Using Recursion to Describe Polygonal Surfaces

Brian Wyvill, Breen Liblong & Norman Hutchinson

Department of Computer Science, University of Calgary.

Abstract.

We describe a three dimensional graphics system called PG (Polygon Groper) and its use by artists for defining and viewing solid objects. A simple set of commands in PG are used to manipulate 3D models. The models are stored as a hierarchy of objects, where each object represents a geometrical transformation of another object or a primitive polygon. It has been found that a hierarchical structure provides a more powerful way of building and manipulating 3D objects than conventional linear structures. The object hierarchy also allows recursion to be used to describe pictures which are difficult or impossible to define in other graphics systems without considerable programming effort. Although any recursive process may be expressed iteratively, a certain class of pictures is more concisely defined and more easily visualised recursively. Many tools are available to aid the artist manufacture primitive objects. These tools generate the surface polygons to describe different solids. PG offers a number of alternative viewing algorithms, these provide the artist with successively higher quality images at the cost of correspondingly greater time for computation. PG is an integral part of the *Graphicsland System*, a film animation system under development at the University of Calgary.

Keywords: Interactive Graphics, Recursion, Graphics Languages.

Background

A survey of the basic ideas and approaches to the subject of computer art was made by Charles Csuri [Csuri 1974]. The survey notes the use of various programming techniques including; iteration, recursion, branching, nesting and subroutining. Various mathematical methods such as Robbins and Hendriks' [Robbins 1971] use of the bug problem, and Csuri's Mathematical surface representations use these elemental techniques. Although various *user friendly* 3D graphics systems exist these either require the artist to learn a general purpose high level algorithmic language, for example LOGO [Papert, 1970 and 1972] and a LISP based system [Goates 1980], or they lack suitable structures or they are special purpose systems aimed at particular pieces of hardware, for example GRAMPS [O'Donnell 1981]. Finally, other systems such as MIRA [Magnenat-Thalmann 1983] contain suitable structures for object description but lack the elegance and compactness of recursive definitions.

Recursive techniques can be used to simplify the complexity of a picture definition in a way that produces more interesting effects than the usual iterative methods. Controlled recursion was used to define 2D pictures in a series of experiments in the 1970s [Wyvill, G. 1974 and Wyvill, B. 1977] and many interesting pictures were produced (see Figure 1). The Graphics language Groper emerged from these experiments [Wyvill, B. 1982] and the basic idea of structuring pictures as recursive hierarchies has been tested by various artists. Groper enabled complex 2D pictures, based on a line primitive to be defined.

3D Polygon Groper

The latest in this series of experiments is to extend the ideas developed in Groper to 3D using planar polygon primitives. Polygon Groper is a graphics language for the construction and editing of 3D objects. An object consists of a series of edges defining a primitive planar polygon or a series of geometric transformations of other primitive or complex objects. An object may also refer to itself to produce a recursive object. For example a cube may be simply defined as follows:

Assume a primitive object called SQUARE exists as a polygon in the X-Y plane bounded by edges between vertices (0,0,0), (1,0,0), (1,1,0), and (0,1,0). The *origin* of the SQUARE is defined at the point (0,0,0).

The definition of a CUBE contains;

An instance of SQUARE at the current origin.
An instance of SQUARE rotated by 90 degrees counter-clockwise about the y-axis and with its origin at 1,0,0
An instance of CUBE rotated by 90 degrees counter-clockwise about the y-axis and then rotated 90 degrees clockwise about the z-axis

When the user wishes to view this object the definition of CUBE is entered by a recursive drawing algorithm and two instances of SQUARE are drawn; one with its origin at (0,0,0) and another rotated about the y-axis by 90 degrees and with its origin at (1,0,0). An instance of CUBE is encountered and the appropriate transformation matrix calculated. The definition of CUBE is entered recursively modified by the matrix. Note it is the axes of CUBE which are transformed. Since CUBE consists of two instances of SQUARE, the SQUAREs are then drawn in the new coordinate system. The technique adopted to terminate this recursive process is to count the number of times the procedure is entered and to stop at a specified limit, in this case three.

Although this type of definition is difficult to comprehend at first, experience with the earlier 2D languages have shown that artists very quickly become accustomed to thinking recursively, a conclusion also reached by Papert with Logo [Papert 1972]. Once the cube is defined, then objects such as a row of cubes, a row of rows of cubes or even a double helix of cubes are relatively simple to create.

It can be seen that recursion is a powerful tool when objects are defined that contain more than one reference to themselves. For example, Figure 1 shows a tree defined with GROPER in 2D and Figure 4 shows the corresponding 3D level 6 tree built from ten sided cylinder approximations. The tree is a multiply-recursive object containing four references to itself (the branches).

Tools for constructing 3D objects

PG also contains a variety of commands that enable commonly used primitive polygons and objects to be defined. For example a single command defines n-sided regular polygons. Polygon meshes can be defined given two templates with the same number of vertices. This provides an alternative method of building objects like the cube and makes it easier to define other objects such as a sphere or a torus.

Figure 1: Tree (level 10) produced with GROPER

The user does not need to know the coordinates of any particular sub-object but can refer to them using a dynamic labeling facility similar to the idea put forward in the language PDL2 [Wyvill, G.K. 1975]. For example, one could refer to the jth vertex of the ith instance of a recursive picture definition with a label. The system then does the work of placing one object relative to another.

Advantages of using hierarchical picture structures

The use of recursive hierarchies in picture definitions has several advantages. It encourages the user to maximise the symmetry and regularity inherent in the composition of a picture from sub-pictures. Another important feature of recursive structures is that objects are completely specified with very compact definitions, at the expense of computation time involved in matrix concatenation. Pictures also become easier to modify because of the modularity inherent in the hierarchical approach. This reduces the amount of time the user spends in developing a picture. For example, different space filling curves may be developed in 2-D Groper simply by changing the lowest level (i.e. primitive) component in the definition.

A third advantage of a structured hierarchy is that object coherence may be easily exploited. For example, if an object is defined as a polygon mesh then at rendering time the polygons which comprise the back of the object as seen by the viewer may be easily discarded. Another good example is a possible solution to the problem of requiring a very detailed picture database but not wanting those details when viewing from far away. If details are modelled farther down the hierarchy then for far views of an object, sub-pictures which project to less than a predefined minimum area, such as a pixel, may be replaced by a single polygon representing some averaged colour value. This leads to great savings in rendering time while preserving the integrity of the object database.

Various other advantages to this approach concerning 2D systems are detailed in JAGGIES [Wyvill 84].

The artist and PG

PG currently consists of two parts; a simple interactive textual editor for defining and modifying pictures, and a module that traverses the data structure recursively plotting the pictures. The artist interacts with PG via a keyboard with a set of simple commands. This is not an ideal user interface but was convenient to test out the fundamental ideas of using recursive structures to represent 3D objects. A more elaborate graphical user interface is currently being designed. Several photographs (Figures 2 - 7) have been included to show how PG has been used by artists to define various interesting objects. The photographs were taken on black and white film from colour images produced using Warnocks algorithm. Some image quality has been lost in this process. The picture *Tiger Lillys* (Figure 3) was exhibited at the Digicon 83 exhibition in Vancouver.

Display Algorithms

PG outputs the resulting polygons in a format that is compatible as input to a variety of display modules. These modules allow the user to view the pictures with varying amounts of realism and corresponding amounts of processing time. The following different algorithms are currently available for use with PG:

- Wire Frame
- Fuch's Algorithm
- Warnocks Area Sub-division
- Ray Tracing

The *wire frame* algorithm provides the user with the quickest means of immediately seeing a degraded image. A simple line drawing is produced in perspective showing an outline of the objects in the scene. This is used when storyboarding an animation sequence.

Fuch's algorithm provides a perspective view and removes hidden surfaces. It produces views of the same object from different eye points very quickly once the initial sorting process has been carried out, thereby allowing rapid generation of frames where the scene is the same and only the camera is moving.

Warnocks Area Sub-division algorithm has the advantage of providing a higher quality picture the more times it is allowed to subdivide. Provided subdivision extends past the pixel level then anti-aliased pictures may be produced as detailed in [Sabella 83].

Ray Tracing is the most time consuming of the algorithms available. It solves all of the above problems but may take several hours on a Vax 11/780 to produce a single frame. Another research project at the University of Calgary is attempting to solve the problem by designing and building a special purpose machine to perform the ray tracing calculations [Cleary 1983].

PG and Graphicsland

Graphicsland [Wyvill, 1983] is a suite of programs developed at the University of Calgary towards a 3D animation system. An animator can design 3D objects using PG and the object descriptions will be stored for use by other parts of the system. Several different objects can be put together in a scene and motions specified for the objects, the camera and lights. The hierarchic structure lends itself well to the idea of animating whole sub-objects or parts of an object. For example when moving a character's arm, using Graphicsland, sub-objects such as hands and fingers will automatically be moved. The system also contains a paint system which provides an artist with the ability to create backgrounds. PG objects can then be merged and anti-aliased to the painted background.

Implementation of PG

The PG interpreter manufactures a compact data-structure which represents an object hierarchy. Each object is stored as a linked list of geometric transformations of other objects or primitives. Thus a very small amount of data is used to represent a complex object. When a drawing request is received the data-structure is traversed and the polygons calculated and sent to the appropriate display program.

Several programs also produce input for PG such as ANI the animation script interpreter, and SKETCH, an architectural design tool. PG consists of approximately 5400 lines of C and runs under the Unix operating system on a VAX 11/780.

Parallelism and future development of PG

PG is able to exploit parallelism by overlapping the various processes / tasks of object creation, (user interaction), polygon generation (traversal of the data structure), and image generation (rendering). Currently these processes are active on a single processor, however work is continuing to implement the system using the JADE [Unger 1984] inter process communications. (JIPC) When this is done the object creation system, one of a series of different user interfaces which are being developed, will send JIPC messages to the process which handles updates to the data structure. It is intended that the traversal algorithm will be resident on a separate processor as will the appropriate rendering algorithm. Various algorithms, including *Ray Tracing*, *Warnocks* and **Z-Buffer** are being implemented as a set of parallel processes each working on a subset of pixels. A prototype of the parallel Ray Tracing algorithm is now running on a VAX 11/780 [see also Cleary 1983].

Acknowledgements

This work is supported by the Natural Science & Engineering Research Council of Canada. We would also like to acknowledge the students who have worked so hard on the Graphicsland system in particular Dave Maulsby, Craig McPheeters and Reddy Vatti.

References

Cleary, J. , Wyvill, B., Birtwistle G., and Vatti, R. (1983.) "Design and Analysis of a Parallel Ray Tracing Computer." *Proc. XI Association of Simula Users Conference*..

Wyvill, G.K. (1974.) "Pictorial Description Language" *Proc. DECUS* .

C. Csuri. (April 1974.) "Computer Graphics and Art, Proceedings of the IEEE."

Goates, G.B., Griss, M.L., and Herron, G.J., "PICTUREBALM : A LISP based graphics language system with flexible syntax and hierarchical data-structure." *Proc. SIGGRAPH 1980*, 93-99.

Magnenat-Thalmann, N. and Thalmann, D. (December 1983) "The Use of High-Level 3-D Graphical Types in the Mira Animation System" *IEEE Computer Graphics and Applications*, 9-16.

O'Donnell, T.J. and Olsen, A.J. (1981) "GRAMPS - A Graphics Language Interpreter for Real Time, Interactive, 3D Picture Editing and animation." *Proc. SIGGRAPH, 15* (3) 133-142.

S. Papert. (1970) "Teaching Children Thinking" *IFIPS*.

S. Papert. (April 1972.) "Twenty Things to do with a Computer" *Education Technology*.

D. Robbins, Hendriks, L., and J. Reichardt, J. (1968) "'The Computer in Art' Studio Vista: New York; From the Cybernetic Serendipity Exhibition, Studio International." *Van Nostrand-Reinholdt 1971*.

Sabella, P. and Wozney, M. (Nov. 1983) "Towards Fast Color-Shaded Images of CAD/CAM Geometry." *IEEE Computer Graphics & Applications*., 60-71.

Unger, B., Birtwistle, G., Cleary, J., Hill, D., Lomow, G., Neal, R., Peterson, M., Witten, I., and Wyvill, B.L.M. (February 1984.) "Jade: A Simulation & software prototyping environment" *Proc Soc. for Computer Simulation Conference on simulation in strongly typed languages*..

Wyvill, G.K. (1975) "Pictorial description language 2" *Proc ONLINE 75, Computer graphics*, Brunel University, Uxbridge.

Wyvill, B.L.M. (1982) "Some Recursive Techniques in Picture Description." Research Report 82/106/25, University of Calgary, Department of Computer Science.

Wyvill, B.L.M., Maulsby, D., and McPheeters, C. (1983) "Computer Animation at the University of Calgary" *Research Report No. 83/16/127*, University of Calgary, Department of Computer Science.

Wyvill, B.L.M., Levinson, D., Neal, R., and Bramwell, B. (1984) "JAGGIES: A Distributed Hierarchical Graphics System." *Proc. CIPS Session 84 Calgary*..

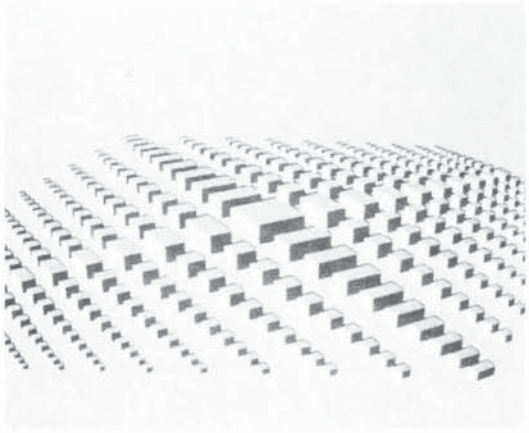

Figure 2: Row of row of cubes.

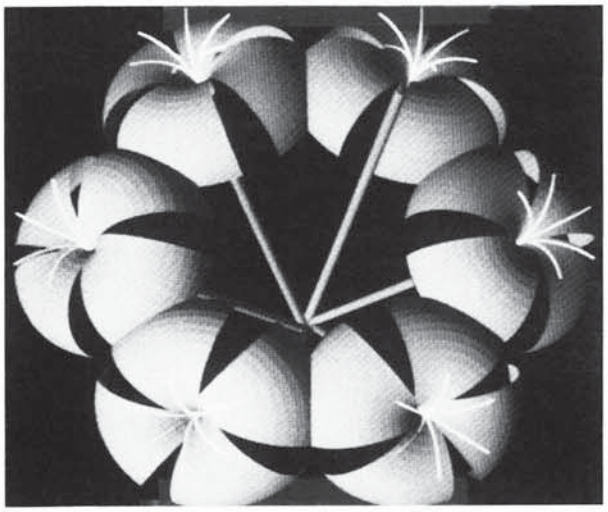

Figure 3: Tiger Lilies.

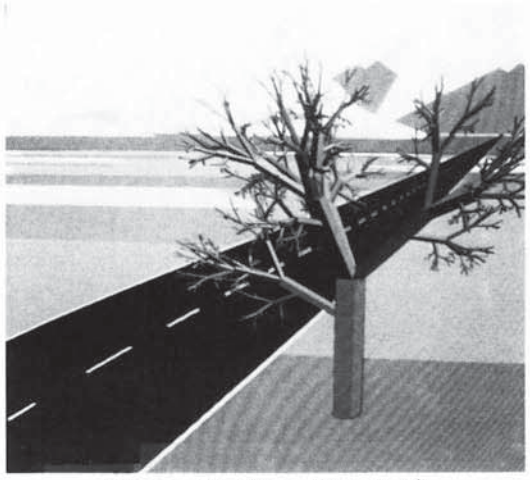

Figure 4: Tree (level 6).

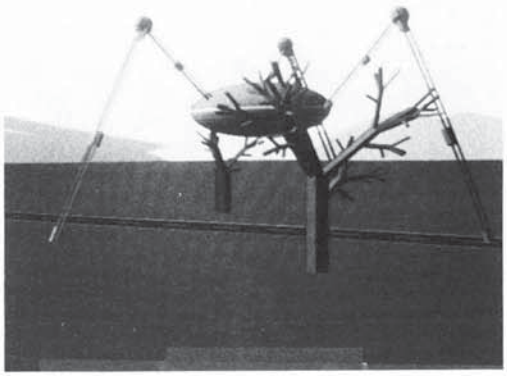

Figure 5: Martian Walk.

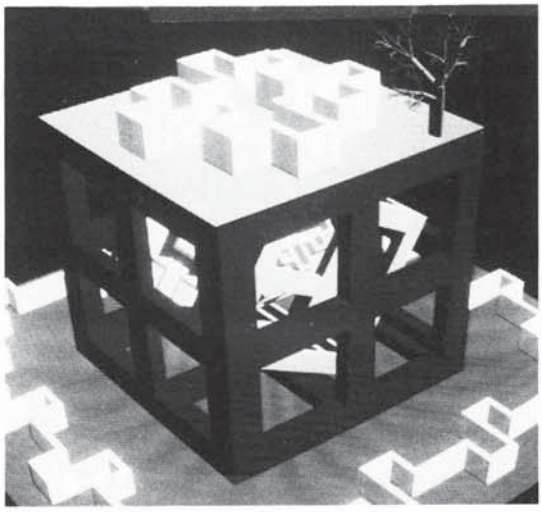

Figure 6: The Maze.

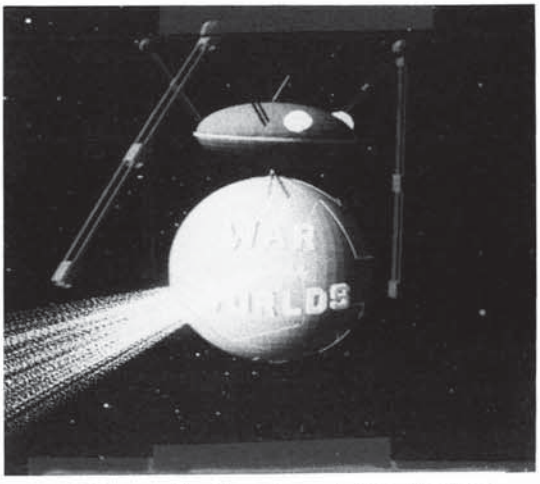

Figure 7: The War of the Worlds.