A Programme for the Development of a Mathematical Theory of Computer Graphics

Eugene Fiume
Alain Fournier

Computer Systems Research Group
Department of Computer Science
University of Toronto
Toronto, Ontario, M5S 1A4

ABSTRACT

In response to the growing diversity of computer graphics, it is argued that a broad mathematical theory of computer graphics is required. Three aspects of such a theory are considered: the semantics of raster images, the semantics of interactive systems, and the theory of graphic input-output devices. Examples of our approach for each aspect are also presented.

RESUME

Nous essayons de montrer que pour répondre à la diversité croissante des problèmes et des méthodes de l'infographie, une théorie mathématique générale de l'infographie est nécessaire. Nous présentons trois aspects d'une telle théorie: la sémantique des images pour quadrillage, la sémantique des systèmes interactifs, et la théorie des appareils graphiques d'entrée et de sortie. Nous donnons également des examples de notre approche dans chaque aspect.

KEYWORDS: raster graphics, rendering, interactive systems, mathematical semantics, computational complexity, computational geometry, computability, measure theory, visible surface determination, anti-aliasing, temporal logic.

0. Introduction

Almost daily, we all hear of exciting announcements of new graphic input-output devices, display systems, and graphic processors. Computer graphics (CG) is a dynamic field which encourages innovation, new technology, and diversity. While this is one of the things that makes CG stimulating, the ever-increasing pervasiveness and diversity of CG is also worrisome. Our concern is that the meaning of fundamental graphic concepts and practices is being blurred by the overwhelming diversity of implementations of these concepts. One response to this problem is standardisation, and the past five years has seen several major efforts in this direction. However, even standardisation efforts are prone to undue influence from the prevailing technologies of the day. Is it really true, for example, that the segment is the best possible standard graphic data structure? More likely, the choice of the segment coincided with the once-widespread proliferation of refreshed display-file systems--a choice we still have to live with today, despite the current popularity of alternative display technologies.

Our response to the growing diversity and applicability of CG is to study its mathematical foundations. We hope to accrue two major, long-term benefits of a theoretical approach: to give the mathematical semantics of important graphic concepts, and to understand the essential properties of specific classes of input-output devices. This would provide us with a basis for reasoning about graphic algorithms and concepts independently of their implementations. In an age of changing models of computation, this has important practical benefits.

Many aspects of CG are not without a mathematical basis. Unfortunately, its pragmatic nature has resulted in isolated pockets of mathematical formulation rather than a unified whole. Our programme for a theory of CG consists of three basic components:

(1) A semantics of raster images.

(2) A semantics of interactive systems.

(3) A theory of graphic input-output devices.

This brief paper presents an overview of these components. The appendix gives a concrete example for each component. Our approach is in some sense "meta-semantic", in that our formulations generalise beyond the task of providing the semantics of a specific system. We stress the fact that our approach is mathematical. This does not mean that it is inaccessible. The meaning of a graphic concept is given by presenting a formal model for it, which is specified using straightforward mathematical notation.

1. The Semantics of Raster Images

Our image semantics is based on raster or pixel representations. Given a scene composed of a set of objects, the *semantics* of the rendered image is a prescription for the scene's expected appearance. Formally, to any scene S represented by a list L of objects, we associate a "meaning" function $\mathbf{Ras}_L(i,j)$, which uniquely prescribes the intensity value for each (i,j) in the raster display. This process requires the development of:

(a) an object representation schema.

(b) a pixel model and the induced topology of scenes defined using (a).

(c) the rendering semantics for each pixel.

An object O prescribes a triple (S,Z,I), where S denotes its coverage in the xy-plane, and Z and I are its depth and intensity functions, respectively. *Any* object specification technique can be used to define objects, as long as its semantics prescribes such triples (e.g. [Mall82; Cars83]). A pixel model is a tessellation of the Cartesian plane based on one prototile. Using this model, together with topological notions such as connectedness, a theory of raster operations can be developed. This is particularly important, as it readily provides a basis for combining conventional graphic primitives and transformations with raster operations. Combining these concepts has been an elusive goal which others have attempted (e.g. [GuSt82; AFSW82]).

The intensity of each pixel is defined as a function of the coverage of each object within a pixel. This is easily generalised to multiple-pixel schemes. The concept of *measure* from measure theory is used to unify the many rendering possibilities. Measures are developed to give point-sampling, super-sampling, and area-sampling semantics. All of the major anti-aliasing techniques can be given a concise characterisation. This includes techniques such as exact and weighted (i.e. convolution-based) area-sampling. Moreover, since measures extend to spaces of higher dimensions, temporal anti-aliasing schemes are simply accommodated.

2. The Semantics of Interactive Systems

The semantics of most images is best viewed as a mathematical function instantaneously mapping a set of input values to output values. However, when applied to interactive systems, this view can be unnatural, for the crucial temporal component is abstracted away. We instead model interactive systems in terms of communication channels along which information flows. Information is characterised by messages. A message appearing on one channel may trigger the appearance of others elsewhere. These relationships are specified by means of an iterative grammar. Formally this grammar is a generalised morphism (a finite substitution) mapping messages to messages. The content of all communication channels in a system is its current state. A single iteration of the morphism gives rise to the current state's possible transitions. The prescription of a particular system is given by specifying constraints among states, thus restricting the set of admissible executions. The *semantics* of an interactive system is given by this set of admissible execution sequences. The iterative nature of the grammar very naturally leads to a temporal interpretation. Temporal system execution constraints are thus conveniently constructed. Moreover, the implicit execution characteristics of a specification, such as deadlock, real-time performance, channel bandwidth, etc., can be formally studied. Perhaps the most important practical benefit of this approach is that a simulation mechanism is implicit in the specification technique.

3. The Theory of Graphic Input-Output Devices

As was indicated in the introduction, implementers of interactive graphic systems have an increasingly rich variety of devices from which to choose. As yet, however, there has been no formal study of interesting device classes. This is of more than just mathematical interest. Such a study would have important practical applications since a formal model of a device clearly indicates its capabilities and the constraints placed upon an implementation of a graphic system utilising the

device. The algorithms applicable to the device and their expected performance could be established prior to implementation. A model is particularly important for a family of devices whose capabilities differ widely. The frame buffer is a good example. Various frame buffers have highly disparate sets of capabilities. A frame buffer model that indicates how much is gained, both theoretically and practically, from the addition of a new memory capability, for example, would be of great value.

Theorists have long built various models of computation. For example, the Turing machine is one possible abstraction of computing mechanisms. Our device models are also at this level of abstraction. A model for a class of devices is an automata-theoretic construction specifying only its essential properties, the implications of which one wishes to study. For example, a frame buffer model might specify how the frame store is accessed, the format of the store, the available memory per access, and general timing characteristics. A strategy can then be developed to optimise the use of devices having these properties. This approach is particularly appealing since it allows one to specify algorithms that are both general and optimal for the desired set of implementations. Moreover, it permits infeasible algorithms to be identified and ruled out as candidates for implementation.

4. Conclusion

This summary has presented an overview of the three components we believe are essential to a practical theory of Computer Graphics. Research into these areas is well underway, and we believe substantial progress has been realised in each area [FoFu83; FiFo84]. More concrete examples of our approach are found in the appendix to follow.

5. Acknowledgements

The act of writing down or presenting a few predicates and functions is unlikely to ever qualify us for entry into the gigaflop club. We nevertheless gratefully acknowledge the financial assistance of the Natural Sciences and Engineering Research Council of Canada. Figure 1 was generated using software written by David Grindal.

Appendix

In this appendix, we shall consider some basic examples of our approach. The semantics of raster images will be illustrated by defining the semantics of *exact area sampling* [FiFR83], a common rendering technique which incorporates both visible surface determination and anti-aliasing. The semantics of a very simple interactive system will then be given. Temporal logic will be used to express powerful assertions in a concise, natural fashion. Lastly, we will illustrate our approach to the theory of graphic input-output devices by briefly considering a theoretical model of the frame buffer.

A. The Semantics of a Rendering Technique

Suppose we have a scene which is specified by a list L of graphic objects. Recall that our aim is to produce a function $\mathbf{Ras}_L$ which prescribes the intensity of each pixel in the display. Clearly the semantics of L depends on the rendering technique and the model of pixels we choose to employ. For simplicity, we shall assume that the pixel model is a tessellation of the Cartesian plane into unit squares. A given pixel area in the pixel model is denoted by P_{ij}. Recall from above that each object O in L is a triple (S,Z,I), where S is its coverage, Z is the depth function and I is the intensity function over S.

We typically represent S implicitly by its characteristic function, χ_S. The *characteristic function* χ_S of a set S takes on the value 1 for all points in S and 0 otherwise. Such functions are often more useful than the sets they define by extension. For example, clipping an object to a region is expressed by multiplying their characteristic functions.

Since L specifies a scene embedded in a continuous space, its rendering requires discretisation and therefore some form of sampling. The nature of sampling performed determines a particular rendering semantics. The essence of all sampling processes is that the contribution of an object to a pixel's ultimate intensity is related to the visible coverage of that object within the pixel (and perhaps within neighbouring pixels as well). From measure theory [Halm50], a *measure* μ defined over a class of sets C is a non-negative set function $\mu: C \rightarrow \mathbf{R}$ having the following properties:

(a) $\mu(\phi)=0$.

(b) Countable additivity: if the sets $S_1, S_2, \cdots$ are pairwise disjoint, then $\mu(\bigcup_i S_i)=\sum_i \mu(S_i)$.

When the range of μ is [0,1] rather than **R**, it is called a *probability measure*. These measures nicely capture the intuitive notions of "weight" or "fractional coverage" often found in anti-aliasing literature.

We are now in a position to formulate a pixel-rendering semantics. Suppose $L = \{O_1, \cdots, O_N\}$, where each object $O_i = (S_i, Z_i, I_i)$. Define $Vis(i,j) = \{V_1, \cdots, V_N\}$, where object V_k is the visible portion of object O_k within pixel P_{ij}. Formally,

$$V_i = (S'_i, Z_i, I_i), \text{ where}$$

$$S'_i = \{p \in S_i \cap P_{ij} \mid Z_i(p) = \min_{k=1}^{N} Z_k(p)\}.$$

The *visible surface problem* (VSP) is that of determining $Vis(i,j)$ for a given L for all (i,j).

To simplify the mathematics, assume that the intensity of each visible surface within a pixel is constant. Formally, we associate a constant Int_k to each visible object V_k in $Vis(i,j)$.

Let us assume the background is defined as some object O_k in L. Then observe firstly that all visible surfaces in $Vis(i,j)$ are disjoint, and secondly that their union is all of P_{ij}. Thus we can exploit the countable additivity of measures to define the rendering semantics for a given pixel:

$$\mathbf{Ras}_L(i,j) = \sum_{k=1}^{N} Int_k \times \mu^{ij}(V_k),$$

where μ^{ij} is a (probability) measure expressing the desired sampling process about P_{ij}. This simply says that the ultimate intensity of P_{ij} is the weighted sum of the intensity approximations for each visible surface within the pixel. This is an extremely flexible arrangement: a different rendering semantics is obtained simply by "snapping in" a different measure. Indeed, various measures can be used on the same scene, if desired. It is nearly trivial to extend the definition of **Ras** make the ultimate intensity of a single pixel depend on sampling processes over many others [Crow77].

We now consider the measure-theoretic characterisation of a common rendering technique: exact area sampling. This technique states that the contribution of an object to the intensity of a given pixel (i,j) is proportional to the object's visible coverage over P_{ij} [FiFR83; FoFu83]. Its formal definition is straightforward:

$$\mu^{ij}(S) = \frac{\int\int_{P_{ij}} \chi_S(x,y)\, dx\, dy}{A_{P_{ij}}}$$

where $A_{P_{ij}}$ is the area of pixel P_{ij} (which is 1 in our case). Given a visible surface V over pixel P, $\mu^{ij}(V)$ is precisely the proportion of P covered by V. All known area- or point-based rendering techniques can be defined just as easily [FiFo84].

B. The Semantics of an Interactive System

Recall that an interactive system is modelled as a set of communication channels and information flowing through them. The basic atom of information flow is the *message*. An interactive system is specified by a grammar of message production and consumption rules, as well as a set of temporal assertions that guide the application of these rules.

Production and Consumption Rules

The grammar for specifying productions and consumptions is based on so-called 0L-systems [Salo81]. Suppose the system we wish to specify has n channels. At any instant of time, the contents of all channels is reflected by a single string (although many such strings may be possible at a given instant). The "end" of each channel k is indicated by $\#_k$. Let Σ denote the set of all possible messages (or alternatively message tokens).

Two types of message production rules are permitted. An *event* is of the form $\#_k \rightarrow a\#_k$, where $a \in \Sigma$. This causes message "a" to appear as the last message in channel k. An *idle* is in effect a non-operation: $a \rightarrow a$, where a is any character. The grammar is usually nondeterministic. For example, consider the following production rules.

P_1: $\#_1 \rightarrow a\#_1$

P_2: $\sigma \rightarrow \sigma \qquad \sigma \in \{a, \#_1\}$

In this example, channel 1 can either grow (P_1) or idle (P_2) at any iteration of the grammar. A message consumption is of the form $a \rightarrow \varepsilon$, where $a \in \Sigma$, and ε is the empty string.

A grammar, which operates independently on characters, can be easily extended to a string-mapping relation f. Informally f maps a string to one or more strings by performing a character-wise translation as defined by the grammar. An *execution sequence* of an interactive system is defined as a sequence $w = <w_0, w_1, w_2, \cdots>$, where w_0 denotes the initial contents of all channels, namely the "empty" string $\#_1 \cdots \#_n$, and $w_{i+1} \in f(w_i)$, $i \geq 0$. An execution sequence of the above example is $<\#_1, a\#_1, aa\#_1, \cdots>$, as is

$<\#_1, \#_1, \#_1, \cdots>$. We shall assume that messages are produced and consumed in a FIFO manner. This could be easily specified using temporal assertions.

Temporal Assertions

Our assertion language, which is based on temporal logic [ReUr71; Pneu79; OwLa79], has been developed to express the truth of assertions over the passage of real time. The model of time we exploit is the inherent "ticking" of the iterative grammar. Let P be a predicate. We introduce the following temporal qualifiers of P:

P means "P is true now"
∇P means "P is true now or will be"
∇_i P means "P will be true at time i"
$\Box$P means "P is true now and forever will be"

Composite temporal qualifiers can be also be formed. For example,

$\Box\nabla$P means "P will be true infinitely often"
$\nabla_i \Box$P means "P will be forever true after time i".

We now consider a simple-minded example, a system composed of two channels. On one channel, a graphics tablet produces a steady stream of spatial co-ordinates. On another channel, a screen handler consumes messages sent to it, and displays a tracking symbol at the position specified by the message. To maintain good feedback, clearly the production of a tablet message should result in the production of a message to the screen handler within a reasonable time. Our specifications will state that this must be the case, but it will not say "who" accomplishes this, nor how it is to be done. These tasks are duties of implementations. Let $\Sigma = \{a, b\}$, where a is a token for tablet messages and b is a token for screen messages. We shall omit the functional specifications of the processes and concentrate instead on the information flow in this system, which the following grammar prescribes.

TabIdle:	$a \rightarrow a, \#_1 \rightarrow \#_1$
ScreenIdle:	$b \rightarrow b, \#_2 \rightarrow \#_2$
TabProd:	$\#_1 \rightarrow a\#_1$
ScreenProd:	$\#_2 \rightarrow b\#_2$
TabCons:	$a \rightarrow \varepsilon$
ScreenCons:	$b \rightarrow \varepsilon$

This grammar defines an infinite set of execution sequences. However, not all of these executions are desirable. For example, sequences in which screen productions occur before any tablet production are in the set, as are sequences in which tablet productions are arbitrarily "far ahead" of screen consumptions. Let us therefore apply some constraints to this set.

First, suppose the tablet produces co-ordinates every 20 ticks:

$\forall i \geq 0 \; . \; \nabla_{20i}$ TabProd.

A tablet message must be consumed, and a message must be sent to the screen within 10 ticks of the tablet's message being issued:

$\forall i \geq 0 \; . \; (\nabla_i \text{ TabProd} \Rightarrow \nabla_j \text{ TabCons} \wedge \nabla_k \text{ ScreenProd})$,

where $i < j < k < i + 10$.

To ensure immediate feedback, one could specify that a screen message must be immediately consumed:

$\forall i \geq 0 \; . \; (\nabla_i \text{ ScreenProd} \Rightarrow \nabla_{i+1} \text{ScreenCons})$.

Lastly, a screen production cannot occur before its corresponding tablet production. That is, there is no 20-tick time window in which more than one screen production occurs:

$\forall i \geq 0 \; \exists \, !j < 20 \; . \; \nabla_{20i+j}$ ScreenProd.

The notation $\exists \, !x$ means "there exists a unique x".

C. A Formal Model of a Frame Buffer

A graphics device which is particularly valuable to model is the frame buffer. We shall view a frame buffer as an $m \times n$ array of pixel automata. Each automaton models the capabilities of a distinct pixel within the frame buffer, and contains the following components:

- a read-only input tape
- a local memory
- a finite state control

The input tape contains the information necessary to perform a certain graphic operation at the pixel level. The memory is composed of a small set of bounded registers; these registers could store depth and intensity values, for example. The finite state control defines a small set of functions and/or predicates defined over each pixel's memory registers and input tape. By varying the number of registers per pixel, their use, and the power of finite state control (i.e. the number of functions and predicates), many theoretically and practically interesting results can be proven regarding the overall power of the frame buffer.

Many such results are summarised in [FiFo84]. [FoFu83] is a more rigorous presentation of the model and these results.

One nice application of this model is to determine the minimal properties of a frame buffer which are necessary and sufficient to implement nontrivial graphics algorithms. In fact, it is surprising how few registers and functions are actually required. For example, [FoFu83] proves that with tnree registers per pixel, convex polyhedra can be rendered by the pixel automata. This was demonstrated by presenting a two-pass algorithm which was based on the convex intersection of half-spaces. Moreover, it is also shown that a two-pass algorithm *and* at least three registers are necessary to do the job. Figure 1 illustrates a well-known convex polyhedron generated in this manner. The two-pass algorithm for the pixel automata was directly transferred to a microcode implementation on a bit-slice microprocessor.

References

AFSW82 Acquah, J., J. Foley, J. Sibert, and P. Wenner, "A Conceptual Model of Raster Graphics Systems", *Proceedings of SIGGRAPH '82*, also published as *Computer Graphics 16*, 3, (July 1982), 321-328.

Cars83 Carson, G.S., "The Specification of Computer Graphics Systems", *IEEE Computer Graphics and Applications 3*, 6 (Sept 1983), 27-41.

Crow77 Crow, F.C., "The Aliasing Problem in Computer-Generated Shaded Images", *Commun. ACM 20*, 11 (Nov. 1977), 799-805.

FiFo84 Fiume, E., and A. Fournier, "Elements of a Theory of Computer Graphics", Dynamic Graphics Project Technical Memo., Department of Computer Science, University of Toronto, 1984.

FoFu83 Fournier, A., and D. Fussell "On the Power of the Frame Buffer", Dynamic Graphics Project Technical Memo., Department of Computer Science, University of Toronto, 1983.

GuSt82 Guibas, L.J., and J. Stolfi, "A Language for Bitmap Manipulation", *ACM Transactions on Graphics 1*, 3 (July 1982), 191-214.

Halm50 Halmos, P.R., *Measure Theory*, D. Van Nostrand Company, New York, 1950 (also Springer-Verlag, New York, 1970).

Mall82 Mallgren, W.R., "Formal Specification of Graphic Data Types", *ACM Transactions on Programming Languages and Systems 4*, 4 (Oct. 1982), 687-710.

OwLa82 Owicki, S., Lamport, L., "Proving Liveness Properties of Concurrent Programs", *ACM Transactions on Programming Languages and Systems 4*, 3 (July 1982), 455-495.

Pnue79 Pnueli, A., "The Temporal Semantics of Concurrent Programs", *Semantics of Concurrent Computation*, Proceedings of the International Symposium, Evian, France, Springer-Verlag, 1979.

ReUr71 Rescher, N., and A. Urquhart, *Temporal Logic*, Springer-Verlag, New York, 1971.

Salo81 Salomaa, A., *Jewels of Formal Language Theory*, Computer Science Press, Rockville, Maryland, 1981.

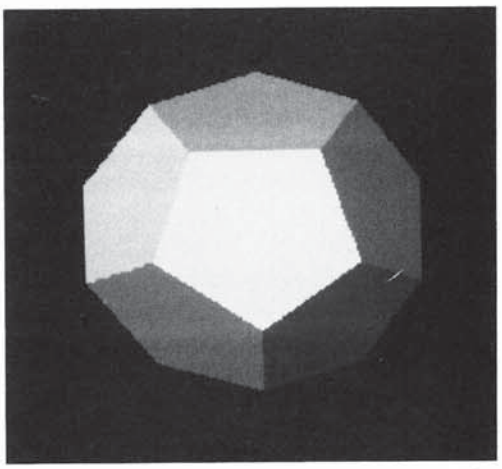

Figure 1. Dodecahedron. Any convex polyhedron can be generated by computing the convex intersection of the half-planes defining its faces.