A SYSTEM FOR DESIGNING AND ANIMATING OBJECTS WITH CURVED SURFACES

M. Green

McMaster University
Hamilton, Ontario

ABSTRACT

A large amount of work has been devoted to developing algorithms for producing realistic displays of objects with curved surfaces, but very little has been published on systems which use these algorithms. One such system, called PATCHES, will be presented here.

The PATCHES system provides facilities for the creation, editing, and display of objects with curved surfaces. These operations can be performed both interactively and under program control. This system also has the ability to produce line drawing and colour shaded animation.

An overview of the PATCHES system is presented. The basic components of the system are described along with the underlying data structures. A discussion of the design philosophy is also included.

RÉSUMÉ

Un travail volumineux a été effectué dans le domaine des algorithmes, en vue de produire des affichages réalistes des objets à surface courbe; mais très peu a été publié au sujet des systèmes utilisant ces algorithmes. Un de ces systèmes, appelé PATCHES, sera ici présenté.

Le système PATCHES procure des moyens pour la création, l'édition et l'affichage des objets à surface courbe. Ces opérations peuvent être effectuées à la fois de manière interactive et sous le contrôle d'un programme. Ce système a de plus l'habileté ou la capacité de dessiner une ligne et de produire une animation au moyen de la nuance des couleurs.

Nous présentons une vue d'ensemble du système PATCHES. Les composantes de base du système sont décrites de concours avec les structures informationelles qui les sous-tendent. Vous trouverez aussi, ci-incluse, une discussion sur les principes généraux, ou philosophie du "design".

1. Introduction

PATCHES is a system for both the modeling and display of objects with curved surfaces. The basis of this system is a data structure for representing objects with curved surfaces and a library of routines for manipulating them. A number of programs have been built on top of this data structure. These programs can be used to design, animate, manipulate, and display objects. Since a common data structure and library of manipulation routines have been used throughout these programs they can easily be combined to perform more complex functions. The two main aims of the PATCHES system are to show how one data structure can be used as the basis for a modeling and display system, and to determine what software components should be included in such a system.

The PATCHES system as currently implemented resides on a PDP 11/45 computer running the UNIX operating system [Ritchie and Thompson 74]. The graphics devices used by this system are a Three Rivers Computer Corporation GDP vector display, a 256x256x8 raster display, and a digitizing tablet. The PATCHES system has been designed to be relatively independent of the display devices used. To fully support PATCHES a vector display, a raster display, and a pointing device are required. The characteristics of these devices are largely unimportant. At the present time PATCHES is being adapted to run on an LSI 11/23 computer.

PATCHES is mainly a modeling system. Its main purpose is to provide facilities for modeling objects with curved surfaces. Where an object may be a simple geometrical object like a sphere or torus, a more complex object such as a space craft or an automobile, or a surface generated by a scientific experiment or model. Some of the operations which can be performed on these objects are animation, display, and transformation. PATCHES has been used in a number of different applications. One of these applications is data display. The usual format of the data to be displayed is a collection of three dimensional points. PATCHES routines are used to fit surfaces to these data points. The investigator can now use the PATCHES display programs to view his data. In order to view the surfaces from different angles they can either be transformed or animated. Even more important, several surfaces can be superimposed to see where they intersect. By making several of the surfaces transparent and displaying them together it is easy to see how they are interrelated.

Another application of the PATCHES system is computer animation. In particular the animation of three dimensional objects. The approach used here is to first develop a library of objects to be used in the animation. The techniques used to construct these objects are outlined in Section 3. The objects are animated by applying transformations to them in each frame of the film. These transformations are specified by a script supplied by the animator. The PATCHES program used for animation is described in Section 5.

A third application of the PATCHES system is the production of colour images. The PATCHES structure used in the creation of these images is called a scene. A scene consists of a number of objects. These objects are constructed in the same way as the objects used in animation. The display program used to produce these images uses a scene structure and a background image as input. The output is a raster image to be displayed on a colour raster display or plotted on a microfilm recorder.

2. Overview of PATCHES

The data structure used in PATCHES is outlined in fig. 1. The top level of this data structure is called the scene level. This level represents a still scene made up of one or more objects. Each scene structure has an object table listing all the objects in the scene. The location of the viewer and the direction of the illuminating light sources are also stored in the scene structure.

The next level in the data structure is the object level. Each object structure corresponds to one object. This structure contains a table of the surface patches which define the shape of the object. The object structure also contains texturing information and the light model parameters for the object.

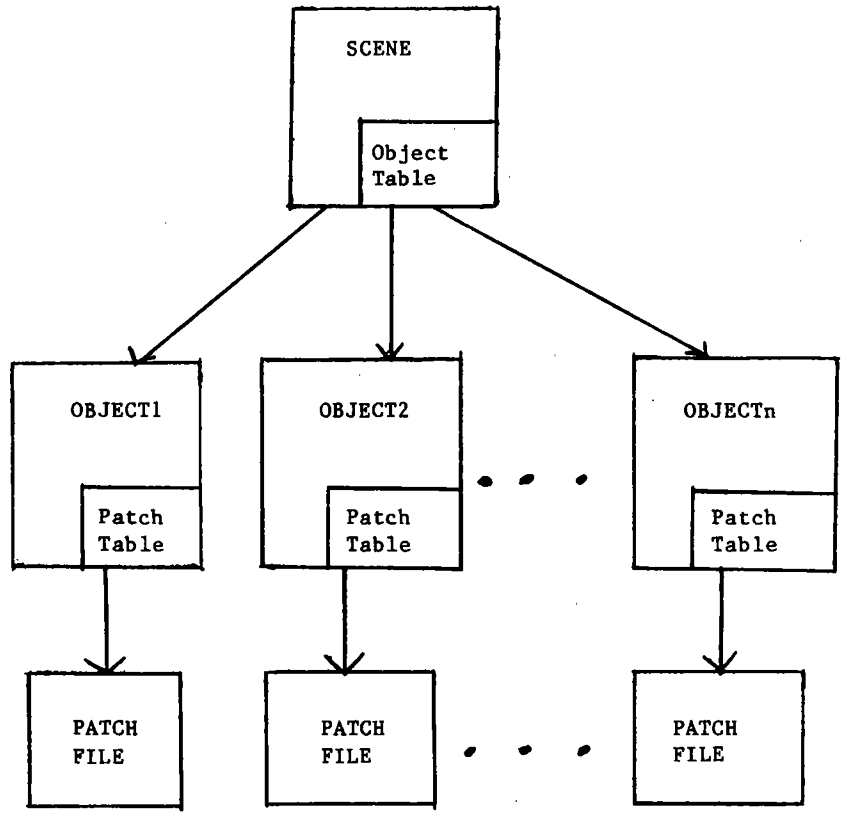

Fig. 1 Outline of PATCHES Data Structure

structure used in PATCHES is hierarchical in nature. At the top of the hierarchy is the scene structure. This structure points to the object structures representing the objects in the scene. Each of these object structures points to the file where its patches are stored and contains a table describing each of these patches. Each scene and object structure is stored on a separate file. The pointers to these structures are the names of the files they are stored on.

The two major features of this data structure are separate structures for scenes, objects, and patches and storing these

The lowest level in the data structure is the bicubic surface patch. All the patches for an object are stored on one file. There is a table in the object structure which contains the colour and transparency of each patch. A patch is identified by a patch number. The first patch in an object is assigned the number zero and subsequent patches are assigned the next available integer. The patch number is used to index the patch table in the object structure and to retrieve a given patch from the patch file. A large number of patches are required to define the shape of some objects. Because of this patches are normally stored on disk and transferred one at a time to main memory when they are required.

structures on separate files. This has several advantages over storing the same information in one structure. The first advantage is that it divides the information into logical units. In animation we want to deal with objects and object instances. By storing object information in a separate structure this information can be easily accessed. In data display and image production applications the scene structure is also important since it governs the contents of an image. Since all the scene information is grouped together the structure of a scene can be changed without effecting the objects in it.

The second advantage of this structure is that it allows scenes and objects to share information. An object can appear in more than one scene and a patch file can be used by more than one object. This sharing of data saves on disk space. It also adds to the consistency of the stored information. Two identical objects can be represented by the same object structure. Any operations performed on one object will automatically be reflected in the other.

Another advantage of this structure is that it accomodates different views of the same data. For example, the same patch file can be used by several object structures. Each of these objects can have different colour, light model, and texturing information. These objects will appear different even though they have the same basic shape.

The PATCHES data structure can be manipulated either interactively through the use of editing programs or under program control by calling routines in the PATCHES library. The user written programs are used mainly for modeling individual objects. The modeling facilities provided by PATCHES are outlined in Section 3.

The interactive programs are used mainly to manipulate existing objects and combine them into scenes. These program are divided into two groups. The first group consists of a number of small programs which perform a single editing or manipulation function. These utility programs can be used to quickly make small changes in the data structure. They can also be combined by the user to perform more major changes. By using UNIX shell files these collections of commands can be saved and called up at any time. The second group of programs consists of interactive graphical editors capable of changing most of the components of the data structure. These programs can display wire frame representations of the objects in a scene along with the other values in the data structure. The shape of an object can be changed by the use of geometrical transformations. The effect of these transformations can be seen on the display. The values of most of the components of the data structure are displayed on a menu and can easily be changed by the user. More details on these programs will be presented in Section 4.

The PATCHES system has programs for object display and animation. These programs can produced either wire frame displays to be used with vector displays or full colour shaded images to be used on raster displays. These programs will be discussed in Section 5.

3. Object Modeling

Bicubic surface patches are used to represent the shape of objects in the PATCHES system. There are three reasons for choosing this representation:

1) A large number of objects can be modeled using bicubic surface patches, including those with curved surfaces.
2) Algorithms exist for producing very realistic images of objects represented by surface patches.
3) The manipulation of objects made up of surface patches is relatively straight forward.

A bicubic surface patch is made up of three bicubic polynomials, one for each of the x, y, and z coordinates. These polynomials have the following form.

$$x(u,v) = a_{33} u^3 v^3 + a_{32} u^3 v^2 + \ldots + a_{00}$$

$$y(u,v) = b_{33} u^3 v^3 + b_{32} u^3 v^2 + \ldots + b_{00}$$

$$z(u,v) = c_{33} u^3 v^3 + c_{32} u^3 v^2 + \ldots + c_{00}$$

The independent variables, u and v, are called patch parameters. As the patch parameters vary from 0 to 1 the values of these polynomials give the (x,y,z) coordinate of the corresponding point on the surface of the object The surface of the object will typically be divided into a number of patches each represented by a bicubic surface patch. The surface patches for an object will normally not overlap, but adjacent patches will meet at their common boundary. Sixteen parameters are required to define the shape of a bicubic surface patch. These shape parameters are typically points on or near the surface of the object, or derivatives of the surface. The exact nature of the shape parameters and how they are used in determining the coefficients of the bivariate polynomials depends upon the fitting technique used. The PATCHES system provides fitting routines for most of the common patch fitting techniques.

Regardless of the patch fitting technique used the result will always be a bivariate polynomial. Because of this PATCHES uses the coefficients of the bivariate polynomial to represent the patch instead of the shape parameters used to define it. This means that once a patch has been fitted it can be manipulated in a standard way. A patch is stored as three 4x4 arrays of coefficients.

One approach to object modeling in the PATCHES system is to write a program using the modeling routines in the PATCHES library. The best way of describing this technique is to outline the structure of a typical modeling program. This structure is shown below.

```
create scene structure.
create object1 structure.
add object1 to scene.
loop
  calculate shape parameters for patch
  fit patch
  add patch to object1
until all object1 patches generated.
create object2 structure.
add object2 to scene.
loop
  calculate shape parameter for patch.
  fit patch.
  add patch to object2.
until all object2 patches generated.
               ...
create objectn structure.
add objectn to scene.
loop
  calculate shape parameters for patch.
  fit patch.
  add patch to objectn.
until all objectn patches generated.
write complete scene structure to disk.
```

The first thing an object modeling program does is create the scene structure the objects are to be placed in. This is done by calling the routine "crscene". Next the object structure for the first object is created and added to the scene structure. This is accomplished by calling the routines "crobject" and "addobject". The modeling program then computes the patch shape parameters for each of the patches in the first object. After the shape parameters for a patch have been computed a patch fitting routine is called. The patch fitting routine used will depend upon the shape parameters and the patch properties desired. These routines return a bicubic surface patch which is added to the object structure. The routine "addpatch" can be used for this purpose. This routine writes the patch onto the patch file associated with the object and initializes its patch table entry. These operations are repeated for each object in the scene. At the end of the program the routine "putscene" is called to write the entire data structure to disk.

There is a transformation mechanism associated with the patch fitting routines. The routines "scale", "rotate", and "translate" can be used to build up a transformation matrix. This transformation matrix is applied to the bicubic surface patch before it is returned to the user. PATCHES maintains a stack of transformation matrices to enable the user to save a transformation matrix for later use.

The transformation matix used in PATCHES is a 4x4 matrix with the same format as the transformation matrices used in vector graphics packages. Bicubic surface patches do not transform the same way as points do. A matrix multiplication can be used to scale and rotate a patch but it cannot be used to translate it. In PATCHES the transformation is performed in two steps. In the first step the upper 3x3 sub-matrix of the transformation matrix is extracted and each coefficient in the patch is multiplied by it (recall that each coefficient has three components). In the second step the first three entries in the last row of the matrix are extracted and added to the constant coeffecent in the patch. These three entries correspond to the translation component of the transformation.

The user can define a three dimensional window through the use of the "window" routine. All objects created in the program must lie within this window. When a patch is written to disk it is transformed to a standard space. This space is the unit cube. When a patch is read from a patch file it is transformed to the current window. The user is unaware of these transformations. Thus, the stored representation of a patch is independent of the coordinate system in use when it was created.

4. Object Manipulation

The programs described in this section are used to edit existing scenes and objects. Unlike the approach taken in object modeling these operations are usually performed interactively by editing programs. Two types of editing program are used in the PATCHES system. The first type are special purpose editing programs used for changing a small number of parameters. The second type are general purpose editing programs capable of changing most of the parameters in the data structure.

Each of the special purpose editing programs performs a well defined atomic editing function. Examples of these programs are "colour" which sets the colour of an object, "texture" which sets the texturing parameters for an object, and "trans" which can be used to transform an object. These programs prompt the user for the new values of the data structure components they change. The main purpose of these programs is to facilitate making small changes to the data structure. They are easy to invoke and perform their operations very quickly. By using UNIX shell files these editing programs can be combined, named, and saved. Shell files can have parameters. These parameters are available to the programs within the shell file. The shell file mechanism can be used to construct special purpose editing programs out of the atomic editing programs. This is one of the benefits of the common data structure and atomic editing programs.

The PATCHES general purpose editing programs have been evolving over the past three years. The most recent of these programs is called "edit" [Griffin 80]. The main purpose of the general purpose editing programs is to provide a mechanism for editing the complete PATCHES data structure. These programs are used for more extensive editing operations than the previous set of programs.

There are two basic categories of editing operations performed by the general purpose editing programs. The first category involves changing scalar values in the data structure. These values include the light model parameters, position of the viewer, and the transparency of patches. The second category of operations involves major changes to the data structure. These operations include creating new scenes and objects, adding and removing objects from scenes, and transforming objects. The operations in these two categories are performed by different mechanisms.

A menu is used in the first category of editing operations. This menu contains the values of all the components of the data structure. To change a value displayed on a menu the user points at it and enters the new value. This menu is divided into a number of panels. There is one panel for each scene and object the user is working with. The editor has commands for specifying the current scene and object. Only the menu panels for the current scene and object are displayed. All other scene and object panels are not.

The second category of editing operations makes use of a wire frame display of one or more objects. This display can be used to visualize how the objects will appear in a movie or on a raster display. The operations in this category are at a higher level than the operations in the previous one. This part of the editor can be used to create new scenes. The "create scene" command is used to create a new scene structure. This scene structure will have default values for its components and will have no objects. Existing objects can be added to this scene by the use of the "add object" command. The object to be added to the scene is selected from a menu of all objects known to the editor. The other values in the scene structure can be set by the using the first category of editing operations. Objects can also be added and removed from existing scene structures.

Similarly, new objects can also be created by this editor. The "create object" command is used to create an empty object structure. The "addpatch" command is used to add patches to the new object. These patches are taken from the patch file of an existing object. Patches can also be added to an existing object. This editor cannot be used to create new patches.

This editor maintains a stack of

transformation matrices. A transformation matrix is constructed by a series of "scale", "rotate", and "translate" commands. Any of these transformation matrices can be applied to an object. There are two modes of object transformation. In the first mode the transformation is only temporary. The transformed object is displayed for the user. This mode is used for exploring different transformations. The second mode of transformation permanently changes the object. This mode is used once the user has decided upon the transformation he wants.

There are several utility commands in the editor. One of these commands is used to indicate the scenes that will be used in the editing session. There are other commands for controlling the display. They can be used for setting the position of the viewer's eye and the number of grid lines to be drawn for each patch.

5. Display and Animation

There are two display programs in the PATCHES system. One is used for producing colour raster images and the other one is used for animation.

The program "display" is used for producing colour raster images. The input to this program is a scene structure and background image. The output is a raster image containing a display of the objects in the scene superimposed on the background image. This output raster can be of any size and can have up to 30 bits of colour. The display program has been designed so that the size of the raster image it produces is a compile time parameter. This allows the same program to produce images for both raster displays and microfilm plotters.

The basic algorithm used by display is the Catmull patch subdivision algorithm [Catmull 75]. A Z-buffer is used for hidden surface removal. The display program uses two large arrays, one for the Z-buffer and the other for the raster image. Ideally these arrays would be stored in a frame buffer. This approach has not been taken here since large frame buffers do not exist on most systems. Instead of using a frame buffer we simulate one using disk files. The display program only keeps a portion of the Z-buffer and image raster in memory at any one time. Segments of these two arrays are swapped in and out of memory when they are required.

Display uses a standard light model in its intensity and colour calculations [Newman and Sproull 79]. It can also perform standard texturing and transparency calculations. There is a dithering algorithm built into this program to be used when producing images for raster displays with a limited colour range.

The PATCHES animation program is based on the use of scripts. A script specifies the objects involved in the movie, how they change over time, the frames they appear in, and control information for the animation process. The script is prepared by the user using a standard text editor.

Three types of statements can appear in a script. The object definition statement is used to name the objects used in the animation. This statement creates an instance of an object previously created by a modeling program. The format of this statement is

```
object object-instance is scene-name / object-name ;
```

The keywords in this statement are "object" and "is". The name of the object instance created is "object-instance". This object instance is a copy of the object "object-name" in the scene "scene-name". There may be several instances of the same object in a film.

The draw statement is used to indicate the frames an object instance is to appear in and the transformations to be applied to it. The format of the draw statement is

```
draw object for transform1 to transform2 in integer1, integer2 ;
```

The keywords in this statement are "draw", "for", "to", and "in". The object instance "object" is drawn in frames integer1 to integer2. In the first frame of this sequence the transformation named transform1 is applied to the object instance, and in the last frame of the sequence transform2 is applied. In the frames between integer1 and integer2 a linear interpolation of the two transformations is applied.

The transformation statement is used to define transformations. The format of this statement is

```
transform transformation-name is ( exp exp exp exp exp exp exp exp exp) ;
```

The keywords in this statment are "transform" and "is". This statement defines the transformation "transformation-name" to be the nine expressions within parenthesis. These nine expressions give the translation, rotation, and scale components of the transformation matrix. The expressions can contain constants, arithmetic operators, and the corresponding components of other transformations. By using the components of another transformation one transformation can be defined in terms of another.

The animation program can produce films in two different formats. In the first format wire frame drawings are used to represent the objects. A vector device is used to view a movie in this format. This format is used when a script is being debugged since it can be generated relatively quickly. Once the script has been debugged a movie in the second format is produced. In this movie format full clour shaded representation of objects are produced. The individual frames in this format are raster images which can be plotted on a microfilm recorder. The "display" program is used to produce these raster images. The generation of this movie format requires much more computer time than the other format. For this reason it is not used in debugging scripts.

6. Conclusions

The major components of the PATCHES system have been presented along with the data structure used in this system. A number of applications of this system have also been discussed. The data structure presented in Section 2 has been used in all programs in the PATCHES system. There are two major advantages to using this common data structure. First, all the program in the system can be used together. No conversion programs are needed to convert the output of one program to the input format required by another program. Second, since all the programs use the same data structure they can share the routines which manipulate it. This has saved a considerable amount of programming time.

Acknowledgements

Most of the PATCHES system was developed at the Computer Systems Research Group of the University of Toronto. The Dynamic Graphics Project within this group provided the ideal environment for developing this system. I would like to acknowledge the support and assistence of Dr. R. Baecker, Bill Buxton, David Galloway, Sanand Patel, and Bill Reeves.

References

[Catmull 75] Catmull E., "Computer Display of Curved Surfaces", Proceedings of the IEEE Conference on Computer Graphics, Pattern Recognition, and Data Structure, May, 1975.

[Griffin 80] Griffin S., An Interactive Patch Manipulator and Editor, BASc Thesis, Division of Electrical Engineerin ng, University of Toronto, 1980.

[Newman and Sproull 79] Newman W., and R. Sproull, Principles of Interactive Computer Graphics, McGraw-Hill, 1979.

[Ritchie and Thompson 75] Ritchie D., and K. Thompson, "The UNIX Time-Sharing System", CACM, vol.17, no.7, 1974.