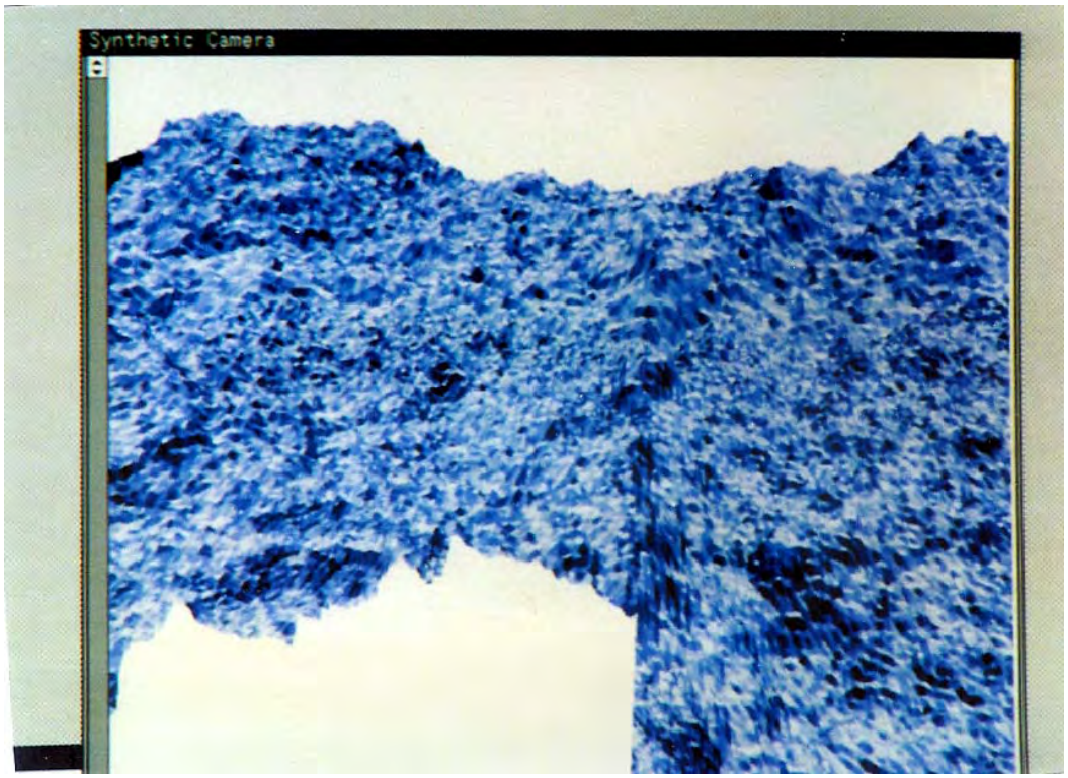


Figure 6.9. Still of the process of fractal generation. This is a side view covering two quadrants, it illustrates how the *horizons* cause separate random recursions to match up. The striations on the right foreground are due to the deficiencies of the midpoint subdivision algorithm. This view emphasizes the problem because it is along a major midpoint subdivision. It is also possible to pick out the Fourier synthesized part of the mountain in the background (top third of the frame) because it lacks such striations..

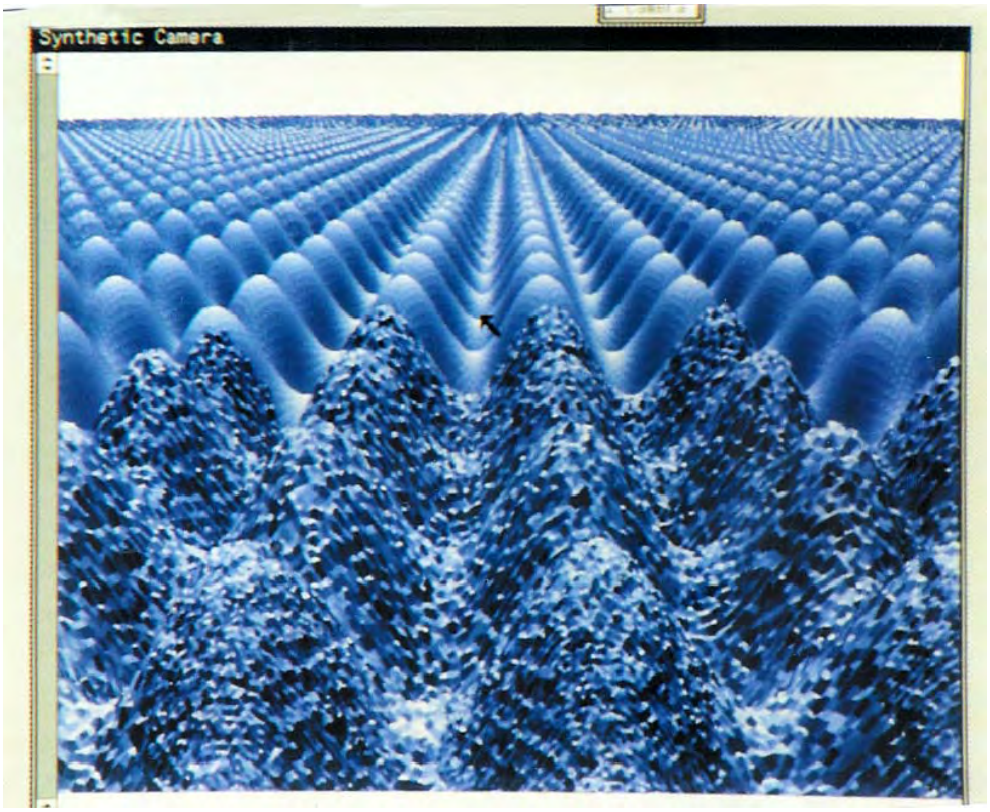


Figure 6.10. The fractal generation method applied to a sine wave field, chosen because it is the antithesis of a fractal surface. This clearly illustrates how the random midpoint subdivision effect starts up to increase resolution in the foreground, against a predefined background. In the background distinct spatial aliasing can be observed. The method presented in this chapter is naturally not meant to be applied to such height fields in such a manner.

Chapter VII

The Third Experiment: Priority Based Execution of Motion.

This chapter describes an experimental investigation into the two aspects of the *temporal priority metric*: (a) adaptive updating of moving figures according to their angular speed relative to the viewer and (b) the adaptive breakdown of three-dimensional motion as perceived by the viewer into its two-dimensional approximations. There is also an investigation of the trade-offs between temporal and spatial detail. The essential feature of the test was the application of the dynamic metric to moving objects with a high level of detail, the other features of a complete animation system were simplified.

An important, apparently novel, result of this experiment was the inversion of optic flow analysis for the purpose of synthesizing moving images. The simple animation system produced very encouraging results for the adaptive breakdown of three-dimensional motion into its ‘‘optic flow’’ components.

§7.1 Moving Figures and the Dynamic Metric.

The viewer centered approach to computer graphics and computer animation is a central theme of this dissertation. We have been exploring the benefits of regarding animation problems from the standpoint of the viewer of a scene in the natural environment. The alternative approach has traditionally been one in which objects and movements are modelled as realistically as possible (and ignoring the question of what ‘‘realism’’ really means). These realistic models are then updated and rendered by using the most powerful computing machines available. The justification for this approach is that, while machines are not yet powerful enough to achieve the current vision of the Grail, they soon will be.

The viewer centered approach is based on a ‘‘corollary’’ of that argument: existing machines will *never* be powerful enough for the tasks our imaginations can set them. The approach taken here tries to find a methodology which will unify existing tricks of the trade and allow a systematic exploration of new techniques. The aim is to provide realistic pictures without a full simulation of ‘‘physical reality’’ (§2.1).

In this chapter we apply a measure of the extent to which the following effects are realistic:

1) Moving images need not always move. Depending on the rate with which an object moves with respect to the observer we might not need to update its image every frame.

2) 3-D motion of an object can be replaced by 2-D motion of its image. A precise analysis is given of those cases where 3-D motion can be replaced by 2-D in-betweening†.

3) Highly detailed objects need not be rendered in their full detail. We have already seen how detail decreases when objects recede into the distance (the spatial priority metric), we will now explore the trade-off which occurs between spatial and temporal detail: fast moving objects need to be rendered less accurately than slow moving objects.

Actually the above statements are probably not as startling to the reader as they might have been, since the theoretical foundations were already laid in Chapter 2 and formalized in §4.3 as the *temporal priority metric*. We are now concerned with an experimental test of the metric.

The rest of this chapter is as follows: in the next subsection (§7.1.1) we formulate an experiment which will allow us to test and quantify some of the benefits of using the temporal metric and also uncover its shortcomings. Section 7.2 provides the mathematical algorithms and precise analysis of using 2-D image motion to replace 3-D object motion in this particular experiment. It depends on §4.3.

The other theme of this dissertation, object oriented programming, makes its entrance when we discuss the actual animation system (§7.3). In fact one of the occasionally controversial features of object oriented languages, dynamic binding, makes a vital contribution to this system. Lastly there is a presentation and analysis of the results of this experimental implementation (§7.4), followed by a conclusion.

7.1.1 Formulating a Test of the Temporal Priority Metric.

To test the temporal metric textured objects have to move about in three-dimensions. The objects should have the high level of spatial detail associated with natural objects. The implementation should allow rendering of the objects to more than one level of detail depending on the speed of movement relative to the synthetic camera.

---

† ''In-betweening'' is a technique used in hand drawn animation to create a sequence of frames from a few key frames. Drawing key frames requires greater skill and more effort than in-between frames. Initially is was hoped that computers could draw in-between frames automatically from the key frames. People (even unskilled ones!) apply a great deal of knowledge about relations in the 3-D world however and this automatic process has largely eluded computer animators. The 3-D information had to be made explicit as 3-D models (see §4.3.2)

A constraint on this experiment was that it be done on existing hardware: a microprocessor based colour workstation. The fact that there was no existing software for the task allowed a free hand for us to explore the benefits of using object oriented programming.

The experiment decided on was to model planar but highly textured objects executing simple movements around a mobile camera. The essential and deciding feature of such an experimental setup is that it gives highly detailed objects which we can still hope to render fairly quickly (a frame rate much better than one picture per second). There was also a secondary benefit: the optic flow effects encountered with planar objects can be made independent of depth (§4.3 & §7.2). In principle 2nd order optic flow effects are needed to completely characterize the motion of planar facets, but 1st order effects apply over small areas. First order optic flow effects are linear transformations of images. The adaptive rendering of detail was achieved by having a highly accurate, anti-aliased, but slow method for making these transformations and another, less accurate, but much faster method.

The other features of an animation system, which are not directly relevant to the test of the dynamic metric, were reduced to their essentials. The user interface was a window and mouse based system allowing the rapid alteration of experimental conditions and output parameters. The scripts which the actors followed were the simplest consistent with producing worthwhile experimental results: incremental translations and rotations over a specified set of time steps.

In spite of the simplicity all the essential elements needed for an experimental implementation were gathered: The test consisted of textured planar objects moving about. There were two ways of rendering the motion which traded speed off against spatial detail. Three-dimensional motion was mimicked by replacing it with a two-dimensional image motion when this could be done satisfactorily.

§7.2 Approximating Projected 3-D Motion with Optic Flow Effects.

There are two major questions addressed in this subsection: (a) how exactly does one compute an appropriate two-dimensional transformation of an image from the movement in three-dimensional space of a planar object; and (b) given the parameters of such a 2-D transformation how does one effect the change in the visible image corresponding to those transformations.

This section depends on sections §2.2.2 and §4.3.3 where Gibson's idea of optic flow was first qualitatively introduced and then formulated in terms of its zero, first, second and higher order effects. To recap briefly: a surface in a given direction may be described by a Taylor's series expansion of its distance about that direction. Such a two-dimensional Taylor's series first gives the distance in that direction, then the orientation of the tangent plane, then the curvatures and so on. Movement of the surface can be analysed as movement of the features in a corresponding Taylor's expansion as projected onto the image plane. In the case of a planar object the only non-zero terms in its Taylor series are the distance (zero order) and orientation of the plane (1st order). The optic flow effects above 2nd order are zero and there are only two independent 2nd order terms (§7.2.2). More complex surfaces also need the higher order terms for a complete description.

We first provide a formulation of frame-to-frame coherence which we believe provides new insight into an old problem (§7.2.1). We then describe how we intend to measure this coherence (our temporal priority metric — §7.2.2) and then how we will apply the results of these measurements computationally (§7.2.3).

7.2.1 Orders of Frame-to-Frame Coherence.

Frame-to-frame coherence was reviewed in Chapter 1 (§1.3.5). It describes the way in which one frame of an animation is normally very much like the preceding and succeeding frame. It follows that if we can identify which parts of an image have changed we need only alter those parts in the next frame. A clear, applicable, formulation which measures and predicts frame-to-frame coherence would be very useful for the adaptive updating of moving figures. We will show that optic flow analysis can be used to provide such a formulation. From our point of view, frame-to-frame coherence, or adaptive update of moving figures, is an application of optic flow effects. The more terms in the optic flow expansion which are negligible the greater the coherence between frames.

In other words, if the displacement remains relatively constant then the zero order optic flow term is very small and so nothing need be updated. In fact, using the expression for optic flow expansion we can state this more accurately: if *all* orders of optic flow effects for a particular object at a particular frame fall below some threshold then the image from the previous frame may be reused. The actual threshold used depends on the frame rate and the display resolution.

If only the camera is moving, and depending on the size and order of the depth discontinuities, the optic flow effects will apply globally over the field of view. Discontinuous depths (occluding edges) lead to disparities in the lower order optic flow effects, higher order depth discontinuities (e.g. changes in curvature) lead to higher order optic flow effects. If parts of a scene are moving independently they are treated separately.

Having formulated frame-to-frame coherence in these terms a natural progression becomes possible and we can ask questions such as the following:

1. What if all the terms except the zero order optic flow effects are insignificant?

2. What if all the terms except the first two orders are insignificant?

etc

With planar objects one would not go beyond these first two, since the most general case can only have terms up to 2nd order.

If we adopt the term *zero-order frame coherence* (which is short for ‘‘frame-to-frame coherence for optic flow effects of zero order and above’’) instead of frame-to-frame coherence, then we have the following definitions:

Zero order frame coherence: (a.k.a. frame-to-frame coherence) if *all* orders of optic flow effects for a particular object at a particular frame fall below some threshold then the image from the previous frame for that object may be reused.

First order frame coherence: If all orders of optic flow effects, except for the zero order fall below some threshold then the image of the object from the previous frame may be reused provided it is translated to its new screen position.

Second order frame coherence: If the optic flow effects for all orders from the second upwards are negligible then the image of the object from the previous frame may be reused provided it is subjected to the affine transformation determined by the first order optic flow effects.

Third order frame coherence: Defined in the same way. All planar objects exhibit this form of coherence. Third order coherence would allow one to reuse a previous image provided a transformation which changed from pixel to pixel was applied.

Clearly we can define higher order frame coherence. Equally clearly these become less useful as the transformations required become more complex. The alternative is to choose small regions over which only lower order effects reign. If these regions do not coincide with occluding edges then the separately transformed facets have to be stitched together.

7.2.2 The Temporal Metric: A measure of frame coherence.

The temporal metric is a single concept but can only be crudely approximated by a single number. As we have seen in the previous subsection the best a single number could tell us is the highest order of optic flow effect which we cannot ignore.

The basic formulation of the temporal metric is then: an integer, O_T, indicating the order of frame-to-frame coherence, or equivalently, the lowest order optic flow effect which can be neglected for an object. For the planar objects of the experiment there are four cases:

0. No relative movement, nothing has changed.

1. Translation only in the image plane — zero order effects.

2. Linear transformation in the image plane (shear, rotation, scaling) — first order effects.

3. Image must be recomputed from underlying 3-D representation — non-uniform motion, accumulated distortions too high.

For each of cases 1 & 2 there are measures of the extent to which the change has occurred. In case 1 (translation) this is specified by two numbers and in case 2 (affine transformation) it is specified by four numbers.

Measure of Image Translation: T_0.

This measure is easily arrived at. It is the projection of the instantaneous velocity vector of the object on the image plane, **v** .

Thus the translation temporal metric is:

$$T_0 = \left(v_x, v_y \right) \tag{7.1}$$

The object motion is analysed in terms of its steady translation **V** and rotation **Ω**. In that case the values of v_i (the components of **v**) are given by Equation 4.13. If we further assume

that the object is composed of planar facets we can simplify Equation 4.13 further, by eliminating the dependence on depth.

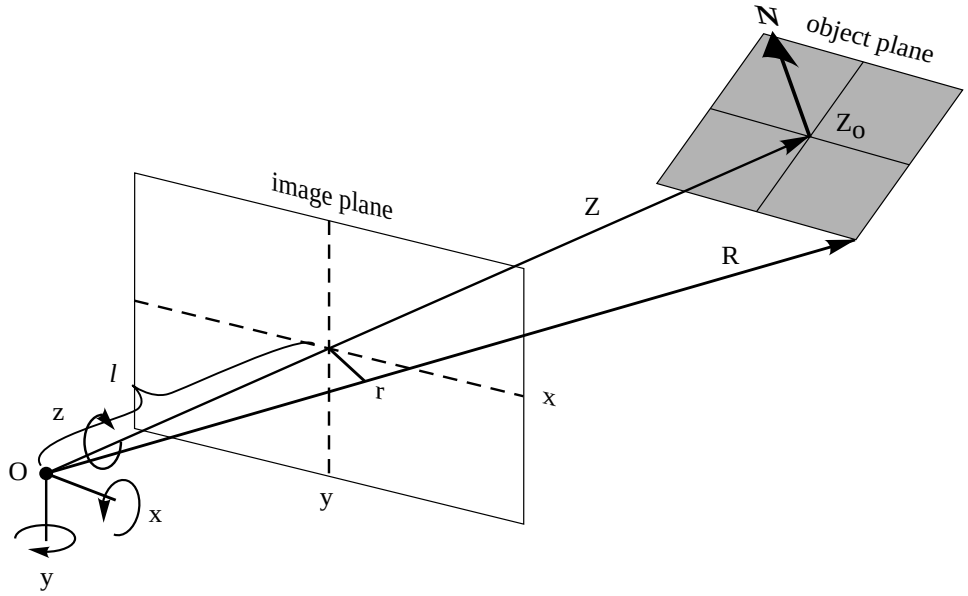


Figure 7.1. ''Re-inverted'' image projection. This figure illustrates the relations between image and plane which is used in synthesizing pictures.

Consider a plane with normal **N** which passes through the line of sight (the Z-axis) at Z_0 (Figure 7.1). The equation of the plane is:

$$\mathbf{R} \cdot \mathbf{N} = Z_0 N_z \tag{7.2a}$$

or equivalently:

$$Z = Z_0 - X \frac{N_x}{N_z} - Y \frac{N_y}{N_z} \tag{7.2b}$$

In terms of coordinates Equation 7.2a can be written as:

$$X_i N_i = Z_0 N_3 \tag{7.2c}$$

Substituting the Perspective Projection Equation (Eqn. 4.8) into 7.2c we get:

$$Z = \frac{l Z_0 N_3}{x_i N_i} \tag{7.2d}$$

The Optic Flow Equation (4.13) can then be written, for the case of the flow induced by a

plane, as:

$$\dot{\mathbf{r}} = v_i = \frac{x_m N_m V_i}{Z_0 N_3} - \frac{V_3 x_m N_m x_i}{l Z_0 N_3} + \varepsilon_{imn}\Omega_m x_n - \frac{x_i}{l}\varepsilon_{3mn}\Omega_m x_n \tag{7.3}$$

Measure of Linear Image Distortion: T_1

The measure of linear image distortion is slightly more involved. We use Equation 4.16 for the first partial derivatives w.r.t. image coordinates, $\frac{\partial \dot{x}_i}{\partial x_i}$, of a re-inverted image at a distance l from the origin:

$$T_1 = \begin{pmatrix} \frac{\partial \dot{x}}{\partial x} & \frac{\partial \dot{x}}{\partial y} \\ \frac{\partial \dot{y}}{\partial x} & \frac{\partial \dot{y}}{\partial y} \end{pmatrix} \tag{7.4}$$

In the case of a planar facet we can again simplify the equations by eliminating the dependence on depth. If we differentiate Equation 7.3 then the 1st order terms of Equation 4.14 become:

$$\frac{\partial v_i}{\partial x_j} = l V_i - V_3 x_i \frac{N_j}{l Z_0 N_3} - \varepsilon_{ijm}\Omega_m + \frac{x_i}{l}\varepsilon_{3jm}\Omega_m \tag{7.5}$$
$$- \frac{\delta_{ij}}{l}\left(\frac{V_3 x_m N_m}{Z_0 N_3} + \varepsilon_{3mn}\Omega_m x_n \right)$$

Instead of using Equation 7.5 directly for the 1st order optic flow effects, we could also substitute:

$$\frac{1}{Z^2}\frac{\partial Z}{\partial x_j} = -\frac{N_j}{l Z_0 N_3}$$

$$\frac{V_3}{Z} = \frac{V_3 x_k N_k}{l Z_0 N_3}$$

in the right hand side of Equation 4.16, viz:

$$\frac{1}{Z}\begin{bmatrix} \frac{-lV_x+xV_z}{Z^2}\frac{\partial Z}{\partial x} - \frac{V_z}{Z} + 2\frac{x}{l}\Omega_y - \frac{y}{l}\Omega_x & \frac{-lV_x+xV_z}{Z^2}\frac{\partial Z}{\partial y} - \Omega_z - \frac{x}{l}\Omega_x \\ \frac{-lV_y+yV_z}{Z^2}\frac{\partial Z}{\partial x} + \Omega_z + \frac{y}{l}\Omega_y & \frac{-lV_y+yV_z}{Z^2}\frac{\partial Z}{\partial y} - \frac{V_z}{Z} + \frac{x}{l}\Omega_y - 2\frac{y}{l}\Omega_x \end{bmatrix}$$

It can be seen above that $\frac{V_z}{Z}$ occurs on the diagonal. It is therefore a uniform scaling due to translation along the line of sight. This might not be so clear once the substitution is done.

In any event, by expanding Equation 7.5 or by eliminating *Z* from Equation 4.16 as suggested above, we get the following:

$$a = \frac{\partial \dot{x}}{\partial x} = \mathrm{p}V_x - \frac{\mathrm{p}}{l}V_z x - \frac{s}{l}V_z + \frac{2x}{l}\Omega_y - \frac{y}{l}\Omega_x \tag{7.6a}$$

$$b = \frac{\partial \dot{x}}{\partial y} = \mathrm{q}V_x - \frac{\mathrm{q}}{l}V_z x - \Omega_z - \frac{x}{l}\Omega_x \tag{7.6b}$$

$$c = \frac{\partial \dot{y}}{\partial x} = \mathrm{p}V_y - \frac{\mathrm{p}}{l}V_z y + \Omega_z + \frac{y}{l}\Omega_y \tag{7.6c}$$

$$d = \frac{\partial \dot{y}}{\partial y} = \mathrm{q}V_y - \frac{\mathrm{q}}{l}V_z y - \frac{s}{l}V_z + \frac{x}{l}\Omega_y - \frac{2y}{l}\Omega_x \tag{7.6d}$$

Where we have substituted $\mathrm{p} = \frac{N_x}{Z_0 N_z}$ and $\mathrm{q} = \frac{N_y}{Z_0 N_z}$, and where $\mathrm{s} = \frac{\mathbf{r}\cdot\mathbf{N}}{Z_0 N_z}$.

The terms of the first order transformation are not independent of screen position. The function of the temporal metric will be to indicate whether the terms for the transformation matrix from the previous frame are still valid.

The second order optic flow effects (Equation 4.15) can be written for a plane as:

$$\frac{\partial^2 \dot{x}_i}{\partial x_k \partial x_j} = \frac{\delta_{ij}}{l}\left(\varepsilon_{3km}\Omega_m - \frac{V_3 N_k}{Z_0 N_3}\right) + \frac{\delta_{ik}}{l}\left(\varepsilon_{3jm}\Omega_m - \frac{V_3 N_j}{Z_0 N_3}\right) \tag{7.7}$$

Using $\dot{x}_{xx}$ for $\frac{\partial^2 \dot{x}}{\partial x^2}$, and $\dot{x}_{xy}$ for $\frac{\partial^2 \dot{x}}{\partial x \, \partial y}$, etc., then the only non-zero terms can be written as:

$$\frac{\dot{x}_{xx}}{2} = \dot{y}_{xy} = \dot{y}_{yx} = \frac{1}{l}\left(\Omega_y - \frac{V_z N_x}{Z_0 N_z} \right) \tag{7.8a}$$

$$\frac{\dot{y}_{yy}}{2} = \dot{x}_{xy} = \dot{x}_{yx} = \frac{1}{l}\left(-\Omega_x - \frac{V_z N_y}{Z_0 N_z} \right) \tag{7.8b}$$

The definition of the temporal metric is now essentially complete. There are three related measures:

1. An indicator of the order of optic flow effect: O_T. This summarizes the information so that basic processing decisions can be made:

   (i) no update required,

   (ii) translation only,

   (iii) distortion + translation,

   (iv) recalculation of transformation matrix.

2. A vector measure of screen translation.

3. A transformation matrix for planar facets of the image. The four terms in Equation 7.6 indicate how the transformation of the image changes.

Translation of an image on a display is trivial. The only remaining algorithmic analysis concerns ways of implementing image distortions and rotations.

7.2.3 Two-dimensional Image Transformations.

The general linear (affine) transformations of images (translations, rotations shear etc) can be achieved by a vector addition, and multiplication by a 2×2 matrix. This is the way in which the results from optic flow analysis was presented: the temporal priority measures T_0 and T_1 result in linear transformations.

In the discussion below we shall refer to *source* and *result* images. In the context of implementing optic flow transformations the source image is always the original accurately computed projection of the 3-D planar facet. The result is one of a series of output images

generated purely on the basis of the source image for as long as the temporal metric indicates that this is feasible.

Translations of images are by far the easiest and cheapest transformation and can be trivially implemented. Translations by fractional pixel distances would require resampling of the source image for each result image. This does not seem to be necessary in practice.

Once translations have been performed we are left with the transformations represented by the same two-dimensional matrix applied to the coordinates of each point in the source image. It might be convenient to separate this transformation into components which can be dealt with in different ways: i.e. change in the size in the x and y directions, shear in those directions and pure rotation.

Since we are going to deal with images on a raster questions about re-sampling and aliasing will arise. We shall present two fundamentally different approaches. The one approach deals with *fractional pixels* and attempts to minimize sampling errors by distributing parts of a source pixel amongst the destination pixels. The other approach deals only with *whole pixels* and reduces temporal aliasing (i.e. flicker) at the expense of spatial resolution.

The *whole pixel method* is of course a nearest neighbour point sampling technique. It won't work very well with large changes in scale [Heckbert, 1986]. The *fractional pixel approach* implies low-pass filtering before sampling, this is computationally expensive but should eliminate aliasing effects. The terminology adopted focuses on the computational aspects of the problem, which is the prime interest here.

When shearing images the whole pixel method produces pixels which might not be quite right for their position but no pixels are lost or duplicated. The whole pixel approach can be used for overall size changes in the images. When the size of the image changes we either cull or duplicate certain pixels. Particularly with size reduction this can lead to unacceptable results because certain features in the image may disappear. The alternative is then the fractional pixel resampling approach.

There is another important difference between the two approaches however: the fractional pixel approach has to examine and interpret pixel values and must be able to combine them. The whole pixel approach for shearing combined with a duplicate/cull method of size change is *independent of the values of the pixels*. This value independence (which is really a 'type' independence) is important where the pixel values are themselves isolated points in a

much bigger colour space. For example, if the pixels are pointers into a colour table then we cannot in general expect to be able to average the two pointers and come up with a pointer to the intermediate colour (Figure 7.5 illustrates this).

These simple operations repeated over a large number of pixels make them ideal candidates for special purpose VLSI or SIMD machines. In fact the routines described below are partly inspired by algorithms originally intended for the miniDAP (a 32×32 SIMD machine) [Jackson, 1987] or for a special purpose processor [Fant, 1986].

7.2.3.1 Composition of Image Transformations.

The 2×2 transformation matrix can be written as a composition of transformations in a number of ways. The decomposition used depends on the order in which the transformations are applied. The basic image transformation which leaves its source unchanged is the unit matrix. This basic transformation is altered by the optic flow equations. The terms a, b, c, d, below, refer to the basic image transformation as altered by the corresponding terms from Equation 7.6.

If we first shear in the y direction, then in the x direction, and finally scale the image, the decomposition is as follows:

$$\begin{aligned}\begin{pmatrix} a & b \\ c & d \end{pmatrix} &= \begin{pmatrix} m & 0 \\ 0 & n \end{pmatrix} \times \begin{pmatrix} 1 & x \\ 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ y & 1 \end{pmatrix} \\ &= \begin{pmatrix} m + mxy & mx \\ ny & n \end{pmatrix}\end{aligned} \tag{7.9}$$

In the decomposition y is a shear parallel to the y-axis, x is a shear parallel to the x-axis, n represents a scaling in the y dimension and m a scaling in the x dimension. By identifying terms we can see that:

$$n = d$$

and provided $d \neq 0$

$$y = \frac{c}{d} \quad \& \quad m = a - \frac{bc}{d} \tag{7.10}$$

and provided $ad - bc$ is non-zero

$$x = \frac{db}{ad - bc}$$

If the determinant, ad - bc, is zero then the transformation is singular. This means that the image need not be treated as a plane. If the transformation is non-singular but d = 0, then we can rotate the image by ninety degrees and exchange -b for d.

If on the other hand we first apply all the y-direction changes (shear and scaling) and then the x-direction changes, the decomposition is as follows:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} e & f \\ 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ g & h \end{pmatrix} \tag{7.11}$$

$$= \begin{pmatrix} e + fg & fh \\ g & h \end{pmatrix}$$

In the decomposition *g* is a shear parallel to the y-axis and h the scaling in that direction, *f* is a shear parallel to the x-axis and *e* the corresponding scaling. Then:

$$h = d$$

$$g = c \tag{7.12}$$

and provided $d \neq 0$

$$f = \frac{b}{d}$$

$$e = a - \frac{bc}{d}$$

Similar considerations as in 7.10 regarding *d* apply to 7.12.

7.2.3.2 Fast, Pixel Preserving Image Transformation: ‘‘QUICK’’

In order to refer to this method of image transformation easily it will be called the ‘‘Quick’’ method. The quick method always moves whole pixels without interpretation, it is thus independent of the underlying type of the representation. There are four basic kinds of operation we shall need:

1. Pixel shear (translation) according to position.

2. Shearing of pixels in two dimensions together, without writing out the intermediate results.

3. Selective pixel duplication to achieve fractional size increases.

4. Selective pixel culling to achieve fractional decreases in image size.

A common theme to all these transformations is distributing a fraction of an integral set of operations over an interval. For example, choosing which pixels to duplicate to get 10% size increase, or which lines to shift for a fractional shear. This has a very efficient solution in Bresenham's well known algorithm, which is mainly used for line drawing [e.g. Newman & Sproull, 1979].

The other problem is combining two shears in two-dimensions. The simplest way would be to write out the intermediate image and then transform it a second time. However, by tracing the jagged path which an output line follows (Figure 7.2) we can derive a scanline algorithm which produces the sheared image. By duplicating or neglecting selected pixels and then duplicating or neglecting selected lines we have an efficient and general image transformation method. The algorithm notionally employed the decomposition of Equation 7.9 but without writing out any intermediate results.

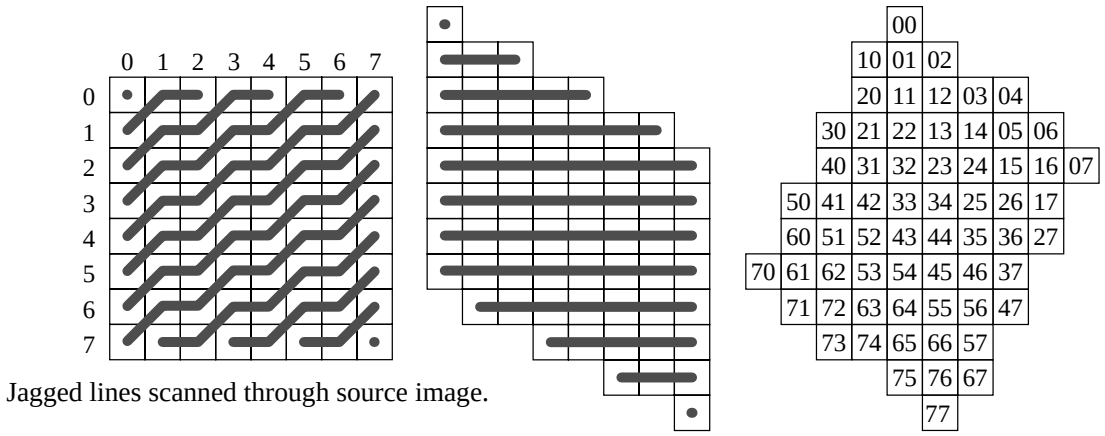
This algorithm corresponds closely with that described by Braccini & Marino [1980]. However, it avoids the problem with holes which they describe. The difference lies in the fact that they allow only traces of the source image which correspond to the diagonal moves of a chess 'bishop', while the algorithm considered here uses 8-connected 'queen' moves. The diagrams clarify why this is necessary.

This algorithm was fully implemented and its combination of features may very well be unique.

7.2.3.3 Two-Pass, Anti-Aliased Image Transformation: ''PERFECT''

This image transformation writes out its intermediate results. The pixel values are interpreted, which means that colour errors can occur from resampling. For grey level images however the output is much better, because anti-aliased, than the previous method. For brevity it will be referred to as the ''Perfect'' method.

The algorithm interpolates a selected window of input pixels to produce the desired view of output pixels. The columns of the image are first processed and then the rows of the resulting intermediate image. Because of the two pass nature of the algorithm shears can be done easily for each dimension separately. The ''perfect'' method is well suited to VLSI implementations [Fant, 1986], and the reader is referred to Fant's article for more details. The decomposition derived in Equations 7.11 and 7.12 apply to this algorithm. Using the



Jagged lines scanned through source image.

Bresenham's Code in Example: 1010101.
If x-shear were less than y-shear, then source scanlines of the following shape could arise:-

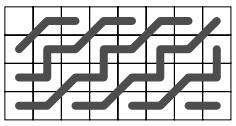


Figure 7.2. Rotation by multiple shearing of an image. The diagram shows the correspondence between result scanlines and the source image. Note: the intermediate image is purely for illustration, it is not actually produced

decomposition into the shear and scaling components avoids the rather laborious procedure used by Fant which involved calculating the positions of the four corners of an image.

This algorithm was implemented as an option which can be applied to all images, instead of the more normal one pass method.

7.2.3.4 Simple Averaging Image Transformation. ''PRETTY''

Because the two pass algorithm was so slow there was an attempt to produce an intermediate algorithm. This algorithm is essentially the same as ''Quick'' except that pixels are not culled when the image decreases in size. Instead a simple averaging is performed. Successive pixels which would be discarded are accumulated and averaged with the normal output pixel of a location.

The algorithm was only used for images which decreased in size. This was the most common bad case for the ''Quick'' method.

§7.3 A Simple Priority Based Animation System.

The basic algorithms for calculating temporal priority in terms of optic flow were presented in §7.2.2. In the subsequent section (§7.2.3) the algorithms for performing the image transformations were given. In this section we describe the way in which these algorithms were implemented and linked up to produce an animation system. Not just *any* animation system, but one where the temporal priority of an object determined how it would be processed and rendered.

The basic specification for the animation system were determined by the requirements of the experiment (§7.1.1), that is, to illustrate and provide a simple testbed for the temporal priority metric. The system was implemented on a SUN-3 colour workstation and embedded in the standard sun windowing system. The user interface was built up with the toolkit provided by Sun. The main requirement of the user interface was that experimental parameters could be altered quickly and easily.

The main programming language was C++, although the interface between the animation system and the windowing system was written in C.

When it was discovered that a significant proportion of the execution time was spent in the computer supplied raster operations, a few critical routines were speeded up by the judicious application of assembly code. This optimization applied to getting images on the screen and not to the image transformation routines which were all written in C++.

7.3.1 The Parts of the Animation System.

The animation system is thus split between the user interface part and the animation production system proper (see Figure 7.3). The animated world consists of three basic types of objects and numerous supporting objects (class names are indicated by capital letters):

- Stage. The global object which contains all other objects as parts or sub-parts. An important part of the Stage is the Clock. There is only one Stage per animation.

- Actor. An actor is the basic unit of animation. The actors have a Script which is the sequence of actions which they execute. They have an Appearance which determines how they will look on a display. There can be any number of Actors in the world. Appearances exist for every Camera.

- Camera. The camera takes the Appearance of an Actor and renders it on a Segment.

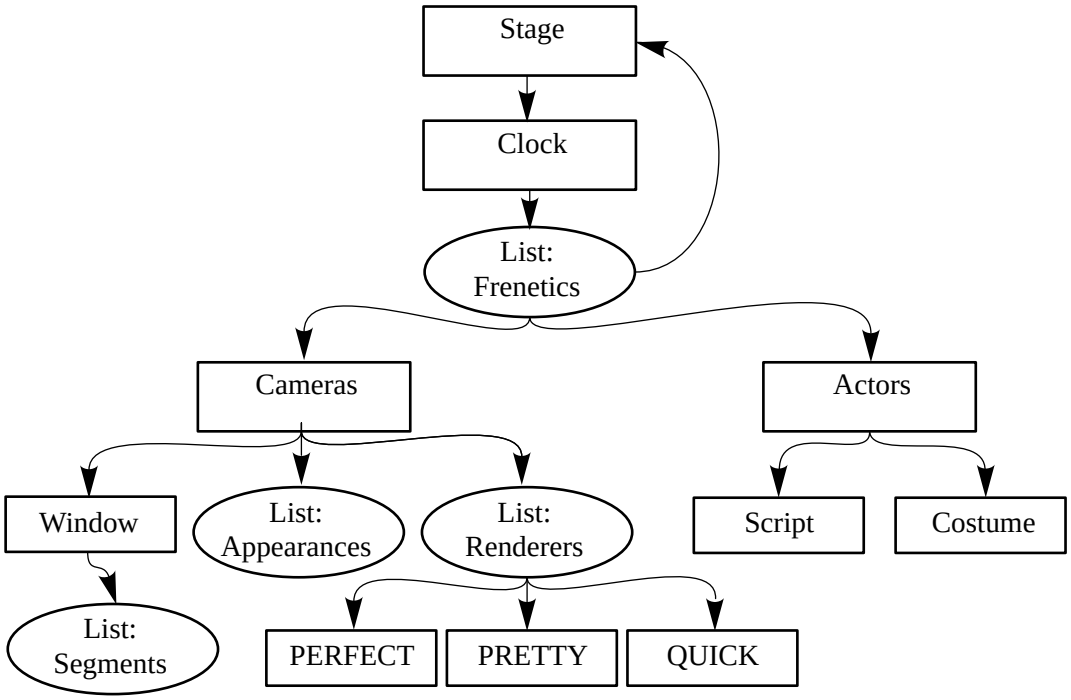


Figure 7.3. The part hierarchy of selected classes in the animation system.

The Segments are submitted to a Window to be displayed. Although there can be a number of Cameras the current user interface only allows one to be set up. The Camera owns a number of renderers, Quick, Pretty and Perfect, corresponding to the algorithms discussed above.

These objects are linked by a number of lists. The three most important ones are:

- Appearance List. The temporal priority is embodied, if it is embodied anywhere, by the list of Appearances ordered according to their temporal priority. An instance of this list is owned by each Camera.

- Segment List. Each Window has a list of Segments ordered according to their display priority (depth priority).

- Kinetic Object List. Every object which can move is placed on a list owned by the Clock. These objects are updated once for each each frame that is rendered.

The actual calculation of the temporal metric is performed by an instance of TimeMetric or its subclass PlaneMetric. Each Appearance owns one of these objects. The basic class TimeMetric can only calculate the simple translation temporal metric T_0 and is used with

actors which cannot move in three-dimensions. The subclass PlaneMetric finds T_1.

7.3.2 The Animation Classes and Processing Cycle.

Up to now we have been talking in terms of the part hierarchy, that is, which object owns which, how the parts fit together to form the whole. The other side of an object oriented implementation is the class inheritance hierarchy (Figure 7.4, cf. §3.6.1). All objects which can change over time are subclasses of the abstract superclass KineticObject. So the Stage, Actors and Cameras are all subclasses of KineticObject.

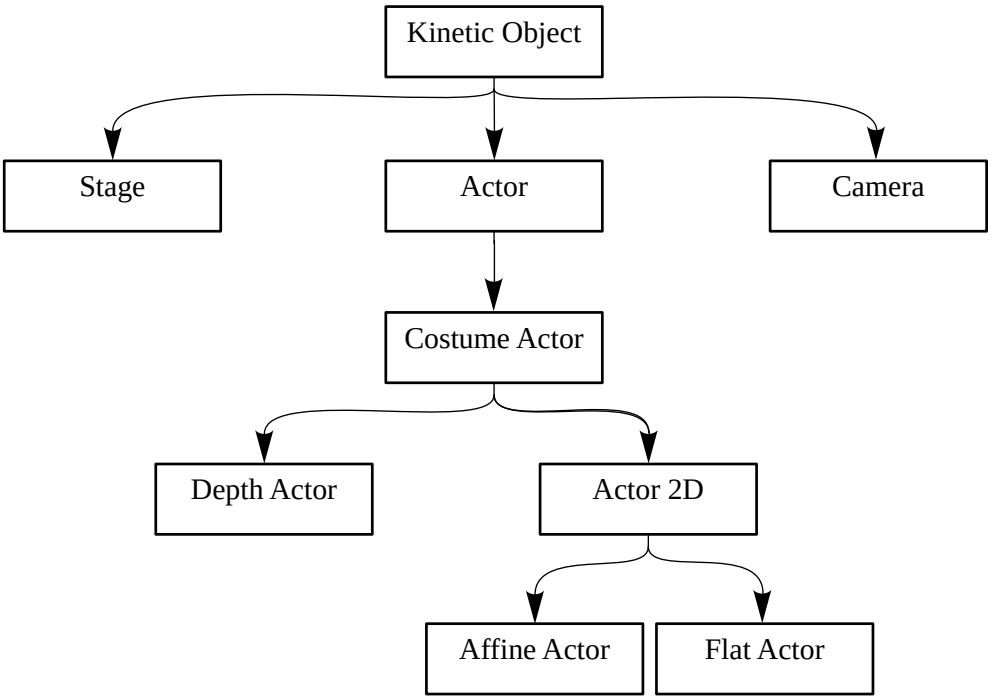


Figure 7.4. Part of the class hierarchy of the animation system.

Appearance is an abstract superclass, the real task of interpreting the way an Actor looks to a Segment is performed by its subclasses. The importance of these class hierarchies will become apparent once we have considered the processing steps which take place during animation.

Animation processing is centered around the cycle of operations which produce the output frames. At the center of this processing cycle is a clock object which distributes a ‘tick’ to all objects and which notices when a cycle is complete. Since this system is implemented on a single processor machine synchronization is achieved by distributing the ‘tick’ in some correct predetermined order, but the idea is extensible (see Appendix D). This order is

determined by the order in which objects appear in the KineticObject list mentioned above.

The first objects to receive ‘tick’ are the Cameras. They then proceed to render the Appearances received in the previous cycle. Then they update their own positions. This has to be done before the Actor’s are activated because temporal priority depends on the relative positions and speeds of Camera and Actor.

Next the Stage receives ‘tick’. Actually the very first object to be created is the Stage, it owns and sets up the Clock and is informed when the Clock has completed processing. During processing the stage maintains a subsidiary list of suspended but not deleted Actors, but the main purpose of sending a tick to Stage is to allow it to function as a kind of background actor.

Finally the Actors receive a ‘tick’, in no particular order. In response they update their position by asking their Script for the next action to be performed. Once updated their temporal priority is calculated w.r.t. each Camera. The Stage supplies an iterator object which will answer with each Camera in turn, when it is interrogated by the Actor.

The vital temporal priority calculation is performed at this stage. For each Camera a new Appearance is created if needed. The Actor’s 3-D state is transformed and stored in the Appearance as a 2-D state relative to the Camera. Each Appearance is then submitted to the Camera by placing it on the appropriate list. This list reflects the temporal priority of the Appearances by the order they are kept in.

There can be numerous kinds of Actors. The full optic flow calculations are done for DepthActors. The other actors (AffineActors and FlatActors) allow more direct control over their transformations and are mainly used for testing and for timing the adaptive updating aspect of the experiment.

The Appearances for all these Actors belong to the subclass Picture. It is Pictures which know about the 2-D textured images (instances of the class Costume).

The Camera takes the Appearances from its appearance list and depending on their temporal priority has them rendered by a Renderer. These renderers (instances of PerfectPic, QuickPic and PrettyPic) embody the algorithms discussed in §7.2.3. The is also another renderer not mentioned before which can only translate Appearances.

The Renderer returns a Segment which has a screen position and depth priority. A bounding box is also provided to indicate the area of the segment actually used by the image. These Segments are maintained in a depth priority list. This list provides a simple method of hidden surface removal. Segments are rendered depth first.

The processing cycle for one frame is now complete and a new one can begin. The Stage is informed of this. Certain housekeeping such as image recording or user interaction takes place at this stage of the cycle.

7.3.3 Generic Lists and Dynamic Binding.

In the discussion of the Appearance and KineticObject list we have glossed over a crucial point: these lists can contain many different kinds of object. When a particular object is taken from the list we do not know its type beyond the fact that it is either a kind of KineticObject (on the Clock's list) or a kind of Appearance (on the Camera's list). Clearly this will be a feature of any animation system with many kinds of active objects or any implementation of a priority measure which must apply to different kinds of objects.

It is here that dynamic binding provides the most elegant solution. We send the same message to all objects in the clock queue (i.e. 'tick') and the same message to all objects in the Camera queue (i.e. 'scanConvert'). Then at run-time the system binds the correct function corresponding to the type of the object encountered. In C++ these functions are known as *virtual* functions. In general at least two more virtual functions are needed: 'printOn' to print a description of the object on some output stream, for debugging and general enquiries. And a virtual destructor which will clean up behind the object if it is deleted.

A less elegant feature of the generic lists which are used to implement these queues is the way they are declared: they are generated by a rather baroque collection of C preprocessor macros. Their virtue is that creating a new kind of list is simply a question of naming it.

§7.4 Results of Using the Temporal Priority Metric for Animation.

The experiment described in this chapter had a number of different aims which all came under the single heading of testing the temporal priority metric. The aims may be broadly divided into three categories:

1. Adaptive updating depending on relative movement.

2. Replacing three-dimensional motion by two-dimensional linear image transformations with an error bound.

3. Showing trade-off between spatial and temporal detail.

Generally we were able to obtain promising results, particularly in the first two cases. The most important result was the ability to mimic three-dimensional motion by two-dimensional flow effects. However in all cases areas for further investigation were uncovered. In the first case (adaptive updating) this was because the experiment only really allowed a few levels of adaptation. In the second case (optic flow effects) this is probably because it is a new technique with many more possibilities than could be examined here. Finally, trade-off between spatial and temporal detail seems to depend on being able to filter fast moving images more accurately and yet still display them at high speed.

The results presented here include timings. These timings can be used to indicate the comparative worth of the various adaptive detail schemes within this implementation, but there are no absolute benchmarks for testing speeds. This experiment, unlike the previous one described in Chapter 6, tests a large range of effects. It shows that the temporal priority measure can be used to unify some old ideas (adaptive updating, frame-to-frame coherence) with some newer ideas. The experiments also indicate that the new techniques (optic flow analysis, detail trade-offs) are practical.

7.4.1 Adaptive Updating.

There are various kinds of adaptive updating available on the system. Firstly images may be rendered by the ‘‘Quick’’ method or the ‘‘Perfect’’ method. Then if the system detects that a previously rendered image needs only to be translated on the display no rendering is required (see Figure 7.6). The previous segment is merely redisplayed in the new position.

Within this three tier system the temporal metric provided an adequate measure of the importance of the objects. The results of various timings conducted are collected in Appendix C. Even in this simple test there were many variables. For example, an object receding into the distance becomes smaller and quicker to render and so the transformations which mimic this effect have an advantage. For this reason the transformations used were all approximately rotations.

Actor		Frame Rate (Frames/second)		
code	nominal size	Translate	QUICK transform	PERFECT transform
bc	200 pixels	16	3	0.4
fd	75	60	33	3

Table 7.1 Summary of timings from Appendix C. The code refers to the codes used in Appendix C to identify actors. The nominal image size refers to a typical dimension of the image before transformation. All frame rates have been rounded to one or two digits of precision.

The other transformation method, ‘‘Pretty’’, produced times almost exactly the same as ‘‘Quick’’ and its visual appearance was also very similar. The most visible aliasing effects are on the edges of objects and ‘‘Pretty’s’’ simple averaging did little to correct this.

It can be seen from Table 7.1 that being able to decide what the dynamic priority of an object is can produce very great speed improvements. The table indicates that the benefit of avoiding anti-aliasing is much the same for both big and small images (about 10 fold speed increase). When it comes to doing translation only instead of image transformation the benefit is much larger with large images (5 fold *vs* double speed-up).

7.4.2 2-D Image Motion Instead of 3-D Object Movement.

The basic results were very encouraging. Three-dimensional motion effects were created in the absence of any explicit underlying three-dimensional simulation (see Figures 7.7a-f). For certain extreme distortions the image transformation matrix was not stable enough and resulted in visibly inaccurate image positioning. However the knowledge of these cases can be incorporated into the temporal metric.

The timings for these transformations are the same as for the basic image transformation case given in the previous subsection. The calculations involved are minimal, and many results are re-used and can be cached from one frame to the next. For example if the three-dimensional velocity and rotation of the object remain unchanged then only the terms which depend on image position need be recalculated.

The success of this section of the experiment depends on the conviction that the sequence of images in (Figure 7.7) represent an example of three-dimensional motion. These images

represent a way of doing three-dimensional in-betweening on the basis of two-dimensional transformations.

7.4.3 Trade-off between Spatial and Temporal Detail.

The effect was illustrated with small images rendered by the ‘‘Quick’’ method. When such images recede into the distance their images get smaller. Thus there is a loss of spatial detail. Normally such images would be filtered to remove the higher frequencies and prevent aliasing. However the purpose of this investigation was to see if moving such objects at high speed without filtering would be acceptable. The assumption was that the jagged loss of detail would be associated with high spatial frequencies. This tested the hypothesis that such high spatial frequencies would be less objectionable in the fast moving image.

The control for this experiment was the same transformation done by the ‘‘Pretty’’ and ‘‘Perfect’’ method. The hypothesis would be proven if the ‘‘Quick’’ method (not spatially anti-aliased but fast) gave the same visual impression as the other slower, but better anti-aliased, methods.

The results were reasonable given that a fast play-back of the images could only be achieved by making them small (60×60 pixels). The results for both the ‘‘Quick’’ and the ‘‘Pretty’’ methods (i.e. the experiment and the control) seemed very similar.

The problem with the experiment lay in the controls. The ‘‘Perfect’’ method was too slow to give a real sensation of movement on the present hardware. While the anti-aliasing of the ‘‘Pretty’’ method was not good enough to form the ideal control: a moving anti-aliased image.

In absolute terms the ‘‘Quick’’ method does cause ‘‘boiling’’ and flickering of the moving images. This experiment was not conclusive in showing that this was less objectionable in very fast moving images. Further tests of this trade-off will require much faster rendering techniques.

As a result of this experiment we can conclude that it might be better to look for the benefits of the trade-off between spatial and temporal detail at an earlier stage in the viewing pipeline. In other words, all final images will have to be properly anti-aliased. The benefit of the trade-off will be that for fast moving objects less detail will have to be calculated for eventual rendering. For example: it can be conjectured that fast motion over the fractal field of the

previous chapter will allow the recursion depth for fractal calculation to be decreased compared to the static case.

§7.5 Conclusion.

We have presented a coherent analysis of adaptive detail in time for use with animated pictures. This analysis is based on the idea of a temporal priority metric which measures the visual importance, in a precisely defined sense, of an image to the viewer. This resulted in a new, extended, formalization of frame-to-frame coherence.

This technique should find application whenever highly detailed objects are subject to motion which is smooth over a number of frames. For example: a recent application of object oriented animation to natural environments was Reynolds' simulation of flocks, herds and schools of animals [Reynolds, 1987]. The motion of such animals can usefully be rendered using all the techniques presented above. Although Reynolds ignored wing motion this could also be allowed for modelling the two wings and the body as separate actors. The cyclic nature of wing movement could be incorporated directly into the 2×2 image transformation matrix.

The synthesis of pictures from an optic flow decomposition in particular seems to be new and quite promising. One area of application would be to bring the possibility of three-dimensional animation low cost personal workstations which could perform fast two-dimensional transformations on images. The transformations required are more complex than the raster operations currently implemented for such machines. A possible architecture would be some kind of SIMD parallel array processor such as the 32x32 AMT DAP 510 ('miniDAP') or special purpose VLSI.

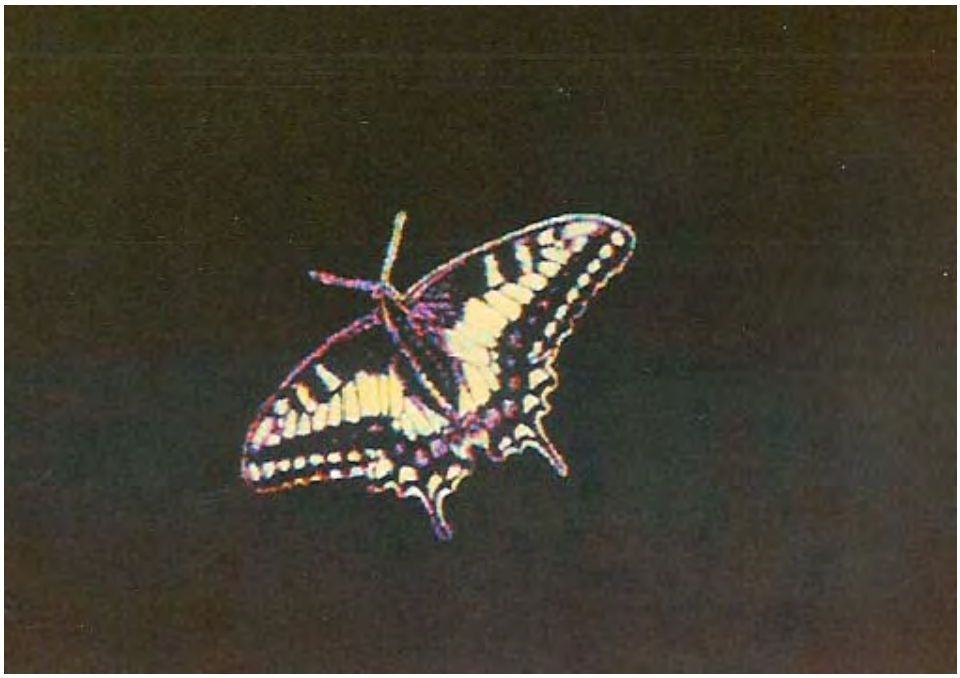


Figure 7.5. Illustration of colour effects of smoothing pixels in a colour table. This effect is observed because the actual values of the pixels was manipulated. The other full colour pictures show the correct colours. (see §7.2.3)

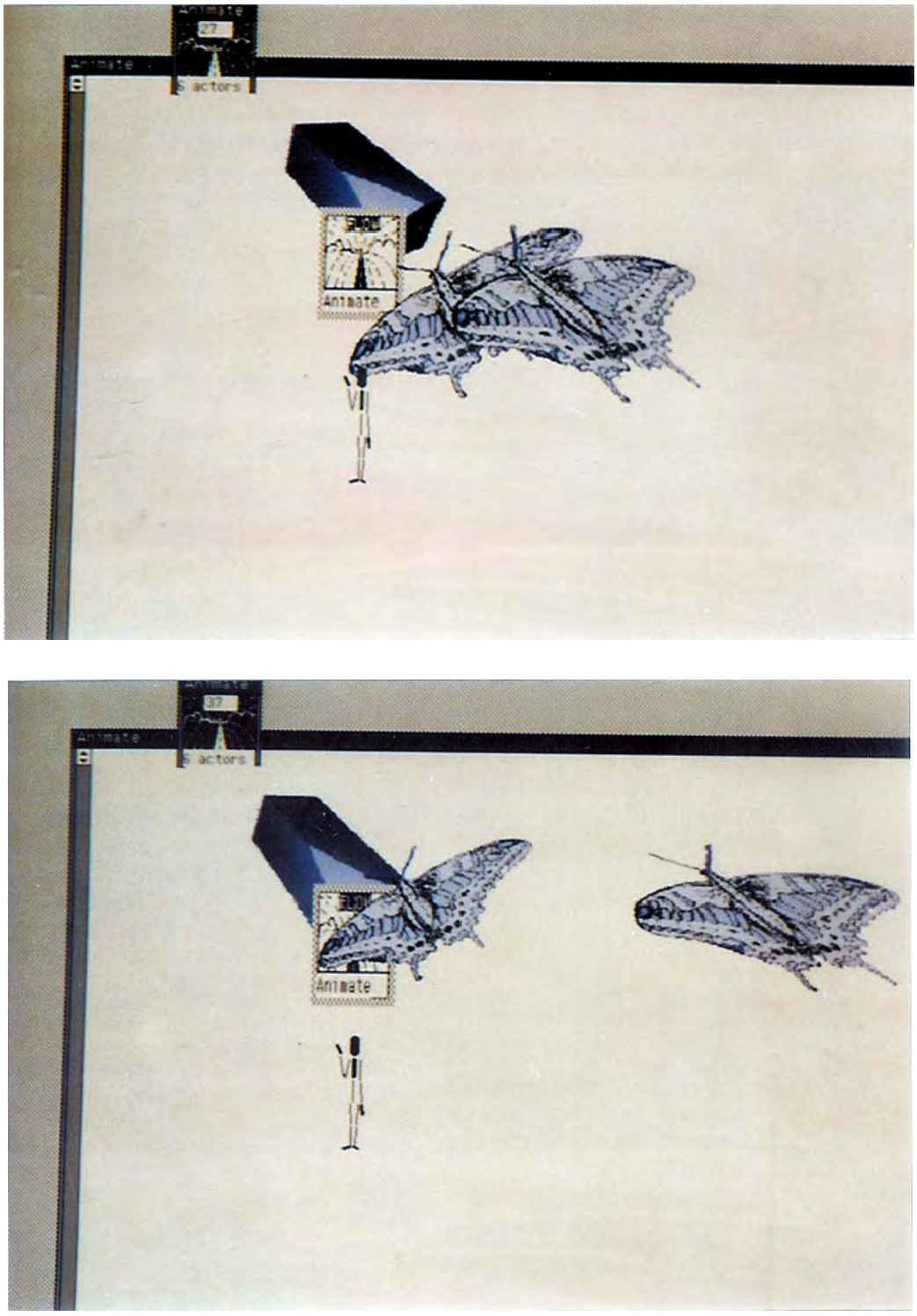


Figure 7.6: a & b. Multiple actors executing different motions and being updated for different frame-to-frame coherence effects: 0 order for the ‘stamp’ and stick figure, and 1st order for the butterflies and brick.

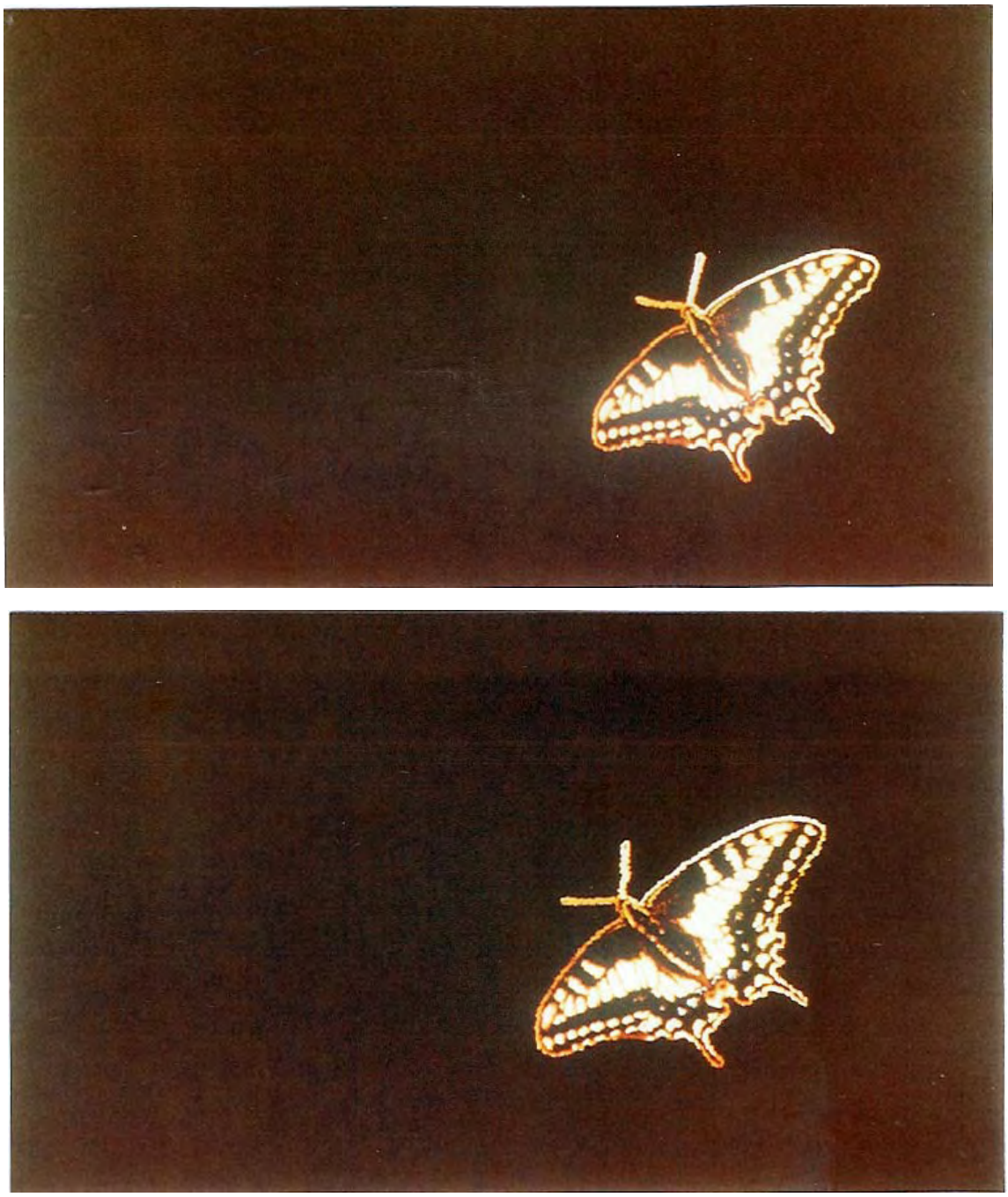


Figure 7.7: a-b. Frames from a sequence of distorting two-dimensional pictures which mimic three-dimensional movement. The sequence comprised many more frames than the 6 shown over the following figures. The only internal representation is a planar picture. The images were transformed by the ‘‘Quick’’ algorithm.

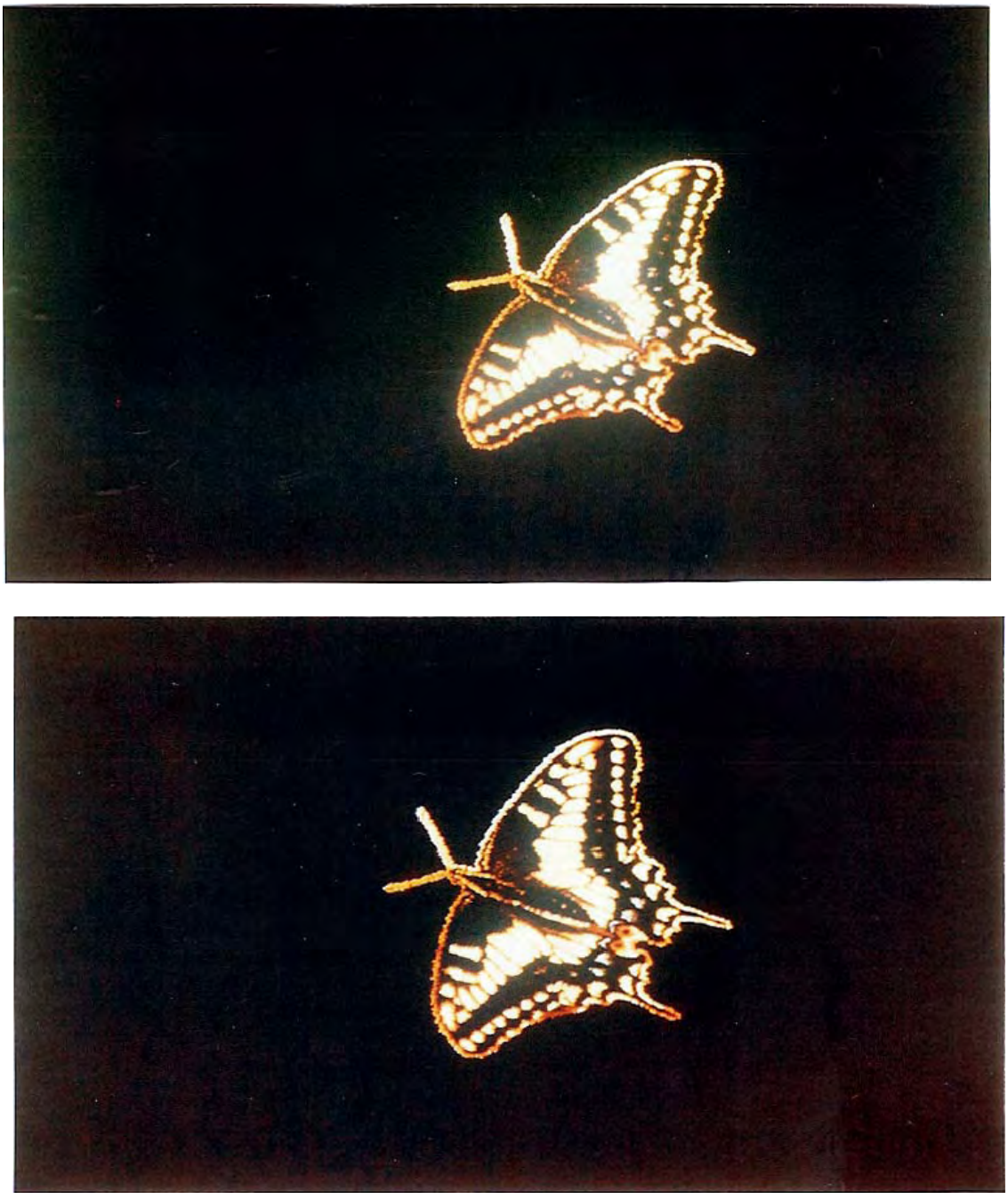


Figure 7.7: c-d. Continuing frames from a sequence of distorting two-dimensional pictures which mimic three-dimensional movement.

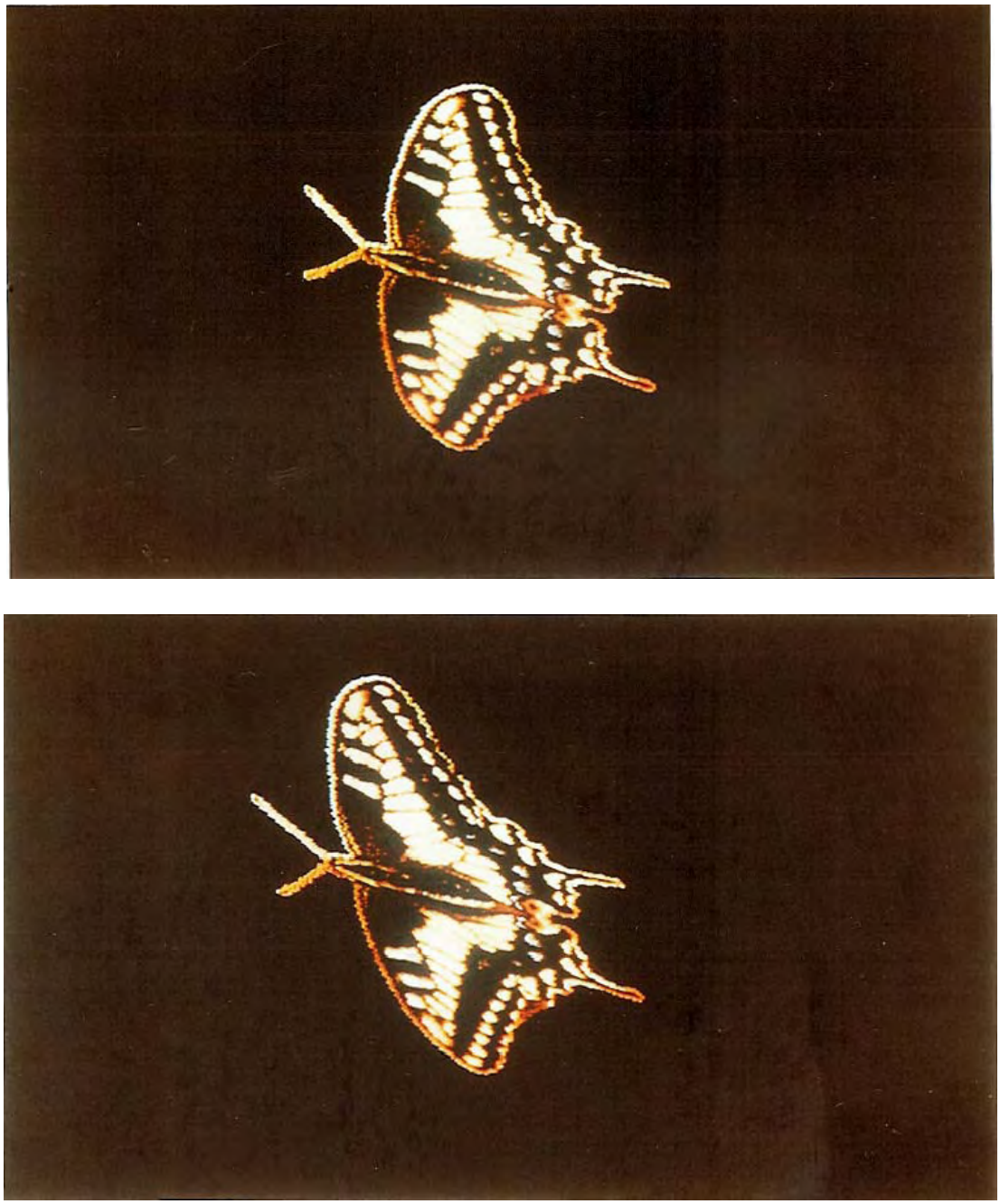


Figure 7.7: e-f. Concluding frames from a sequence of distorting two-dimensional pictures which mimic three-dimensional movement.

Chapter VIII

Conclusion

This chapter summarizes the achievements of the research in the light of the main goals: the formulation and application of the spatial and temporal priority metrics. A methodology for designing adaptive detail computer animation systems, another aim of the research programme, is summarized. Some of the theoretical results are also reviewed, particularly the work on representing physical objects in object oriented programming.

The chapter concludes with a number of suggestions for further research.

§8.1 The Main Implications of the Space-Time Priority Metric.

We have defined a collection of priority metrics which measure how much effort should be expended by a computer animation system in rendering an object. The kinds of objects considered were explicitly those encountered in nature, since the intricate texture of natural objects makes them particularly challenging. Nature is the proving ground for any technique which concerns itself with realism.

As we have seen there are a number of different metrics: the *static priority metric* which deals with spatial detail, the *temporal priority metric* which deals with (various orders of) dynamic detail, and trade-off relations between the two metrics (summary in §8.1.2).

The theoretical issues involved are summarized separately (§8.1.1): the main contributions are:

- Distinguishing part hierarchies from type hierarchies and providing an access mechanism for parts which is consistent with data encapsulation principles.
- Refining the notion of a variable detail object to mean those objects which possess an ‘Appearance’. The protocol of this class mediates the adaptive display of an actor by a camera on the basis of the actor’s static and dynamic priority.

The practical work in this research programme concentrated on applying the priority metrics to three features of the natural environment (the reasons for this choice are reviewed below in §8.1.1):

1. Animals — the stick figures of Chapters 3 and 5.

2. Textured landscapes — the fractal height fields of Chapter 6.

3. Optic flow — Textured planar facets in motion of Chapter 7.

In general these experiments confirmed the main tenets of the thesis:

- The priority metrics could be defined and applied.
- Using a static priority metric reduced processing time because redundant detail was eliminated at an early stage.
- The static metric reduced storage requirements because the object models were not enumerated to their full extent. Less intermediate storage was also required during enumeration.
- For animation the temporal metric allows adaptive updating to be exploited in a consistent and unified manner.
- Frame-to-frame coherence was extended to include various orders of coherence between objects on successive frames. That is, images of objects could be completely unchanged, translated in the image, undergo other affine distortions or be subjected to more complex changes.
- By directly appealing to the optic flow equations, and by making use of the temporal priority (error) measure the full 3-D rendering of an object could be bypassed by a synthetic camera for a limited number of frames. This process depended only on the previous image and the ‘‘Appearance’’ and not on the 3-D object model.
- Some indication of the possible trade-offs between spatial and temporal detail was given.
- The temporal metric reduces intermediate storage requirements by allowing simpler rendering options to be used. For example, one pass rendering on the basis of 2-D image rather than full 3-D rendering.

This research programme had the aim of providing a method of tackling the complexity of representing objects in computer animation. This method has now been provided and applied. The summary is provided in §8.1.3.

More broadly speaking, the thesis defended was that it is rewarding to view the representation of objects in terms of the impression we want to make on the viewer's senses. The success of the priority metrics can be taken to illustrate this benefit. It is hoped that this dissertation can contribute to shifting the emphasis of naturalistic animation away from the physics of objects and towards the mind of the viewer.

8.1.1 Summary of Some Theoretical Issues Explored.

The purpose in defining the priority metrics is to capture the idea that fidelity to reality depends as much on catering for the needs of human perception as it does on accurate modelling. The generation of realistic pictures also depends on the characteristics of the display device and display viewing conditions, since these come between the viewer and the computer model of the environment. For this reason it is perhaps better to refer to ''observer conviction'' rather than to ''realism'' as the desired quality of pictures.

It was emphasized in Chapter 2 that the human eye is nothing like a camera. It deals with continually changing arrays of light, it has no shutter, it does not see static images. In a sense it does not see patterns of light in space: it analyses the incoming sense impressions in a series of tuned spatio-temporal frequency channels. One can conclude from this that we need not necessarily be concerned with creating sequences of complete frozen frames for an animation. Instead we can adaptively update parts of the frame in such a way that the sequence of distorting images taken as a whole provides the information which the human senses require.

This research has set itself the aim of measuring properties of an image which are needed to make those images convincing to the viewer. It has concentrated on adaptive detail in space and time. The particular metrics used will be discussed below (§8.1.2). It is important for the purpose of this research that practical measures of detail can be derived from the general principles and can be applied to a range of data representations.

It is also important though that we establish the general principle of *a priority metric* which can be applied to a wide range of objects. In order to do this we defined the metric in terms of its applicability to a general variable detail abstract data type. This was first done in Chapter 4 and was then elaborated as the experiments illustrated more of the issues involved. The central idea here is that an object creates an instance of an *''Appearance''* type for every camera in the environment. This Appearance incorporates the current spatio-temporal detail

level required without explicitly rendering the object. The detail levels required are exactly the priority metric values.

The ability to define such an Appearance has implications for the modelling of the underlying object — some objects will behave more efficiently than others in this respect. An Appearance is an encapsulator: the underlying model could have only one level of detail without preventing its use in this situation. Needless to say the most efficient application of the priority metric is with objects where an early determination of priority can eliminate unnecessary computation or reduce storage requirements (see also §8.1.3).

The question then arises of whether the kinds of objects typical to natural scenes would interact efficiently with the priority metrics. Chapter 2 was partly concerned with establishing some distinguishing features of the appearance of the natural environment. The most important property was the many, *nested, levels of detail* and the way these nested levels of detail distort during motion: *the optic flow field*. The other important feature of the natural environment are its animals. From this we decided on the kinds of objects to investigate in the experimental implementations: textured fields: both static and in motion, and animals.

Chapter 3 presented some results on the implications of modelling physical objects in an object oriented fashion. We had decided on object oriented programming since it embodied the principles of data abstraction in close coupling with the notion of a hidden local state. The hierarchical modelling of objects was shown to be vital in many scientific and engineering descriptions of objects, and of course it is familiar in computer graphics as the modelling hierarchy.

We wanted controlled access to the parts of objects which was done in such a way that the integrity of the whole object was not compromised. This allowed such objects to be incorporated into an animation system without violating the principles of information hiding but also without becoming too unwieldy. It was found that existing approaches to this problem, where it was recognized at all, had not truly grasped the essential distinction between a part-hierarchy and a type-hierarchy.

Our contribution then is to recognize this distinction and provide an extension to the normal message passing syntax of object oriented languages. This extension can be termed ‘‘forwarding of censored messages’’. It provided controlled access to the hierarchy of parts.

8.1.2 The Spatial and Temporal Priority Metrics.

The definitions of the spatial and temporal priority metrics have been refined over a number of steps in this dissertation. The final equations make sense only in their context and will not be repeated here. The purpose of this subsection is to provide the reader with a cross reference and guide to the definitions of the priority metrics.

The purpose of the spatial metric is to measure spatial detail so that the importance of (part of) an object to the image can be determined. The purpose of the temporal metric is to measure temporal detail for the same reason. The spatial metric is used in both static and dynamic scenes. The temporal metric applies only to dynamically changing images. The two measures are not assumed to be independent of one another.

The definition of the meaning of ‘‘detail’’ occupied most of §4.1. The definition involving Fourier analysis equated ‘‘highly detailed’’ with ‘‘significant energy in high frequencies’’ (§2.3 & §4.1).

The most basic definition of the spatial priority metric is simply that it measures the distance to the object. The justification that this is in fact a measure of detail is in §4.2. The argument is essentially that distance shifts the spectrum of an image towards higher spatial frequencies. The same low-pass filtering will still be done however and so more of the detail will be lost. The question of whether the radial distance to the object, or the perpendicular distance to the image plane should be measured is addressed in §4.2 and is related to the discussion of types of perspective found in §2.1.1

This spatial metric can be extended to allow for the fact that the atmosphere actually lowers the passband of the effective spatial filter applied to more distant objects (§2.2.3 & §4.1.4). The full spatial metric would combine atmospheric and eye transmission effects (§2.5.2).

The most basic definition of the temporal priority metric is that it measures the relative angular velocity of an object with respect to the viewer. The faster the object moves the more often its image has to be updated (§4.3.1). This simple idea can be extended to take into account the way the image of the object is changing as it moves (§4.3.2). The end result of this analysis is not a single number but a 2x2 transformation matrix specifying the changing shear, scale and rotation of planar facets of the image (§4.3.2).

The further generalization of the temporal metric leads to the idea that there are various orders of frame-to-frame coherence. Images are not just the same or different from one frame to

the next, they are different according to the kind of affine transformation which relates them. And if they are not exactly related by an affine transformation then the size of error can be determined (§7.2.1).

The last question which was addressed was the trade-off between the two metrics. The way movement changes the spatio-temporal spectrum of an image was discussed in §2.3.5, §2.4.7 and §4.3.3. The effect of movement is to shear the spectrum according to the direction and speed of the movement. This can cause certain frequencies to be shifted to the extent that they are filtered out. The spatial detail required from an object which is moving quickly depends on its velocity (and the orientation of the spatial frequencies — although this is less important). Using this trade-off implies an efficient way of smoothing crude images (§7.4.3).

The definitions spatial and temporal metrics have now been reviewed. Details of their application can be found in the relevant experimental chapters:

Chapter 5
: Spatial metric applied to non-continuous hierarchies.

Chapter 6
: Spatial metric applied to continuous hierarchies.

Chapter 7
: Temporal metric applied to textured planar facets.

8.1.3 A Method for Managing Complexity in Object Representations.

This subsection summarizes the methodology which was developed for modelling objects in a natural scene so that their complexity may be controlled. It draws together the experience gained in Chapters 5, 6 and 7 and depends on the theoretical work of the chapters preceding those.

The environment model is analysed in terms of the sources of spatial detail and how this spatial detail is transmitted. The perceived detail of the natural environment was characterized in Chapter 2. That chapter also tried to identify relevant detail on the basis of human visual perception.

The preferred method for analysis of detail is in terms of a (notional) Fourier analysis of spatial frequencies. The results presented on perception in the environment and on human vision in artificial viewing conditions can be applied, at least to a first approximation, as spatial and temporal filtering steps. The various rendering steps, both modelled (e.g. atmospheric haze) and real (e.g. raster display) are examined for the way they affect the detail levels. The aim is to arrive at a *number* which depends only on the synthetic camera and the modelled object.

The object is modelled in some object oriented animation system. The notion of static data and separate procedures is discarded in favour of having abstract data types. Access to the data is always mediated by procedural protocols. The secondary benefit of this is that this form of modelling is intuitively appealing in simulation and animation. In recognition of this distinction we shall now switch to calling the objects ‘‘Actors’’. All objects of the environment are Actors (also in the precise object oriented sense of belonging to a subtype of the type Actor).

The actor model is extended to include a hierarchy of detail levels. This hierarchy may be implemented as such or by any other method, probably procedural, which can provide to correct interface to the spatial priority measure.

The interaction between the Actor and the Camera (which can itself be a kind of actor) is mediated by an Appearance. This Appearance is the only necessary extra object imposed by the use of the priority metric. The Appearance is owned by an Actor, which creates one for every Camera which requests the Actor’s Appearance. The Appearance has access to the Actor’s current 3-D and 2-D (i.e. image) state. On the basis of the relative positions and movements of the Camera and the Actor (and any other relevant considerations) it must be able to calculate the Actor’s spatial and temporal priority.

Once the priority has been calculated the Appearance is submitted to the Camera for eventual rendering. If the system only produces static frames then the Appearance does not (in principle) return to the Actor, but in an animation the temporal priority of an Actor will depend on the extent to which its Appearance was fully updated in a particular frame. Thus in an animation system the Actor sends out an Appearance and receives a reply.

The benefit of the priority metrics will depend on the extent to which the calculation of priority can be done without incurring the cost of rendering.

Those are the main implications of the priority metrics for modelling objects. The implications for the Camera model were discussed in the experimental chapters.

§8.2 Future Extensions.

This research has uncovered a rich field of further application as a result of the generalization of adaptive detail in space and time. Some proposals below are extensions to work started in the experimental sections. In §8.2.1 we discuss briefly how the part hierarchy might be extended to allow for multiple logical views of an object. The next subsection, §8.2.2, discusses the extension of the discontinuous hierarchy experiment of Chapter 5 in order to use grey level displays where objects can be faded. The extension of the optic flow synthesis method is discussed in §8.2.3. If this last proposal is effective then optic flow could be applied to both animate and inanimate objects in the environment.

The most fully worked out proposal is for a topic which will be very important in the development of computer animation: parallel processing. This has not been discussed thus far in this dissertation but in many ways it is a natural application for the part hierarchy and the spatial priority metric. The abstract class of Appearances discussed in the previous subsections once more performs an important role.

The extension of the spatial detail metric in this dynamic environment might seem surprising. The metric in fact becomes more of an active/inactive indicator for each actor. A detailed design of a parallel processing animation system which deals with the same stick figures of Chapters 3 and 5 is attached as Appendix D.

8.2.1 Extensions To The Part-Whole Hierarchy: Multiple Views.

The part-whole hierarchy which was developed in Chapter 3 can be extended to allow multiple views on the same object. The Appearance of an Actor, which allows it to be rendered in a number of ways, literally provides multiple views of the same Actor. The idea is to allow multiple views of any aspect of an object.

It has been argued that using message forwarding and prototypes one can easily implement multiple representations of knowledge [LaLonde, Thomas & Pugh, 1986]. This can also be achieved with part hierarchies by extending the idea of virtual parts (§3.6.2.1). For example: consider a point defined in either Cartesian coordinates or in polar coordinates. If we wish to implement both views then the polar coordinates can always be calculated from the

Cartesian coordinates, even though only one set of attributes are stored. If the alternative views are very different then extra parts can be stored to represent them [‘‘perspectives’’, Stefik & Bobrow, 1985]. The internal mechanisms of the whole can ensure that the parts remain in correspondence.

Multiple views are used in certain kinds of knowledge representation. In §3.3.1.6 we mentioned Kay’s observer language. Philosophers [e.g., Pirsig 1974] emphasize that the division into parts is arbitrary and depends on the purpose of the analysis. It does seem that a small number of alternate views (e.g., polar vs. Cartesian coordinates) can be accommodated, but whether it is a general extensible mechanism requires further investigation.

8.2.2 Extensions For Discontinuous Detail: Fading small parts.

The stick figure implementation was done on a standard Smalltalk system. Such systems lack colour or grey levels. Recently colour versions of Smalltalk have become available [Miranda, 1987]. In this section we discuss how such a *shaded display* might be exploited.

To get some idea of how the loss of spatial detail affects a very simple model the two-dimensional Fourier transform of an image of a stick was manipulated. The higher spatial frequencies were attenuated according to some plausible model (exponential drop-off in fact) and the result transformed back into the spatial domain. The intention being merely to get some feel for what is happening.

The sequence of pictures (Figure 8.1.) show a stick which is filtered in a number of ways to reduce its higher frequencies. The stick becomes more diffuse and the boundaries ill defined. It should be borne in mind that in nature this blurring is caused by the object receding in the distance and so it should be simultaneously getting smaller. It is this blurring and decrease in size which we use with the priority metric.

Representing the various stages of such a transition is possible but expensive. Because the stick is also getting smaller and smaller it might be possible to make use of the trade-off between small detail features and intensity. That is, such a stick could also simply be slowly faded on a shaded display. We would then actually be *maintaining* the detail but decreasing the intensity of a small object. This ought to be reduce the abruptness of changes of detail as an object drops out of resolution. The degree to which an object is to be faded can depend on the priority.

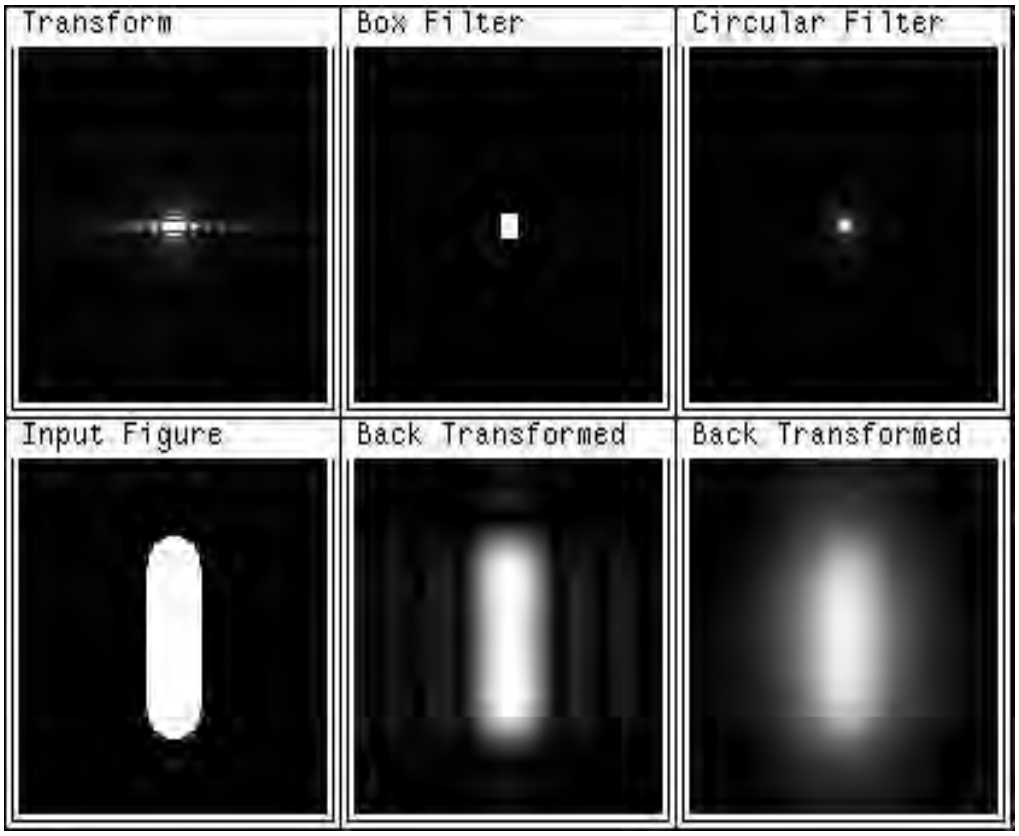


Figure 8.1. Filtering high frequencies of a ‘‘stick’’. The original shape and transform are on the left, next are filter contours and resulting filtered images. The top row is in the frequency domain and the bottom in the spatial domain.

If we had a display which lacked grey levels but which had a high frame rate it ought to be possible to create the same effect by rapidly jumping between the various levels of detail during a transition!

8.2.3 Further Work on Optic Flow Synthesis.

A major contribution of this research project has been the synthesis of three-dimensional animation by means of optic flow decomposition. In the experimental test of Chapter 7 the approach was tried on objects consisting of individual, textured, planar facets. The approach can clearly be extended to allow for objects consisting of a few connected facets, simply by treating these parts as separate objects for the purpose of object flow analysis.

Optic flow analysis can also be performed on surfaces built up out of connected, flexing, polygonal facets [Koenderink & van Doorn, 1986]. Investigating the possibilities of synthesizing optic flow for such surfaces would be a very useful extension of the work presented in Chapter 7.

Another possibility would be to use 2nd order optic flow effects. These optic flow effects completely account for the uniform motion of planar facets, and provide a finite area of convergence for quadric surfaces. Using such transformations would require efficient non-linear image transformation routines.

The points mentioned above are further extensions of the use of *local features* of the optic flow. But as we pointed out in §2.2.2 there are also *global features* of the optic flow. These are concerned with the overall patterns of the flow lines. The shape of the global optic flow field is independent of the spatial layout, it is the values which depend on the actual objects involved.

It might be possible therefore to combine the knowledge of spatial coherence of the environment with the properties of the global flow field in order to predict the way the appearance of the environment will change when the observer moves through it.

An interesting extension of the optic flow synthesis method would be to use local flow synthesis for the animals in the environment, one major class of objects we identified in §2.2, and global flow synthesis for the rest of the environment. The global flow transformations would arise due to observer motion.

The global flow field consists of piece-wise smooth regions separated by discontinuities. These discontinuities have to be identified beforehand. One major source of such discontinuities are the occluding edges in the environment.

Consider rendering the fractal landscape from Chapter 6 as it changes due to camera motion. We would need to identify the occluding edges to some level of detail and find some method of ‘‘stitching’’ together the piece-wise smooth regions thereby detected.

To detect the occluding edges is simple: a popular method for hidden surface removal on height fields is the floating horizon method (§6.2.2). This method proceeds by identifying the current occluding edge while rendering the landscape, starting from the viewer and proceeding into the distance. It would seem ideally suited for optic flow synthesis using global effects.

The animation of observer motion over a height field could begin by rendering the scene as usual from the three-dimensional data. A record would have to be kept of the connected regions bounded by occluding edges. The next frames could then apply the optic flow

distortion to these regions. Some sort of error measure would eventually force a return to full three-dimensional rendering.

The difficulty with this method is dealing with occluding edges once a few optic flow steps have occurred: exactly how do the edges get joined up? Another problem might be that the fractal surfaces are so rough that there are very many small areas of coherence rather than a few large ones. In this case the horizons could be calculated at lower resolution levels. Clearly the application of global optic flow synthesis requires further work.

Appendix A

Implementing a Part-Whole Hierarchy in Smalltalk.

§A.1 Overview.

We assume that the Smalltalk has been extended to allow multiple inheritance [Borning & Ingalls, 1982]. This provides the changes to the parser which allows compound messages and the basic modification to the class Object's instance method for the message "doesNotUnderstand:". This sends a message to the class of the receiver with the selector "tryCopyingCodeFor:".

The class method "tryCopyingCodeFor:" is the one we override in the super (meta) class of all objects which are to have parts. If the message sent was not compound or was sent as part of the multiple inheritance implementation we treat it as before. If it is a compound selector and the prefix starts with a lower case letter (excluding "super" and "all") we compile a message forwarder.

The message forwarder is simply a method which has as selector the complete compound message with all the keywords and arguments. The method sends the prefix to self and then the rest of the compound message. The answer of the method is that result. For example:

To forward:
joe leftLeg.lowerLeg.foot
the method is:
↑self leftLeg lowerLeg.foot

These methods are synthesized and compiled without saving the source code. They are classified as 'message forwarders' for access in the browser, this is not essential but then forwarders are invisible to the programmer. The appropriate class methods are "compileUnchecked:" and "organization classify:under:".

The remaining class methods are equally simple. These are methods to compile code to access parts (instance variables) by name automatically. In this case it is useful to save the source code, since fast compilation is no longer important. A dictionary which associates instance variable names with default classes should also be created. This dictionary can be

used when parts belonging to a whole have to be instantiated.

Appendix B

Timings of the Continuous Spatial Metric Experiment.

§B.1 Introduction.

This appendix contains the times taken to render a particular fractal at three image sizes: 256x256, 512x512 and 1024x1024. The times are given at variable facet sizes in the adaptive detail case or else for various fixed recursion depths. With adaptive detail the facet size and the maximum recursion depth is given, otherwise “fix” is placed in the facet size column and the entry in the recursion depth column is the fixed recursion depth. The “Levels generated” refers to the data generated by the midpoint subdivision method, the upper levels of the fractal having been synthesized by the 1/f Fourier filtering technique.

Batched rendering meant that the image was generated in memory and displayed only when it was complete. Non-batched rendering meant that the image was rendered (incrementally) whenever the area generated exceded some threshold. The associated overhead was only significant at low resolutions. The times are cpu seconds for user and system time. These results are summarized and analysed in §6.4.

§B.2 Image Size: 256x256.

Facet size	Recursion depth	Levels generated	Batched rendering	Times (sec)		Ref. number
				User	System	
1	11	3	n	527.38	14.88	17
2	10	2	n	159.00	10.68	16
4	9	1	y	45.22	3.84	20
4	9	1	n	45.38	4.18	15
8	8	-	y	12.86	1.74	19
8	8	-	n	15.88	1.88	14
16	7	-	y	3.60	0.92	18
16	7	-	n	4.66	0.62	13

Facet size	Recursion depth	Levels generated	Batched rendering	Times (sec) User	System	Ref. number
fix	7	-	y	28.98	2.84	50
fix	8	-	y	102.32	11.68	47
fix	9	1	y	318.40	10.56	44
fix	9	1	n	318.38	11.16	43
fix	9	1	n	346.28	11.96†	42
fix	10	2	y	1156.56	17.82	39
fix	11	3	n	4250.82	27.74	36
fix	12	4	n	15668.00	64.70	31

§B.3 Image Size: 512x512.

Facet size	Recursion depth	Levels generated	Batched rendering	Times (sec) User	System	Ref. number
1	12	4	y	2072.22	13.04	94
1	12	4	y	2105.84	15.76	93
1	12	4	y	2124.70	20.96	55
2	11	3	y	615.38	12.90	92
2	11	3	y	626.42	15.58	56
2	11	3	y	640.56	15.36	90
2	11	3	y	655.80	16.22	91
2	11	3	n	592.48	10.90	12
3	11	3	y	278.04	11.82	88
3	11	3	y	283.32	13.14	87
3	11	3	y	287.26	13.40	89
4	10	2	y	171.02	9.78	86
4	10	2	y	171.60	8.58	57
4	10	2	y	174.96	10.24	85
4	10	2	y	178.32	9.94	84
4	10	2	y	179.36	8.42	29

† Compiled code was not optimized.

Appendix B — Timings of the Continuous Spatial Metric Experiment

Facet size	Recursion depth	Levels generated	Batched rendering	Times (sec) User	Times (sec) System	Ref. number
4	10	2	n	169.08	8.40	11
6	10	2	y	76.80	5.98	83
6	10	2	y	77.40	6.20	81
6	10	2	y	82.32	6.40	82
8	9	1	y	47.62	3.78	23
8	9	1	y	47.78	3.72	58
8	9	1	y	49.12	4.30	79
8	9	1	y	49.92	4.26	78
8	9	1	y	50.62	3.92	80
8	9	1	y	51.04	3.72	28
8	9	1	n	49.50	4.00	10
10	9	1	y	32.62	3.32	75
10	9	1	y	33.16	3.16	77
10	9	1	y	33.18	3.10	76
12	9	1	y	23.04	2.60	72
12	9	1	y	23.66	2.78	73
12	9	1	y	23.70	3.18	74
16	8	-	y	14.00	1.76	22
16	8	-	y	15.06	2.00	70
16	8	-	y	15.22	1.92	71
16	8	-	y	15.40	1.76	69
16	8	-	y	15.50	1.64	59
16	8	-	y	15.34	1.82	27
16	8	-	n	17.90	1.52	9
20	8	-	y	9.48	1.30	67
20	8	-	y	9.50	1.02	68
20	8	-	y	10.68	1.24	66
24	8	-	y	6.74	0.96	63
24	8	-	y	7.02	0.70	64
24	8	-	y	7.10	1.04	65
32	7	-	y	4.46	0.72	61

Facet size	Recursion depth	Levels generated	Batched rendering	Times (sec) User	Times (sec) System	Ref. number
32	7	-	y	4.48	0.68	62
32	7	-	y	4.64	0.52	60
32	7	-	y	4.66	0.38	21
32	7	-	n	7.22	0.82	4
fix	7	-	y	30.30	2.88	49
fix	8	-	y	113.74	12.32	46
fix	9	1	y	402.62	14.74	41
fix	10	2	y	1303.32	15.94	38
fix	11	3	n	4487.90	31.08	35
fix	12	4	n	15978.60	47.70	33

§B.4 Image Size: 1024x1024.

Facet size	Recursion depth	Levels generated	Batched rendering	Times (sec) User	Times (sec) System	Ref. number
1	14	6	n	7926.30	14.04	51
2	12	4	y	2353.26	16.54	52
4	11	3	y	680.26	14.40	53
4	11	3	n	640.54	11.14	8
8	10	2	y	192.28	9.72	54
8	10	2	n	181.96	9.42	7
8	10	2	n	182.80	8.66	3
16	9	1	y	52.04	3.50	26
16	9	1	n	53.72	3.98	?
16	9	1	n	55.48	3.66	2
32	8	-	y	16.04	1.66	25
32	8	-	n	18.98	2.04	1
32	8	-	n	19.26	1.94	5
64	7	-	y	5.64	0.52	24
fix	7	-	y	33.84	2.80	48

Appendix B — Timings of the Continuous Spatial Metric Experiment

Facet size	Recursion depth	Levels generated	Batched rendering	Times (sec) User	Times (sec) System	Ref. number
fix	8	-	y	118.02	11.40	45
fix	9	1	y	426.82	13.42	40
fix	10	2	y	1526.06	18.90	37
fix	11	3	n	4826.40	12.96	34
fix	12	4	n	16925.52	51.98	32
fix	13	5	n	62538.24	125.58	30

Appendix C

Timings of the Dynamic Metric Experiment.

§C.1 Introduction.

Timing the performance of the animation system is complicated by the fact that there are so many options possible. The timings presented here are intended mainly to illustrate the effects of the various adaptive operations which take place. The same basic movement is executed for various rendering options.

Even so there are many variables which have to be held fixed. In all of the cases below only one actor was active at a time. The same action was performed for the full time of the test.

These results are summarized and analysed in §7.4. The parameters which were varied were:

1. The actor.
2. The 2-D transformation and rendering algorithm.
3. The transformation applied.

§C.2 Timings.

To avoid clutter in the table of timings blank entries are used in the first five columns to indicate parameters which are unchanged from the previous row. The columns are:

Actor

Symbol	Actors.	Basic Size.
bc	Butterfly cut-out	224x184
bf	Butterfly in square frame	250x200
fd	Flow diagram drawing	75x74

Appendix C — Timings of the Dynamic Metric Experiment.

Rendering Option

Symbol	Rendering Option
Q	“Quick” — one pass image transformation.
QP	“Pretty” — one pass but with simple averaging for size decrease
P	“Perfect” — two pass anti-aliased, only one shear implemented

Transformation

Symbol	Transformation Matrix
o	none, pure translation, $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
a	$\begin{pmatrix} 1.01 & 0.01 \\ -0.01 & 1 \end{pmatrix}$
b	$\begin{pmatrix} 1 & 0.01 \\ 0 & 1 \end{pmatrix}$
c	$\begin{pmatrix} 1 & -0.01 \\ 0.01 & 1.01 \end{pmatrix}$
d	$\begin{pmatrix} 1.01 & 0.01 \\ 0 & 1 \end{pmatrix}$
e	$\begin{pmatrix} 0.99 & 0 \\ 0.01 & 1 \end{pmatrix}$

Initial — The initial field indicates when a particular set of timings incurred the system initialization overhead, this happens the first time a camera is activated.

Frame Count — The number of frames produced during the run.

Times — The total time (in cpu seconds ≈ elapsed time) for the frames as well as the average time for a single frame. The avarage is taken over all comparable timings.

Frame Rate — The number of frames produced per second on average.

Reference Number — A number which uniquely identifies the experiment.

Appendix C — Timings of the Dynamic Metric Experiment.

Actor	Render option	Trans-form	Initial	Frame count	Times (sec) Total	Average	Frame Rate	Ref. number
bc	Q	o	y	300	20.42	.0681	14.7	1
			n		16.82			2
					17.30			3
					17.32			4
					17.32	.0573	17.5	5
		a	y		107.58	.3586	2.8	10
		a			105.04			6
					105.84			7
					105.50			8
					105.72			9
			n		105.74			11
					105.50			12
					105.56	.3519	2.8	13
		o	y	100	8.86	.0886	11.3	14
			n		6.12			15
					6.10			16
					6.18			17
					6.08			18
					6.12	.0612	16.3	19
		a	y		33.00	.3300	3.0	20
			n		32.58			21
					32.54			22
					32.54			23
					32.58	.3256	3.1	24
	P	o	y		8.94	.0894	11.2	25
			n		6.40			26
					6.14			27
					6.08	.0621	16.1	28
	P	b	y		257.42	2.57	0.39	29
					263.50			30

Appendix C — Timings of the Dynamic Metric Experiment.

Actor	Render option	Trans-form	Initial	Frame count	Times (sec) Total	Average	Frame Rate	Ref. number
					263.74	2.64	0.38	31
		c			209.02	2.09	0.48	32
		d			256.80			33
					256.82			34
					263.64	2.59	0.39	35
bf	Q	o	y	300	29.74	.0991	10.1	36
			n		25.94			37
					25.82			38
					25.96			39
					25.88	.0863	11.6	40
		d			104.36			41
					104.32			42
					104.44			43
					104.28	.3478	2.9	44
		e			36.76			45
					37.16			46
					37.06	.1233	8.1	47
	QP				37.42			48
					37.20			49
					37.18			50
					37.10	.1241	8.1	51
fd	Q	o	y		5.18	.0173	57.9	52
			n		4.78			53
					4.76			54
					4.88	.0160	62.4	55
		b			9.06			56
					9.10			57
					9.10			58
					8.94			59
					8.98			60

Appendix C — Timings of the Dynamic Metric Experiment.

Actor	Render option	Trans-form	Initial	Frame count	Times (sec) Total	Times (sec) Average	Frame Rate	Ref. number
					9.16			61
					9.04			62
					8.84			63
					9.02			64
					8.94	.0301	33.3	65
	P				97.04			66
					97.52			67
					97.50			68
					97.62	.3247	3.1	69

Appendix D

Concurrent Object Oriented Animation.

§D.1 Concurrent Object Oriented Animation: A Research Proposal.

In computer animation the object oriented approach has already been shown to be an excellent way of *dealing with the complexity* of programming and modelling such large simulation systems. Moreover it appears to be naturally conducive to a *parallel processing* implementation which should lead to a much needed *increase in processing speed*.

We give the functional design of an experimental system for animating stick figures and cameras modelled as independent processes. Each part of a figure is represented by an independent actor (or process-object). Attention is given to coordinating and controlling the animation: in this area further research is particularly required.

D.1.1 Part Hierarchies and Parallel Processing.

The assembly of whole objects from their parts was extensively discussed in Chapter 3. For our present purposes we refer to the part-whole hierarchy as a general abstraction for incorporating the notion of "an object made up of active parts which are related via constraints." (§3.2).

Events in nature happen at the same time. Simulating such an environment requires concurrent execution of the objects representing elements of this environment, at least in principle. Thus, provided we have a formalism which accommodates it, concurrency is the *natural and simple* way of describing animation (or any other kind of simulation).

Part hierarchies provide better control over access to the parts than is found in many object oriented languages. This controlled access to names *provides a way of breaking up a a single global name space into separate domains* which is very useful in allowing different processes to function independently in a concurrent environment. There is only a single universal name, the "world". All actors are part of the world. All other parts are named by symbols which refer to their relative position in some part-whole hierarchy.

§D.2 Controlling Figure Animation.

There are two aspects to modelling animated figures: producing computer *representations* which allow movement and *controlling* that movement. The first problem has already been addressed in Chapters 3 and 5, which were concerned with the appearance of moving figures at various levels of detail. The problem of controlling figure movements is related to the problem of modelling motion subject to various constraints.

There are two principal ways of controlling three-dimensional animated figures:

Firstly, we can choose to drive the animation from an external script which specifies every movement explicitly.

Secondly, we can indicate constraints which apply to the objects and the provide a broad outline of the configurations required and then depend on the system to provide the complete configuration.

With the *first approach* the animator has complete control and freedom at the expense of having to be concerned with every detail of the animation. The essential feature from the programming point of view is that there is *no feedback* between elements of the modelled environment: control is strictly top down. The central task of such an animation system is to provide access to all parts of the environment and to determine when all the processes in the simulation have completed there tasks. This access is used to distribute instructions to the objects and to inform them of clock ticks. The objects are also asked to return their relative positions when the rendering process traverses the modelled environment and when the user attempts to pick a displayed object for some input operation.

In the *second approach* we are conducting a much more realistic simulation of some imaginary world. The animator is freed from having to concentrate on mundane tasks at the expense of having to learn about a somewhat more complicated system and not having the same total freedom to manipulate objects. From the programming point of view there is *mutual feedback between objects* in the environment. In addition to the operations specified above, some mechanism must exist to detect interactions between objects (collisions or relatively constrained movements). As a result of such an interaction objects may need to communicate with other objects in order to adjust their configurations. Since any object can set off a new spurt of activity in the system it is somewhat more difficult to detect termination of a frame generation cycle.

§D.3 A System for Concurrent Figure Animation without Feedback.

The existing design of a part hierarchy for animated figures can readily be mapped onto a parallel processing paradigm. The priority metric which governs the levels detail visible to a particular camera can effectively be used to limit the number of process activations.

We construct a stick figure as before out of a hierarchy of parts. Each part can be accessed only via the whole of which it forms a part. The only difference is that each part now executes as an independent process. The part-whole hierarchy thus becomes a useful addressing hierarchy for accessing the processes.

An important object which every Actor and the world possesses is an instance of a kind of *Appearance* (see §8.1.3). There is a Camera which renders the world and its elements. Each Camera has a buffered input queue over which the Actors send their Appearances to the Camera. The world also has a clock which synchronizes all Actors and the Cameras at every time step.

Each Actor has to be able to accept instructions which it executes at each clock tick. These instructions apply for a variable number of ticks. In the simplest case new instructions are issued at every tick.

The Actors concurrently execute their instructions for a particular time step. The world (or clock) has to detect when all Actors have completed their actions and are ready for the next tick. Before another tick can be sent the scene has to be rendered by any Cameras which are active.

Further details of this approach is found in subsection below, which describes how the Smalltalk implementation handles the clock cycle.

D.3.1 A Concurrent Smalltalk Animation System.

In order to build a concurrent Smalltalk animation system the language has to be extended so that the objects become more like Actors. This can be done by placing objects in separate processes and then surrounding these processes by encapsulators which forward messages and wait for replies from these processes.

If the parts which make up a figure are encased in encapsulators which turn them into actor-objects then the part hierarchy (§3.4) becomes a process naming hierarchy. The processes do

not have global names nor are pointers needed. Instead symbolic paths can name the processes.

This does not apply to the “standard” Smalltalk objects like numbers. However since these are relatively unchanging or else independent from processor to processor it should be possible to implement the global name space for these objects via local copies for each processor.

Animation Clock Cycle.

The control of the concurrently executing Actors and their coordination with the rendering process (or synthetic camera) is a prime problem. In the case of Actors operating without mutual feedback the following “algorithm” is proposed:

Let each top-level Actor be registered with the world object (the parts of the Actors are not directly visible to the world). Each Camera is also registered with the world. We assume that at least one Camera is active.

At each clock tick the world distributes the tick to each top-level Actor and the Cameras. The Actors distribute the tick instruction to all their parts before proceeding with their own activities for the particular time step. This distribution naturally terminates at the bottom of the part hierarchy. Once its own actions are complete an Actor blocks on an unbuffered channel awaiting instructions or the next tick from its owner.

The world requests the relative position (transformation) of each Camera and the name of buffered queue on which it will be receiving the Appearances of the various Actors. The world informs each Camera of the number of Actors at the top level and passes its own Appearance which has been transformed to reflect the relative position of the Camera.

The transformation and queue name as well as level number is then passed to each Actor. The Actor concatenates its own coordinate transformation to the transformation it receives. The Actor first tells the Camera its level number and the effective number of parts at that level. The new transformation, queue name and an incremented level number is then passed on to those parts.

This rendering traversal stops when there are no more effective dependent parts. For a particular Camera an Actor can decide that it has no more parts which would be visible due to the distance of the Camera (the *spatial priority metric*). Such an Actor would then assume its “leaf appearance” rather than the normal appearance which it has when it is a node in a

continuing hierarchy. The Actor then transforms its own Appearance for the Camera and passes that on to the buffered queue.

The world now blocks awaiting signals from all the Cameras to indicate completion of the rendering process. Each Camera keeps track of the number of Actors at each hierarchical level. It accepts Appearances as they come and updates the total number of Actors at each level and the number of Appearances so far received at each level. The Camera can therefore detect when it has rendered all possible Appearances for all levels.

The Camera has to scan convert the Appearances and perform hidden surface removal, presumably in some concurrent fashion. If the Camera used some kind of parallel depth-buffer where different processes dealt with different depths then all that is required is that several queues be handed out which are labeled with the range of depths they accept. The Appearances will know which queues to enter.

The world deals with user interaction. If necessary it passes the interaction instructions on to the other parts of the animation. Once all Cameras have signalled completion and all instructions have been sent, the world can distribute the next tick and so a new cycle will begin.

§D.4 A System for Concurrent Figure Animation with Feedback.

As pointed out before, providing the animator with ways of specifying movement at a level where each joint coordinate is not explicitly provided for modification is an area of active research. The basic requirements for providing such a facility is (1) a knowledge base for dealing with eventualities which occur, (2) some way of sensing interactions between objects and (3) a way of allowing objects to change their behaviour based on the implications of what has been sensed. That is, we need *intelligent internal feedback*.

When we are running multiple independent processors it is also important to be able to detect when the parts of the scene have completed their interactions and can be rendered on the display. The complication being of course that with feedback between elements of the environment a single object which has not terminated can trigger off all the other processes into further action.

The way to realize such a high level of control is build a constraint based animation system [Borning, 1979 & Nicol, 1984]. Such constraint based systems lend themselves to an almost

declarative style of programming: the user is concerned with stating the (geometric) relations which develop over time and the system sees to it that these do obtain.

The part-whole hierarchy enables us to build a system of constraints between Actors and to detect when processing has terminated. Recall that messages to parts can be intercepted by owning objects. These messages contain the names of the parts all the way down a part hierarchy. A constraint is then a method which intercepts messages referring to a certain sequence of part names [called paths - Borning, 1979]. The parts are therefore *referred to symbolically*. There is no need for pointers (or absolute references) to be handed out. Maintain absolute names or pointers is very difficult with independent concurrent processes.

To detect termination the following algorithm is suggested:

> Whenever a an extremity (a leaf of the part hierarchy) becomes quiescent it informs its owning object that it is now “locked”. When all the parts of non-leaf parts become quiescent and the part itself has completed processing the whole subtree can be declared locked. This locked state propogates upward in the part hierarchy.
>
> Any message to any of the lower nodes must come via an active part higher up and as the message goes down to the destination part it can unlock the parts of the subtree.
>
> Once all the root Actors in the world object are locked then the processing for that particular time step must be complete and rendering can begin. As soon as a part has been rendered it can again begin sending and receiving messages and so proceed to the next time step.

§D.5 Conclusion.

The basic design of a concurrent animation system without internal feedback between the elements of the animation has been presented. The approach adopted simplifies programming complexity while at the same time speeding up the computer processing by the use of concurrency.

The novel features of the design included the use of part hierarchies to manage the name space of the concurrent processes. This allows access to any process without requiring a global name space. Another new feature is the use of the *spatial priority metric* to reduce the number of process activations. The detection of termination was assisted by the same

general abstraction used to implement the system of priority metrics, namely the Appearance type.

The approach also seems to be extensible to the more complex computer animation problems which include feedback and constraints between Actors.

Bibliography

Abramsky, S. (1984) **in:** Chambers, F.B., Duce, D.A. & Jones, G.P. **(ed)** *Distributed Computing.* Academic Press, London. 307-327
''Reasoning about concurrent systems.''

Ackerman, J.S. (1978) *Journal of the Warburg and Courtauld Institutes* **41** 108-146 & plates 16-17
''Leonardo's eye.''

Adelson, E.H. & Bergen, J.R. (1985) *J.Opt.Soc.Am.A* **2**, 2 284-299
''Spatiotemporal energy models for the perception of motion.''

Adobe Systems (1985) *PostScript Language Reference Manual.* Addison-Wesley, Reading, MA.

Agha, G. (1986) *SIGPLAN Notices* **21**, 10 58-67
''An overview of actor languages.''

Ahuja, N. & Nash, C. (1984) *Computer Vision, Graphics & Im.Proc.* **26** 207-216
''Octree representations of moving objects.''

Amanatides, J. (1987) *IEEE Computer Graphics & Applications* **7**, 1 44-56
''Realism in computer graphics: A survey.''

Anderson, D.P. (1982) *ACM Trans.Graphics* **1**, 4 274-288
''Hidden line elimination in projected grid surfaces.''

Anstis, S.M. (1978) **in:** Held, R., Leibowitz, H.W. & Tueber, H. **(ed)** *Handbook of Sensory Physiology.* Vol.**VIII** *Perception.* Springer, Berlin. 655-673
''Apparent movement.''

Aono, M. & Kunii, T.L. (1984) *IEEE Computer Graphics & Applications* **4**, 5 10-34
''Botanical tree image generation.''

Arya, K. (1986) *Computer Graphics Forum* **5**, 4 297-311
''A functional approach to animation.''

Badler, N.I. (1987) *IEEE Computer Graphics & Applications* **7**, 6 10-11
''Articulated figure animation: Guest editor's introduction.''

Badler, N.I. & Carlbom, I. (1984) *Eurographics'84.* North-Holland, Amsterdam. 185-200
''The computer graphics scene in the U.S.''

Badler, N.I. & Smoliar, S.W. (1979) *Computing Surveys* **11**, 1 19-38
''Digital representations of human movement.''

Bergman, L., Fuchs, H., Grant, E. & Spach, S. (1986) *SIGGRAPH'86: Computer Graphics* **20**, 4 29-37
''Image rendering by adaptive refinement.''

Berry, M.V. & Lewis, Z.V. (1980) *Proc.R.Soc.Lond.A* **370** 459-484
''On the Weierstrass—Mandelbrot fractal function.''

Beyer, T. & Friedell, M. (1987) *Eurographics'87.* Elsevier, Amsterdam. 151-158
''Generative scene modelling.''

Borning, A.H. (1979) *Xerox Palo Alto Research Center report SSL-79-3*
''ThingLab: A constraint-oriented simulation laboratory.''
a revised version of: Stanford University PhD. thesis, Stanford Computer Science Department Report STAN-CS-79-746

Borning, A.H. (1981) *ACM Trans.Programming Languages and Systems* **3**, 4 353-387
''The programming language aspects of ThingLab, a constraint-oriented simulation laboratory.''

Borning, A.H. (1986) *IEEE/ACM Fall Joint Computer Conf.* 36-40
''Classes versus prototypes in object-oriented languages.''
Dallas, Texas, Nov 1986.

Borning, A.H. & Ingalls, D.H.H. (1982) *Proc.Nat.Conf.Artificial Intelligence* 234-237
''Multiple inheritance in Smalltalk-80.''
Pittsburgh, PA.

Bouville, C. (1985) *SIGGRAPH'85: Computer Graphics* **19**, 3 45-52
''Bounding ellipsiods for ray-fractal intersection.''

Braccini, C. & Marino, G. (1980) *Computer Graphics & Im.Proc.* **13** 127-141
''Fast geometrical manipulations of digital images.''

Bracewell, R.N. (1978) *The Fourier Transform and Its Applications.* McGraw-Hill, New York. (2nd edition)

Brachman, R.J. (1983) *Computer* **16**, 10 30-36
''What IS-A is and isn't: an analysis of taxonomic links in semantic networks.''

Bradick, O. (1974) *Vision Research* **14** 519-528
''A short-range process in apparent motion.''

Bradick, O., Campbell, F.W. & Atkinson, J. (1978) **in:** Held, R., Leibowitz, H.W. & Tueber, H. **(ed)** *Handbook of Sensory Physiology.* Vol.**VIII** *Perception.* Springer, Berlin. 3-38
''Channels in vision: basic aspects.''

Brelstaff, G.J. (1984) *M.Sc. Project Report* Dept. Computer Science, Univ. Edinburgh, Edinburgh.
''Surface texture for object models using fractional Brown functions.''

Brooks, F.P. (1987) **in:** Crow, F. & Pizer, S.M. **(ed)** *Proceedings of 1986 Workshop on Interactive 3D Graphics.* ACM, New York. 9-21
''Walkthrough — A dynamic graphics system for simulating virtual buildings.''
(Chapel Hill, NC, Oct 1986)

Brooks, R.A. (1981) *Artificial Intelligence* **17** 285-348
''Symbolic reasoning among 3-D models and 2-D images.''

Burtnyk, N. & Wein, M. (1976) *Comm.ACM* **19**, 10 564-569
''Interactive skeleton techniques for enhancing motion dynamics in key frame animation.''

Buxton, B.F. (1984) *Computer Vision — Image Formation.* Lectures at Queen Mary College, London.

Buxton, B.F. & Buxton, H. (1983) *Proc.R.Soc.Lond.B* **218** 27-47
''Monocular depth perception from optic flow by space time signal processing.''

Campbell, F.W. & Robson, J.G. (1968) *J.Physiol.(Lond.)* **197** 551-556
''Applications of Fourier analysis to the visibility of gratings.''

Cardelli, L. (1984) **in:** Kahn, MacQueen & Plotkin **(ed)** *Lecture Notes in Computer Science, no 173. Semantics of Data Types.* Springer Verlag. 51-67
''A semantics of multiple inheritance.''

Cardelli, L. & Wegner, P. (1985) *Computing Surveys* **17**, 4 471-522
''On understanding types, data abstraction, and polymorphism.''

Carlbom, I. & Paciorek, J. (1978) *Computing Surveys* **10**, 4 465-502
''Planar geometric projections and viewing transformations.''

Catmull, E. & Smith, A.R. (1980) *SIGGRAPH'80: Computer Graphics* **14**, 3 279-285
''3-D transformations of images in scanline order.''

Clark, J.H. (1976) *Comm.ACM* **19**, 10 547-554
''Hierarchical geometric models for visible surface algorithms.''

Coleridge, S.T. (1817) *Biographia Literaria.* Chapter 14 (See e.g. Norton Anthology of English Literature, Norton, New York, 1975)

Cook, R.L. (1986) *ACM Trans.Graphics* **5**, 1 51-72
''Stochastic sampling in computer graphics.''

Cook, R.L., Carpenter, L. & Catmull, E. (1987) *SIGGRAPH'87: Computer Graphics* **21**, 4 95-102
''The Reyes image rendering architecture.''

Cook, R.L., Porter, T. & Carpenter, L. (1984) *SIGGRAPH'84: Computer Graphics* **18**, 3 137-145
''Distributed ray tracing.''

Coquillart, S. & Gangnet, M. (1984) *IEEE Computer Graphics & Applications* **4**, 7 35-42
''Shaded display of digital maps.''

Crocker, G.A. (1987) *IEEE Computer Graphics & Applications* **7**, 9 10-17
''Screen-area coherence for interactive scanline display algorithms.''

Crow, F.C. (1984) *SIGGRAPH'84: Computer Graphics* **18**, 3 207-212
''Summed-area tables for texture mapping.''

Darlington, J. (1984) **in:** Chambers, F.B., Duce, D.A. & Jones, G.P. **(ed)** *Distributed Computing.* Academic Press, London. 57-77
''Functional programming.''

Daugman, J.G. (1985) *J.Opt.Soc.Am.A* **2**, 7 1160-1169
''Uncertainty releation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters.''

Densmore, O.M. & Rosenthal, D.S.H. (1987) *Computer Graphics Forum* **6**, 3 171-180
''A user-interface toolkit in object-oriented PostScript.''

Dippé, M.A.Z. & Wold, E.H. (1985) *SIGGRAPH'85: Computer Graphics* **19**, 3 69-78
''Antialiasing through stochastic sampling.''

Duffieux, P.M. (1983) *The Fourier Transform And Its Applications to Optics.* Wiley, New York. (2nd edition).
Originally published as: ''L'intégrale de Fourier et ses applications è l'optique.'' Mason, Paris, 1970

Eklundh, J.O. & Kjelldahl, L. (1985) *Computers & Graphics* **9** 339-349
''Computer graphics and computer vision — some unifying and discriminating features.''

Fant, K.M. (1986) *IEEE Computer Graphics & Applications* **6**, 1 71-80
''A nonaliasing, real-time spatial transform technique.''

Fikes, R. & Kehler, T. (1985) *Comm.ACM* **28** 904-920
''The role of frame-based representation in reasoning.''

Fiorentini, A. (1972) **in:** Jameson, D. & Hurvich L.M. **(ed)** *Handbook of Sensory Physiology.* Vol.**VII/4** *Visual Psychophysics.* Springer, Berlin. 188-201
''Mach band phenomena.''

Fisher, S.S., McGeevy, M., Humphries, J. & Robinett, W. (1987) **in:** Crow, F. & Pizer, S.M. **(ed)** *Proceedings of 1986 Workshop on Interactive 3D Graphics.* ACM, New York. 77-87
''Virtual environment display system.''
(Chapel Hill, NC, Oct 1986)

Foley, J.D. & van Dam, A. (1982) *Fundamentals of Interactive Computer Graphics.* Addison-Wesley, Reading, Massachusetts.

Forrest, A.R. (1985) **in:** Earnshaw, R.A. **(ed)** *NATO ASI Series. Fundamental Algorithms for Computer Graphics.* Springer, Berlin. 113-134
''Antialiasing in practice.''

Fournier, A., Fussell, D. & Carpenter, L. (1982) *Comm.ACM* **25**, 6 371-384
''Computer rendering of stochastic models.''
Comments in Comm.ACM vol. 25 no. 8 pp. 583-584

Fredkin, E. (1960) *Comm.ACM* **3**, 9 490-499
''Trie memory.''

Fuchs, H., Abram, G.D. & Grant, E.D. (1983) *SIGGRAPH'83: Computer Graphics* **17**, 3 65-72
''Near real-time shaded display of rigid objects.''

Fuchs, H., Kedem, Z.M. & Naylor, B. (1980) *SIGGRAPH'80: Computer Graphics* **14**, 3 124-133
''On visible surface generation by *a priori tree* structures.''

Fujimoto, A., Tanaka, T. & Iwata, K. (1986) *IEEE Computer Graphics & Applications* **6**, 4 16-26
''ARTS: accelerated ray-tracing system.''

Gabor, D. (1946) *J.Inst.Elect.Eng* **93** 429-457
''Theory of communication.''

Gibson, J.J. (1979) *The Ecological Approach to Visual Perception.* Houghton Mifflin co, Boston.

Glassner, A.S. (1984) *IEEE Computer Graphics & Applications* **4**, 10 15-22
''Space subdivision for fast ray tracing.''

Glassner, A.S. (1986) *SIGGRAPH'86: Computer Graphics* **20**, 4 297-306
''Adaptive precision in texture mapping.''

Goldberg, A. & Robson, D. (1983) *Smalltalk-80: the language and its implementation.* Addison-Wesley, Reading, Massachusetts.

Goldsmith, J. & Salmon, J. (1987) *IEEE Computer Graphics & Applications* **7**, 5 14-20
''Automatic creation of object hierarchies for ray tracing.''

Goldstein, H. (1980) *Classical Mechanics.* Addison-Wesley, Reading, MA. (2nd edition)

Gordon, J.I. & Johnson, R.W. (1984) *Applied Optics* **23** 3363-3372
''Equilibrium radiance model applications and comparisons to atmospheric measurements and Rayleigh models.''

Gorraiz, J. & Horvath, H. (1983) *Applied Optics* **22** 3684-3688
''Influence of nonuniform ground reflectance on horizontal visibility.''

Greenberg, D.P. (1988) *Comm.ACM* **31** 123-129,151
''Coons award lecture.''

Haber, R.N. (1983) **in:** Beck, J., Hope, B. & Rozenfeld, A. **(ed)** *Human and Machine Vision.* Academic Press, New York. 157-235
''Stimulus information and processing mechanisms in visual space perception.''

Hall, E.L. (1979) *Computer Image Processing and Recognition.* Academic Press, New York.

Halstead, R.H. (1985) *ACM Trans.Programming Languages and Systems* **7**, 4 501-538
''Multilisp: A language for concurrent symbolic computation.''

Hamilton, W.R. (1969) Vol.**1** *Elements of Quaternions.* Chelsea Publishing Co, New York. (3rd edition)

Haruyama, S. & Barsky, B.A. (1984) *IEEE Computer Graphics & Applications* **4**, 3 7-19
''Using stochastic modeling for texture generation.''

Heckbert, P.S. (1986) *IEEE Computer Graphics & Applications* **6**, 11 56-67
''Survey of texture mapping.''

Hegron, G. (1987) *Eurographics'87.* Elsevier, Amsterdam. 529-542
''Dynamic management of 3D scenes.''

Henderson, P. (1982) *Symposium on Lisp & Functional Programming.* 9pp
''Functional geometry.''

Hubel, D.G. & Wiesel, T.N. (1962) *J.Physiol.(Lond.)* **160** 106-154
''Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex.''

Hubschman, H. & Zucker, S.W. (1982) *ACM Trans.Graphics* **1**, 2 129-162
''Frame-to-frame coherence and the hidden surface computation: Constraints for a convex world.''
also in SIGGRAPH'81

Hunter, G.M. & Steiglitz, K. (1979a) *Computer Graphics & Im.Proc.* **10** 289-296
''Linear transformation of pictures represented by quad trees.''

Hunter, G.M. & Steiglitz, K. (1979b) *IEEE Trans.Pat.Anal.Mach.Intel.* **1**, 2 145-153
''Operations on images using quad trees.''

Ingalls, D.H.H. (1986) *OOPSLA'86: SIGPLAN Notices* **21**, 11 347-349
''A simple technique for handling multiple polymorphism.''

Jackins, C.L. & Tanimoto, S.L. (1980) *Computer Graphics & Im.Proc.* **14** 249-270
''Oct-trees and their use in representing three-dimensional objects.''

Jackson, T. (1987) *Graphics and Image Processing Algorithms for the miniDAP.* Dept. Computer Science & Statistics, Queen Mary College, London University, London. BSc. Project Report

Jansen, F.W. (1986) **in:** L.R.A. Kessener, Peter, F.J. & van Lierop, M.L.P. **(ed)** *Data Structures for Raster Graphics.* Springer-Verlag, Berlin. 57-73
''Data structures for ray tracing.''

Johansson, G. (1978) **in:** Held, R., Leibowitz, H.W. & Tueber, H. **(ed)** *Handbook of Sensory Physiology.* Vol.**VIII** *Perception.* Springer, Berlin. 655-673
''Visual event perception.''

Kahn, K.M. (1976) *User-oriented design of interactive graphics systems (ACM/SIGGRAPH workshop).* 37-43
''An actor-based computer animation language.''

Kajiya, J.T. & von Herzen, B.P. (1984) *SIGGRAPH'84: Computer Graphics* **18**, 3 165-174
''Ray tracing volume densities.''

Kaufman, Y.J. (1981) *Applied Optics* **20**, 9 1525-1531
''Combined eye-atmosphere visibility model.''

Kaufman, Y.J. (1984) *Applied Optics* **23** 4164-4172
''Atmospheric effect on spatial resolution of surface imagery.''

Kay, A.C. (1977) *Scientific American* **237**, September 230-244
''Microelectronics and the personal computer.''

Kay, T.L. & Kajiya, J.T. (1986) *SIGGRAPH'86: Computer Graphics* **20**, 4 269-278
''Ray tracing complex scenes.''

Koenderink, J.J. (1986) *Vision Research* **26**, 1 161-180
''Optic flow.''

Koenderink, J.J. & van Doorn, A.J. (1975) *Optica Acta* **22**, 9 773-791
''Invariant properties of the motion parallax field due to the movement of rigid bodies relative to an observer.''

Koenderink, J.J. & van Doorn, A.J. (1986) *J.Opt.Soc.Am.A* **3**, 2 242-249
''Depth and shape from differential perspective in the presence of bending deformations.''

Kolers, P.A. (1984) *Computer Graphics* **18**, 1 12-16
''Motion from continuous or discontinuous arrangements.''

Korein, J.U. & Badler, N.I. (1983) *SIGGRAPH'83: Computer Graphics* **17**, 3 377-388
''Temporal anti-aliasing in computer generated animation.''

LaLonde, W.R., Thomas, D.A. & Pugh, J.R. (1986) *OOPSLA'86: SIGPLAN Notices* **21**, 11 322-330
''An exemplar based Smalltalk.''

Lasseter, J. (1987) *SIGGRAPH'87: Computer Graphics* **21**, 4 35-44
''Principles of traditional animation applied to 3D computer animation.''

Lee, D.N. (1980) *Phil.Trans.R.Soc.Lond.B* **290** 169-179
''The optic flow field: the foundation of vision.''

Lee, M.E., Redner, R.A. & Uselton, S.P. (1985) *SIGGRAPH'85: Computer Graphics* **19**, 3 61-67
''Statistically optimized sampling for distributed ray tracing.''

Li, Shu-Xiang & Loew, M.H. (1987a) *Comm.ACM* **30** 627-631
''Adjacency detection using quadcodes.''

Li, Shu-Xiang & Loew, M.H. (1987b) *Comm.ACM* **30** 621-626
''The quadcode and its arithmetic.''

Lieberman, H. (1986) *OOPSLA'86: SIGPLAN Notices* **21**, 11 214-223
''Using prototypical objects to implement shared behavior in object-oriented systems.''

MacCurdy, E. (1954) Vol.**II** *The Notebooks of Leonardo da Vinci.* The Reprint Society, London. (reprint of 1938 edition)

MacKay, D.M. (1981) *Nature* **289** 117-118
''Strife over visual cortical function.''

Magnenat-Thalmann, N. & Thalmann, D. (1985) *Computer Animation: Theory and Practice.* Springer-Verlag, Tokyo.

Magnenat-Thalmann, N. & Thalmann, D. (1987) *IEEE Computer Graphics & Applications* **7**, 8 27-44
''An indexed bibliography on image synthesis.''

Mandelbrot, B.B. (1982a) *Comm.ACM* **25**, 8 581-583
''Comment on computer rendering of of fractal stochastic models.''

Mandelbrot, B.B. (1982b) *The Fractal Geometry of Nature.* Freeman, New York.

Mandelbrot, B.B. (1983) *Handout distributed at SIGGRAPH'83, Detroit, MI* 5pp
''Remarks on computer rendering of fractal stochastic models.''

Marr, D. & Hildreth, E. (1980) *Proc.R.Soc.Lond.B* **207** 187-217
''Theory of edge detection.''

Marr, D. & Nishihara, H.K. (1978) *Proc.R.Soc.Lond.B* **200** 269-294
''Representation and recognition of the spatial organization of three-dimensional shapes.''

Marr, D. & Vaina, L. (1982) *Proc.R.Soc.Lond.B* **214** 501-524
''Representation and recognition of the movements of shapes.''

Mastin, G.A., Watterberg, P.A. & Mareda, J.F. (1987) *IEEE Computer Graphics & Applications* **7**, 3 16-23
''Fourier synthesis of ocean scenes.''

Maturana, H.R. & Varela, F.J. (1988) *The Tree of Knowledge: The Biological Roots of Human Understanding.* Shambhala, Boston. Original title: El árbol del conocimiento. Translated by R. Paolucci.

Meagher, D. (1982) *Computer Graphics & Im.Proc.* **19** 129-147
''Geometric modeling using octree encoding.''

Meagher, D.J. (1984) Vol.**2** *Proc.Computer Graphics '84.* 343-351
''A new mathematics for solids processing.''
5th Annual conf., nat.comput. graphics assoc, Anaheim, CA.

Miller, G.S.P. (1986) *SIGGRAPH'86: Computer Graphics* **20**, 4 39-48
''The definition and rendering of terrian maps.''

Miranda, E. (1987) *OOPSLA'87: SIGPLAN Notices* **22**, 12 354-365
''BrouHaHa — A portable Smalltalk Interpreter.''

Neumann, B. (1984) *Computer Graphics* **18**, 1 17-19
''Optical Flow.''

Newman, W.M. & Sproull, R.F. (1979) *Principles of interactive computer graphics.* McGraw-Hill. (2nd edition)

NeWS (1987) *Part No: 800-1498-05.* Sun Microsystems, Inc, Mountain View, CA. ‘‘NeWS Technical Overview.’’

Nicol, C.J. (1984) *Modelling solids in four dimensions.* PhD thesis, University of Glasgow. see also *Computer Graphics Forum* **4** 239-244 (1985)

Oliver, M.A. (1986) **in:** L.R.A. Kessener, Peter, F.J. & van Lierop, M.L.P. **(ed)** *Data Structures for Raster Graphics.* Springer-Verlag, Berlin. 9-37
‘‘Display algorithms for quadtrees and octtrees and their hardware realisation.’’

Oppenheimer, P.E. (1986) *SIGGRAPH’86: Computer Graphics* **20**, 4 55-64
‘‘Real-time design and animation of fractal plants and trees.’’

Pearson, D.E. (1975) *Transmission and Display of Pictorial Information.* Pentech Press, London.

Pentland, A.P. (1984) *IEEE Trans.Pat.Anal.Mach.Intel.* **6**, 6 661-674
‘‘Fractal-based description of natural scenes.’’

Pentland, A.P. (1985) *Image & Vision Computing* **3**, 4 153-161
‘‘On describing complex surface shapes.’’

Perlin, K. (1985) *SIGGRAPH’85: Computer Graphics* **19**, 3 287-296
‘‘An image synthesizer.’’

Pervin, E. & Webb, J.A. (1982) *Research Report CMU-CS-82-150* 15pp
Carnegie-Mellon Univ. .
‘‘Quaternions in computer vision and robotics.’’

PHIGS (1986) *ISO PHIGS revised working draft*
‘‘Programmer’s Hierarchical Interactive Graphics System.’’

Pirsig, R.M. (1974) *Zen and the art of motorcycle maintenance: An enquiry into values.* The Bodley Head.

Plato (1987) *The Republic.* Penguin Books, Hammondsworth, Middlesex, England. Second Edition (Revised). Translated by Desmond Lee.

Potmesil, M. & Chakravarty, I. (1983) *SIGGRAPH’83: Computer Graphics* **17**, 3 389-399
‘‘Modeling motion blur in computer generated images.’’

Preston-Dunlop, V. (1969) *Practical kinetography Laban.* Macdonald & Evans Ltd, London.

Reeves, W.T. (1983) *ACM Trans.Graphics* **2**, 2 91-108
‘‘Particle systems: A technique for modeling a class of fuzzy objects.’’
also in SIGGRAPH’83: Computer Graphics **17**

Reeves, W.T. (1987) *SIGGRAPH’87: Computer Graphics* **21**, 4 335-336
‘‘The physical simulation and visual representation of natural phenomena.’’
Statement for the panel.

Reeves, W.T. (1988) *Comm.ACM* **31**, 2 116-117
‘‘Physically based modeling vs ‘Faking it’.’’

Reeves, W.T. & Blau, R. (1985) *SIGGRAPH’85: Computer Graphics* **19**, 3 313-322
‘‘Approximate and probabilistic algorithms for shading and rendering structured particle systems.’’

Rensch, T. (1982) *SIGPLAN Notices* **17** 51-57
''Object oriented programming.''

Reynolds, C.W. (1982) *SIGGRAPH'82: Computer Graphics* **16**, 3 289-296
''Computer animation with scripts and actors.''

Reynolds, C.W. (1987) *SIGGRAPH'87: Computer Graphics* **21**, 4 25-34
''Flocks, herds and schools: A distributed behavioral model.''

Rogers, D.F. (1985) *Procedural Elements for Computer Graphics.* McGraw-Hill, New York.

Rubin, S.M. (1982) *Computer Graphics & Im.Proc.* **19** 291-298
''The representation and display of scenes with a wide range of detail.''

Rubin, S.M. & Whitted, T. (1980) *SIGGRAPH'80: Computer Graphics* **14**, 3 110-116
''A 3-dimensional representation for fast rendering of complex scenes.''

Salmon, R. & Slater, M. (1987) *Computer Graphics: Systems & Concepts.* Addison-Wesley, Wokingham, England.

Samet, H. (1984) *Computing Surveys* **16**, 2 187-260
''The quadtree and related hierarchical data structures.''

Samet, H. & Tamminen, M. (1985) *SIGGRAPH'85: Computer Graphics* **19**, 3 121-130
''Bintrees, CSG trees, and time.''

Sandor, J. (1985) *Computers & Graphics* **9**, 4 393-405
''Octree data structures and perspective imagery.''

Schachter, B.J. (1981) *IEEE Computer Graphics & Applications* **1**, 5 29-68
''Computer image generation for flight simulation.''

Schade, O.H. (1956) *J.Opt.Soc.Am.* **46** 721-739
''Optical and photoelectric analog of the eye.''

Shelley, K.L. & Greenberg, D.P. (1982) *SIGGRAPH'82: Computer Graphics* **16**, 3 157-166
''Path specification and path coherence.''

Shoemake, K. (1985) *SIGGRAPH'85: Computer Graphics* **19**, 3 245-254
''Animating rotation with quaternion curves.''

Smith, A.R. (1984) *SIGGRAPH'84: Computer Graphics* **18**, 3 1-10
''Plants, fractals, and formal languages.''

Smith, B. (1982) *Parts and Moments. Studies in Logic and Formal Ontology.* Philosophia Verlag, München.

Snyder, J.M. & Barr, A.H. (1987) *SIGGRAPH'87: Computer Graphics* **21**, 4 119-128
''Ray tracing complex models containing surface tessellations.''

Stefik, M. & Bobrow, D.G. (1985) *The AI Magazine* **VI**, 4 40-62
''Object-oriented programming: Themes and variations.''

Stroustrup, B. (1986) *SIGPLAN Notices* **21**, 10 7-18
''An Overview of C++.''

Sutherland, I.E., Sproull, R.F. & Schumacker, R.A. (1974) *Computing Surveys* **6**, 1 1-55
''A characterization of ten hidden-surface algorithms.''

van de Grind, W.A. (1987) **in:** Crow, F. & Pizer, S.M. **(ed)** *Proceedings of 1986 Workshop on Interactive 3D Graphics.* ACM, New York. 197-235
''Vision and the graphical simulation of spatial structure.''
(Chapel Hill, NC, Oct 1986)

Varela, F.J. (1984) **in:** Livingston, P., et al. **(ed)** *Disorder-Order: Proceedings of the Stanford International Symposium.* Anma Libri, Saratoga, Ca.
''Living ways of sense making: A middle way approach to neurosciences.''

Voss, R.F. (1985) **in:** Earnshaw, R.A. **(ed)** *NATO ASI Series. Fundamental Algorithms for Computer Graphics.* Springer, Berlin. 805-835 & plates C1-C9: pp 13-16
''Random fractal forgeries.''

Watson, A.B. & Ahumada, A.J. (jr) (1985) *J.Opt.Soc.Am.A* **2**, 2 322-341
''Model of human visual-motion sensing.''

Watson, A.B., Ahumada, A.J. (jr) & Farrell, J.E. (1986) *J.Opt.Soc.Am.A* **3**, 3 300-307
''Window of visibility: a psychopysical theory of fidelity in time-sampled visual motion displays.''

Waxman, A.M. & Ullman, S. (1985) *Int.J.Robotics Res.* **4**, 3 72-94
''Surface structure and three-dimensional motion from image flow kinematics.''

Waxman, A.M. & Wohn, K. (1985) *Int.J.Robotics Res.* **4**, 3 95-108
''Contour evolution, neighborhood deformation, and global image flow: Planar surfaces in motion.''

Weghorst, H., Hooper, G. & Greenberg, D.P. (1984) *ACM Trans.Graphics* **3**, 1 52-69
''Improved computational methods for ray tracing.''

Weiman, C.F.R. (1980) *SIGGRAPH'80: Computer Graphics* **14**, 3 286-293
''Continuous anti-aliased rotation and zoom of raster images.''

Weng, J. & Ahuja, N. (1987) *Computer Vision, Graphics & Im.Proc.* **39** 167-185
''Octrees of objects in arbitrary motion: Representation and efficiency.''

Westheimer, G. (1972) **in:** Jameson, D. & Hurvich L.M. **(ed)** *Handbook of Sensory Physiology.* Vol.**VII/4** *Visual Psychophysics.* Springer, Berlin. 170-187
''Visual acuity and spatial modulation thresholds.''

Whitted, T. (1980) *Comm.ACM* **23**, 6 343-349
''An improved illumination model for shaded display.''

Williams, L. (1983) *SIGGRAPH'83: Computer Graphics* **17**, 3 1-11
''Pyramidal parametrics.''

Winograd, T. & Flores, F. (1986) *Understanding Compters and Cognition: A New Foundation for Design.* Ablex Publishing Co, Norwood, New Jersey.

Witkin, A.P. (1983) *Int.Joint Conf.on Artificial Intelligence (Karlesruhe, Germany).* 1019-1022
''Scale-space filtering.''

Witkin, A.P. (1986) **in:** Pentland, A. **(ed)** *From Pixels to Predicates.* Ablex, Norwood, New Jersey. 5-19
''Scale space filtering.''

Wright, L. (1983) *Perspective in perspective.* Routledge & Kegan Paul, London.

Yu, F.T.S. (1976) *Optics and Information Theory.* Wiley, New York.

Zeltzer, D. (1985) *The Visual Computer* **1** 249-259
''Towards an integrated view of 3-D computer animation.''