A PROPOSAL FOR A FOUR-DIMENSIONAL GRAPHICS SYSTEM

A. Fournier

Department of Computer Science
University of Toronto

ABSTRACT

A computer graphics system models a world (either the physical world or an abstract one) and displays it for us on a device according to some mapping rules. Heavily influenced by the nature and limitations of display devices, the world modeled so far has been described in two or three dimension spatial coordinate systems. As a result time and motion has been neglected, except in specialized systems as in animation and simulation.

We propose here a graphics system where world objects are described in four dimensions, normally interpreted as being time plus three spatial dimensions, and discuss the advantages, drawbacks and prospective of such a system. They are examined in relation to the classic modeling primitives, the usual transformations and the current display devices and techniques. The advantages are seen to be improved modeling of complex motion, better representation of mechanical systems, built-in key frame interpolation, good potential interface with motion analysis systems. The main drawback is that it exacerbates the limitations of current display devices and interactive object definition systems. We sketch out a prototype system implementing these concepts.

RÉSUMÉ

Un système en infographie modèle un univers (soit une partie de l'univers physique, soit un univers abstrait) et le représente pour nous sur un système d'affichage après un ensemble de transformations. Très influencé par la nature et les limitations des systèmes d'affichages, l'univers modelé jusqu'à présent a été décrit dans un système de coordonnées à deux ou trois dimensions, exclusivement spatiales. En conséquence, le temps et le mouvement ont été négligés, sauf bien sûr dans les systèmes spécialisés, comme en animation ou en simulation.

Nous proposons ici un système graphique où à tous les niveaux les objects de l'univers représenté sont modelés en quatre dimensions, normalement interprétées comme le temps et les trois coordonnées spatiales, et nous discutons les avantages, les inconvénients et l'avenir pratique d'un tel système. Ces points sont examinés relativement aux primitives des modèles classiques, aux transformations courantes et aux systèmes d'affichages présentement utilisés. Les avantages semblent être une meilleure représentation des mouvements complexes et des systèmes mécaniques, l'interpolation des images-clés de façon automatique, et une meilleure interface possible avec les systèmes d'analyse des mouvements en intelligence artificielle. L'inconvénient principal semble être d'exacerber les limitations actuelles des modes de définition interactive des objets et de leur affichage. Nous esquissons un prototype d'un système appliquant ces concepts.

A proposal for a four-dimensional graphics system

Alain Fournier

Department of Computer Science
University of Toronto

1. Introduction

Time and motion are two concepts of prime importance in computer graphics, especially in applications such as animation and simulation, but less obviously in design and even in plotting. In spite of this, basic graphics packages do not explicitly include time as a variable, and classic modeling methods do not allow defining objects with a time dimension. (Indeed, "time" and "motion" do not even appear in the index of the standard interactive computer graphics textbook [Newm79]). The usual way to model motion in a graphic system is to apply a spatial transformation matrix to the objects defined at a finite set of values of real or simulated time. The elements of the transformation matrix can themselves be functions of time. There are many drawbacks to this approach. One is that motion is then totally outside of the graphics package, and requires an elaborate and cumbersome specification for complex motions. Another is that only the entities to which a transformation matrix can be applied in the system can move independently. Thus the type of motion allowed depends upon the modeling primitives used. For example, if a cube is modeled by polygonal faces, it is impossible to represent a transformation in which only one corner of the cube moves. To convince oneself of this, consider than the four vertices defining a face might not be coplanar after such a transformation, precluding the use of a linear transformation. The related motion of several primitives is also at best difficult. Complex motions, such as irregular waves at the surface of water, or of the skin of the human body, are almost completely out of reach with a standard system.

From research in areas where motion synthesis, analysis and/or understanding is essential, several important concepts and approaches emerged.

The first one, from traditional and computer assisted animation is *key-frame interpolation* [Burt76]. Here we usually have entire frames (mostly of 2-D primitives), with each one having a time associated with it. The in-between frames are computed by interpolation, usually linear. The concept of in-betweening has been refined and the techniques extended, notably in [Reev80].

Another relevant concept, from Baecker's pioneering work on interactive computer mediated animation, is the *p-curve* [Baec69]. A p-curve is a parametric curve (time is the parameter) sketched by the user in real time. In the normal use of the p-curve, real time (input time) is mapped directly to movie time (or play-back time in general). The interesting point in the context of this discussion is the fact that an "object", here a moving point, is represented as a two or three dimensional phenomenon.

A third important concept is from Futrelle's work as embodied in GALATEA [Futr74]. It is the notion of *time dependent objects*, which in GALATEA are independently scheduled moving objects (here again "object" is taken rather generally), defined by the user to "track" real motion on the film under study.

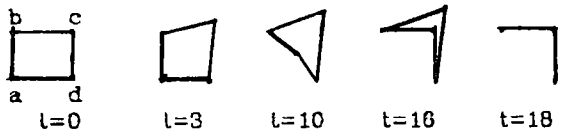
All these examples and other advances in the field lead to the conclusion that there should be significant advantages in including time explicitly as a fourth (or third in 2-D graphics) dimension at *all* the levels of the general graphics system, from modeling primitives to graphic primitives to output primitives, and not forcing the user or the application programmer to introduce it as an outside parameter. Beyond the practical advantages that we will try to put forth, there are conceptual reasons. This way of modeling time and motion should liberate the user from the frame by frame tyranny and allow hem to think in term of concurrently scheduled motions, which is the more natural way. We will see that we will not have to pay the price of a concurrent programming language and/or concurrent graphics package for this. In effect, it is like going from time-driven to event-driven simulation, or to use a more graphic analogy, like defining a line segment by its two end points, instead of having to give all its intersections with the scan-lines. An additional motivation is the fact that display devices are becoming more sophisticated and powerful, and can and should assume more of the computational burden, especially if it reduces the bandwith between the device and the main processor(s).

2. Basic techniques

The detailed way to introduce time as a parameter in the graphics system is as varied as the modeling techniques, the transformations and the display file structures in computer graphics.

To give an example at the display file level, we will assume a transformed segmented display file.

The sequence to be represented can be described using key frames by:



Note that the point c is in motion from t=0 to t=10, and then from t=16 to t=18, while a is in motion from t=3 to t=16. Points b and d do not move. This can be represented in the display file as:

Point	X	Y	t_{in}	t_{fin}
a	2	2	0	3
a	5	5	16	∞
b	2	5	0	∞
c	5	5	0	0
c	6	7	10	16
c	5	5	18	∞
d	5	2	0	∞

The display processor, while going through the file, will execute the following algorithm (it is assumed that it will do the right thing for the first point and the last point of a polygon; t_{in} and X' refers to the time-point following the current one):

for each point sequentially
if $t<t_{in}$ *then skip the point*
if $t_{in}\leq t\leq t_{fin}$ *then drawto(X,Y)*
if $t>t_{fin}$ *then interpolate*

$$X_i=X+\frac{(X'-X)(t-t_{fin})}{t_{in}'-t_{fin}}$$

and Y_i *the same way,*
drawto(X_i,Y_i)

The interpolation could be made easier on the processor by having the display file compiler put in the file the incremental value for X and Y. Also note that a point can be made to disappear by adding the condition that, if $t_{in}\leq t_{fin}$, then the point is skipped, or by using an extra bit to indicate the first occurence of a particular point. A point can be made to appear in the same fashion.

This represents one of the simplest schemes. Other variations include having the intensity or colour of the line be interpolated in the same way; the motion can be made cyclic, by an different opcode which will indicate that clock-time is to be taken modulo t_{cycle} for this point or group of points. As a smart display processor can draw circles or parametric curves, the processor can compute non-linear time interpolation. This would be useful, since the simple scheme illustrated here represents motion by "straight line segment approximation", and therefore suffer from defects similar to polygonal approximations of curves. If the transformations are done at the level of the display processor, then a similar interpolation can be effected on the transformation matrices.

Note that time slippage is not a major problem,if t is the real clock time. The motion will be less smooth, but the picture will "catch up". In refresh displays the time interpolation will not represent much additional work for the processor, and if several processors are used, they can of course do this in parallel.

To see the main advantage of this approach, we note, assuming that one word is required to store each spatial or temporal coordinate, and assuming that k frames are needed in 18 time units, that only 28 words rather than k×8 words are required. Even if each motion spans only a few frames, the required data transfer rate is much lower.

At the modeling level, which is more application dependent, the possibilities are also numerous, but fortunately more work has been done along these lines. The most easily adapted models are procedural models. It is a straightforward extension to have the procedures output X, Y, Z and T instead of X, Y, Z. Points, lines and polygons are similarly easily extended. In fact, a possible structure for polygon+time files is one similar to the structure of the display file described above. If adjacent polygons are supposed to move together, a variant of the points-polygons file structure would be adequate. Parametric curves and surfaces offer interesting possibilities, and the work of Reeves [Reev80] already shows techniques for parametric patch computations where one of the two parameters is time. A generalization to parametric "volumes" should be of interest. Moving volumes, in particular "soft" shapes like beating hearts and growing slimes, should also become easier to handle. Stochastic models [Four80] are also natural for this extension, since the stochastic processes which generate them can be (and generally are) function of time. Actually, the entire stochastic process computation can be pushed down to the display processor (with or without specialized hardware), and we can have "amber waves of grain" by sending the "field" description only once to the display file.

3. Operations and transformations

Of the usual spatial transformations, translation and scaling have easily extrapolated meanings. We can slow down and speed up the motion (we are referring here to the global motion of a graphic primitive *before* it is sent to the transformed display file), and move it in time , as well as in space. We can then instantiate a moving object. From the same master object, for instance a car, we can, using only transformation matrices, create multiple instances that move at different speeds, and start and stop at different times. More intriguing is the effect of rotation. Orthogonal rotations simply map

a coordinate to another, so in effect time is mapped to a spatial coordinate, which can be useful to analyze motion. Other rotations, however, have results harder to conceive, and their use is not obvious. Just recall that this is done in Minkowski's diagrams in Relativity Theory.

The perspective transformation is not affected, and can be done by matrix multiplication if convenient, but the effect on T must be undone for proper behaviour:

$$[XYZT1]\begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&\frac{1}{d}\\0&0&0&1&0\\0&0&0&0&1\end{bmatrix}=[XYZT\frac{Z+d}{d}]$$

$$=[\frac{Xd}{Z+d}\frac{Yd}{Z+d}\frac{Zd}{Z+d}\frac{Td}{Z+d}1]$$

if we use homogeneous coordinates. So T has to be multiplied by $\frac{Z+d}{d}$.

Clipping remains the same for the spatial coordinates. In addition, however, clipping in time can be very useful. It corresponds to a freeze-frame at time t (the clipping time, we clip against a plane of constant time). It should be done for the initial time, and in most cases some final time (the hither time and the yon time). A clipping method like reentrant polygon clipping [Suth74] will work here if polygons are used as graphic primitives. Clipping is also useful for splitting the display file into time slices, to limit its size. The transformation system will clip from t_0 to t_1, send this to the display file, then from t_1 to t_2, and this can be sent any time to the display file as long as it is before t_1 (by the display processor clock). If double buffering is used, it is a simple matter for the processor to switch to the second buffer at t_1. Alternatively, each segment can be processed individually in this manner.

Another operation that deserves investigation is the hidden lines/surfaces/volumes removal in four (formerly three) dimensions. If a depth-buffer approach is used, there is no special problem, if it can be done fast enough. Schemes that depend on a static view (like priority ordering) will not work here, unless we constrain the motion, for instance by having a background, plus linearly separable clusters, and unchanged priority within each cluster. In general, any decision will have to be made after the display processor updates the coordinates. It is interesting to note that we can use the display system for collision detection. For instance, if we use polyhedra, there is a collision if part of one polygon of one object is between a front facing and back facing polygon of another object.

And finally, properties such as speed, acceleration can be computed easily in the graphics system itself if need be.

4. Implementation and Applications

We have seen that the original time dimension in the model can be mapped to another dimension at the level of the output primitive. After that, we have assumed so far that the output device maps time in the display file to real time. But it does not have to be so. (We note in passing that most of the display devices we have are 3-D devices, two spatial coordinates and real time.) We can also make use of the fact that some have another dimension, usually seen as color or intensity, but which can be interpreted in a different way. In particular, we have an easy way to implement colour table animation [Shou79], by mapping the display processor time to colour. The ways in which this mapping is done will depend on the number of frames and colours used. A straightforward ways is to make the display processor event-driven, instead of time-driven, and write each new frame in a different bit-plane of the frame-buffer.

The implementation we are currently considering for the graphics system will begin at the display file and display processor level. We could chose to use a refreshed line drawing device, since they are the ones usually associated with display files, but we find it more interesting to use a frame-buffer/raster-scan device. The reasons are mainly that they allow colour, and are not generally considered as real time display devices, and we want to clearly distinguish between the refresh function and the "scheduling" function. Besides, we have two excellent candidates, SPIWRIT [Baec81], and PERQ, both microprogrammable frame buffers, one with low resolution (256*256*4bits) but some colour, the other with high resolution (1024*768), but only one bit deep.

The next step will be the implementation of the graphics package, which should be fairly similar to a traditional one, but for the added dimension. And last we will attach some modeling systems, probably stochastic modeling at first, and some applications programs. To keep the prototype system simple, we will start with a 2-D (spatial) implementation. But this should be rich enough to provide a basis for evaluation and a guide to future development. Most of the application domain has been mentioned, but an additional area should be discussed here: motion understanding in Artificial Intelligence [Tsot80]. The output of a system analyzing and understanding motion (in medical applications, often 2-D motion) is a set of canonical motions, with starting time (frame) and stopping time, or a period for cyclic motions. These could be directly accepted by an application program using the same motions as modeling primitives, and displayed. This would constitute an analysis/synthesis cycle à la GALATEA, and synergistically increase the usefulness of each part.

In general, though, this approach makes the

object definition problem worse, particularly if the objects are user-defined. A considerable amount of work on input devices and users' interface has been done to facilitate the description and design of 2-D and 3-D objects, and they are barely beginning to result in "friendly" and practical systems, so the addition of another dimension will not make the matter easier. Works in animation [Baec69] and languages [Berg77], [Kahn78] point in the right directions.

5. Conclusion

We have shown that time, instead of being a parameter confined to the application program, and left to be dealt with by the user or the application programmer, should be treated along and equally with the other coordinates. We showed that there is not much added onus placed on the display file processor by turning it into a scheduler, and that the appearance of time in the graphics package gives added power to the standard transformations. We also expressed our belief that the new generation of fast, smart raster scan devices are good candidates for display devices in the systems considered.

Acknowledgements

I would like to thank Ron Baecker for the environment he created, and for the helpful advice. This research was facilitated by the use of the Theory Net, NSF Grant MCS-78-01689.

References

[Baec69]Baecker, R. M. *Interactive computer-mediated animation.* MIT Project MAC, TR-61, (1969).

[Baec81]Baecker, R. M., Miller, D. and Reeves, W. A prototype laboratory instrument for video motion analysis. *This Conference.*

[Berg77]Bergman, S. and Kaufman, A. Association of graphic images and dynamic attributes. Computer Graphics, *11*, 2, pp. 18-23 (SIGGRAPH'77).

[Burt76]Burtnick, N. and Wein, M. Interactive skeleton techniques for enhancing motion dynamics in key frame animation . CACM, *19*, 10 , pp. 564-569 (Oct 1976).

[Four80]Fournier, A. and Fussell, D. Stochastic modeling in computer graphics. SIGGRAPH'80, to appear in CACM.

[Futr74]Futrelle, R. P. GALATEA: Interactive graphics for the analysis of moving images. Information Processing '74, *4*, pp. 712-716 (1974).

[Kahn78]Kahn, K. M. and Hewitt, C. Dynamic graphics using quasi-parallelism. Computer Graphics, *12*, 3, pp. 357-362 (SIGGRAPH'78).

[Newm79]Newman, W. M. and Sproull, R. F. *Principles of interactive computer graphics.* (Second Edition), Mc Graw Hill, (1979).

[Reev80]Reeves, W. T. *Quantitative representation of complex dynamic shapes.* PhD Thesis, Department of Computer Science, University of Toronto, (1980).

[Shou79]Shoup, R. G. Color table animation. Computer Graphics, *13*, 2, pp. 8-13 (SIGGRAPH'79).

[Suth74]Sutherland, I. E. and Hodgman, G. W. Reentrant polygon clipping. CACM, *17*, 1, pp.32-42 (Jan 1974).

[Tsot80]Tsotsos, J. K. *A framework for visual motion understanding.* PhD THesis, Department of Computer Science, University of Toronto, (1980).