Primitives in Computer Graphics (Summary of Presentation)

Alain Fournier

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

ABSTRACT

Before they are embodied in a graphics system as datatypes, objects or functions, the primitives we want or can manipulate have to be clearly defined. The primitives used so far (modelling, geometric and display primitives), their properties, their metamorphoses and anamorphoses and their suitability will be discussed, with illustrations on how they influence our approach to software, hardware and even determine the look of our pictures.

RESUME

Avant d'etre encapsulées dans des systèmes infographiques comme types de données, objets ou fonctions, les primitives que nous voulons ou pouvons manipuler doivent etre définies clairement. Les primitives utilisées jusqu'à présent (les primitives de modelage, géométriques et d'affichage), leurs propriétés, leurs métamorphoses, leurs anamorphoses et leur adequation sont discutées, avec des examples pour illustrer comment elles influencent notre approche du logiciel, du matériel et meme l'aspect de nos images.

KEYWORDS: Modelling primitives, geometric primitives, display primitives.

The definitions and naming of *primitives* in computer graphics has not been uniform and consistent. For the purpose of this discussion, we will call *modelling primitives* the primitives manipulated by an application program and hopefully corresponding to the user's concepts within that particular application. We will call *geometric primitives* the geometric entities to which are applied the transformations of a standard graphics package (translate, rotation, scaling, clipping, projection, shading, etc..). And finally we will call *display primitives* the primitives used by the display device to create the image. The terms *output primitives*, and *graphic primitives* have not been used in order to avoid confusion since they have been used rather loosely in the past.

Examples of modelling primitives are points, lines, polygons, spline surfaces, buildings, cars, transistors, stochastic surfaces and friends of the designer. Examples of geometric primitives are points, lines, polygons, spline surfaces, spheres, dodecahedra and fractal surfaces. Examples of display primitives are points, lines, filled polygons, pixels, dot matrices, sprites and characters.

If some of these primitives look alike to you, look again. A point as a modelling primitive can have many attributes, such as colour, velocity, potential, charm, etc.. As a geometric primitive, a point is mainly characterized by its coordinates (of the required number), even though it can be associated with a normal vector, a colour and other attributes. A point as a display primitive has integer coordinates (at the resolution of the device), and the mapping from points as geometric primitives to points as display primitives can be many to one.

The operations on primitives in a graphics

system can be partitioned into two classes: the *transformations* which transform a primitive into the same type of primitive, and the *metamorphoses*, where the primitive output is of a different type than the primitive input. It is our claim here that the failure to distinguish clearly between these types of primitives and these types of operations has obscured some of computer graphics methods, has caused our standards to be unduly influenced by the devices in use, and reduces our ability to develop freely new techniques and new primitives.

Historically Computer Graphics developed from vector displays, and their display primitives were points, lines and sometimes characters. It is interesting to note that an analogy can be made between the operands in assembly language for a given CPU and the output primitives for a given DPU (Display Processing Unit, in Foley and van Dam's terminology). The trouble here is that we standardized based on functions that map almost one to one to these primitives, forcing the application level to deal with output primitives, when it should deal with modelling primitives. This is equivalent to standardize a "high level" programming language based on some computer assembly language. This did happen to Fortran, most notably in its datatypes.

Of course computer graphics did not stand still since the TX-2. New primitives appeared. Among new modelling primitives are the parametric surfaces (Bezier and B-splines among others), stochastic models and even ellipsoids. Among new output primitives are the ubiquitous pixels, circles, filled polygons, parametric curves, convex intersections of half spaces, and stochastic surfaces. Geometric primitives did not change much, mainly because the stock of geometric entities was already considerable when Computer Graphics came upon the scene. *Rasterops* are an especially interesting case of new output primitives, because they allow new and powerful transformations, when transformations on output primitives were previously rather limited.

As an example of what can happen let us take a circle, which can start as a modelling primitive (when the user says: "give me a circle"). This circle can then become a geometric primitive (with center and radius) when the application program hands it to the graphics package. It will be transformed, clipped (to allow clipping without metamorphoses we should look at our circle as an arc of circle) and projected unto the screen. Then it will be metamorphosed into one or more output primitive. A circle if the output device has hardware circles, a "polyline" if the device only draws lines, a lot of pixels if it is a raster device. Alternately the circle as a modelling primitive can be broken into line segments, if the graphics package does not handle circles. It would then be transformed, scaled and clipped as line segments.

It is interesting to observe how these different types of primitives influence each other. We already saw a nefarious influence of display on geometric or even modelling primitives. Another example (this one benevolent) is when hardware dictates that output primitives be convex polygons (as in the case of PIXELPLANE). The temptation is strong to force modelling primitives to be similar. In the other direction, the motivation is strong to make the parametric surfaces geometric primitives (by applying the geometric transformations on the control points, but it does not always work), or even to output primitives (by building special hardware). This lead to difficulties, notably in trying to convert parametric surfaces directly to pixels as output primitives. Some approaches were more successful than others. More examples (with pictures of course) will be given in the presentation, including a description of our own efforts to turn stochastic models into output primitives.

It is our contention that a better awareness of the various primitives will lead to a better understanding of the basic operations and help focus our approaches to problems in computer graphics. But we also feel that this is not enough by itself. We have to go a level higher, and develop a theory of computer graphics which will allow us to describe and analyze computer graphics systems by treating all existing and future primitives in an abstract yet expressive manner.