LIBTEX - A Software Toolkit for Texture Synthesis in Computer-Generated Images

MARCELO WALTER

Instituto de Informática - UFRGS
Porto Alegre - Brasil

Department of Computer Science
University of British Columbia
Vancouver, BC, Canada, V6T 1Z2
marcelow@cs.ubc.ca

Abstract. The use of texture techniques in computer graphics may significantly improve the realism of the computer generated images. At the same time texture techniques are not able to implement in a straightforward way. Here we present an application independent library of C written routines named LIBTEX which allows the inclusion of texture effects in any photorealistic rendering system. The available functions and the architecture of the library are described as well as some examples stressing the flexibility and easiness of use of LIBTEX.

1 Introduction

A major research goal in computer graphics is the improvement of realism in computer-generated pictures. The final goal is ***photorealism***, that means, a computer-generated picture that depicts the world as a photograph. Many advances in this area have been made possible by adding in the computer-generated images the visual information known as texture.

Texture Mapping [Catmull (1974)][Catmull (1975)], Bump Mapping [Blinn (1978)] and Solid Texture [Peachey (1985)][Perlin (1985)] are well known texture techniques (for a comprehensive survey on texture literature see [Walter (1992)]).

Despite the research advancements and the importance of texture to improve realism, the use of texture techniques is not as general as one could expect. The main reason is that the texture algorithms are usually very specific and are not able to implement in a straightforward way. In some classes of applications the implementation cost of texture techniques is too high when compared to the desired realism and then this realism is neglected. As another evidence, most rendering educational systems do not implement texture synthesis.

In this way we have designed and implemented a library of application-independent texture functions named LIBTEX, addressing some of the problems above mentioned. LIBTEX provides a set of functions to add the texture effects in computer synthesized images. These functions can be seen as a software toolkit which allows the inclusion of texture effects in any application where the final goal is photorealism. The main goal in the design of this library was to construct a flexible and easy-to-use group of C functions that would perform all of the necessary tasks for adding texture effects in computer generated pictures.

Texture techniques are well suited to be managed as a special group inside an integrated system for image synthesis. This was first proposed in [Carey-Greenberg (1985)], where a series of texture mapping techniques were added to the testbed imaging system existing at the Program of Computer Graphics at Cornell University. LIBTEX was strongly inspired on that idea, but it has a significant improvement which is its application independent approach.

This paper is organized as follows: the next section describes LIBTEX more carefully, one example explaining its use is then presented and finally some conclusions are gived.

2 LIBTEX Description

2.1 LIBTEX Architecture

LIBTEX (LIBrary TEXture) is a set of application-independent routines which provides texture effects for computer-generated images. The available techniques are texture mapping [Catmull (1974)], bump mapping [Blinn (1978)] and some solid texture effects [Peachey (1985)][Perlin (1985)]. Together, these three techniques address most of the needs in a photorealistic rendering system. Details of these techniques can be found in [Walter (1991)] and in the above cited papers. The implemented routines are organized into three general categories which correspond to the three texture techniques above cited.

The use of the routines is *occurrence-oriented*, that means, the user has to define occurrences where some parameters are set. A different occurrence is needed for each different desired texture effect. The parameters inside one occurrence define the way it will be used. For example, in a texture mapping occurrence, the name of the texture map is a mandatory parameter which defines where the texture mapping algorithm will look for the texture information.

After defining an occurrence, the user receives an *occurrence identifier number* which must be used inside the routines calling. The occurrence identifier number is the link between the occurrence parameters and the techniques themselves. The general flow of execution for the first two group of routines - texture mapping and bump mapping - is illustrated on figure 1.

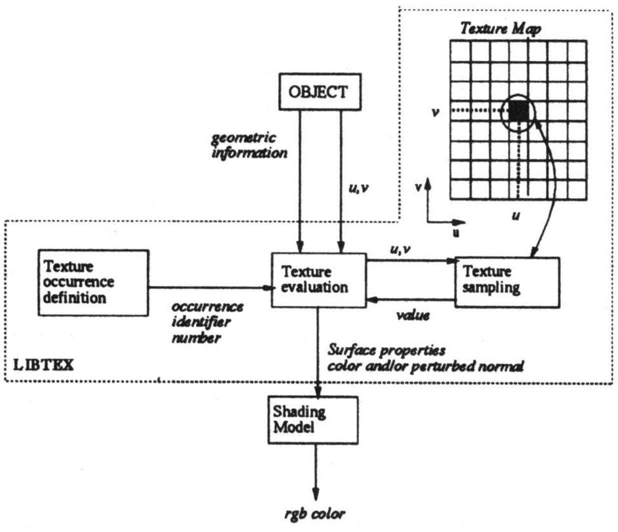

Figure 1: Texture Evaluation (texture mapping and bump mapping) inside LIBTEX

First the user shall define the texture occurrence, receiving the occurrence identifier. Then the texture evaluation procedure will take place. The *Texture Evaluation Module* performs the specific texture algorithm for each texture type. This module receives the parametric coordinates of the location of the point to be rendered, the occurrence identifier number and general geometric information necessary for the texture evaluation. For example, the bump mapping function requires the unperturbed normal in order to generate the perturbed normal.

The parametric coordinates are then sent to the *Texture Sampling Module*. The sampling process is responsible for retrieving the texture value at a given *u, v* location in the texture map. Thus, the basic retrieve operation is a point sampling. This texture value has different meanings according to the specific texture algorithm. For texture mapping this value is a pointer to a *rgb* color in a color texture file. For bump mapping this value express the surface displacement along the normal vector direction. Different maps for each different desired effect will be set by the user when the occurrence is opened.

The final result from the *Texture Evaluation Module* can be the rgb color for the point being rendered (i.e., a texture color) and/or the perturbed normal. This information can finally be usevd in a shading model when the final rgb value for the pixel will be find.

It is assumed that the objects being rendered are represented in parametric coordinates. There is no automatic transformation from geometric object space (x,y,z) into the parametric space (u,v) of the object. There is a mapping from the parametric object space into the parametric texture space.

The last group of routines perform the solid texture functions. The evaluation flow is slightly different from the previous described. Here the object's geometric world coordinates are used directly as an input information to the evaluation module as illustrated on figure 2.

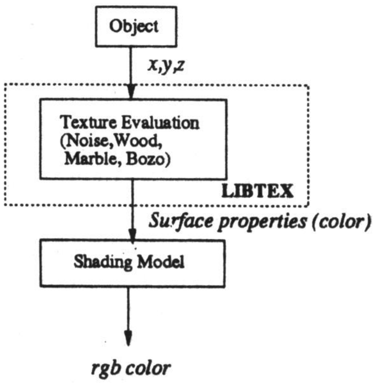
Object
x,y,z
Texture Evaluation (Noise,Wood, Marble, Bozo)
LIBTEX
Surface properties (color)
Shading Model
rgb color

Figure 2: Solid Texture Evaluation inside LIBTEX

The evaluation module then perform the specific solid texture technique desired by the user. The available solid texture functions are `noise` as cited in [Perlin (1985)], `bozo`, `wood` and `marble` effects, which are functionally composed from the basic noise function. The produced information - rgb color for the point being rendered - can be used in a shading model to find the final rgb value for the pixel.

One should bear in mind the modular characteristic of LIBTEX, which is supposed to be used together with other functional processes that form the visualization pipeline in a photorealistic rendering system.

2.2 Functions

The available texture techniques are implemented as a group of functions which perform the texture algorithms. These functions can be functionally categorized into three general categories: *initialization routines*, *evaluation routines* and *finalization routines.*

The *Initialization Routines* initialize any necessary texture information and assign a occurrence identification number for later use. The initialization routines search for the defined texture maps and open the texture occurrence. There are four initialization routines: **bumpopen**, which opens a bump mapping occurrence; **initializenoise**, which initializes the three-dimensional array that defines noise; **textopen**, which opens a *user-defined* texture mapping occurrence; **textopenRGB**, which opens a *rgb* texture mapping occurrence.

The distinction between **textopen** and **textopenRGB** comes from the distinction between the two possibilities of texture maps to be used with the texture technique. The **textopen** function uses a *user defined texture map* which is a texture map where the texture values are set up by the user in a indirect way. For example, if it is necessary a red and white chessboard pattern, the user will define two files as illustrated on figure 3.

```
0808
00001111
00001111
00001111
00001111
11110000
11110000
11110000
11110000
```

chessboard.map file

```
255 255 255
255 0   0
0   0   0
......
0   0   0
```

redwhite.color file

Figure 3: Texture Map file and Color file

The *chessboard.map* file contains the pointers to a color file, in this case the *redwhite.color* file. The link between the map and the associated colors is established when the occurrence is opened. Here the "zeros" from the *chessboard.map* file are pointers to the first entry into the *redwhite.color* file, which is the white color in rgb coordinates (r=255;g=255;b=255). The "ones" from the *chessboard.map* file are pointers to the second entry into the *redwhite.color* file, which is the red color in rgb coordinates (r=255;g=0;b=0). With this indirection one can easily change the color of the chessboard pattern. The numbers in the texture map may vary from *0* to *255*, which allows user defined maps of *256* different colors. All color files have a lenght of 768 bytes, which correspond to the 256 possible rgb colors for a user defined texture map.

On the other hand, the **textopenRGB** function uses a *not user defined texture map* which is a texture map whose texture information comes from a scanned picture, a digitalized image or even a prior rgb synthesized image. In this case the user has no control over the final texture effect.

The **initializenoise** function defines the noise values in the integer coordinates of the solid texture cube. For the non-integer coordinates the values of noise are obtained by cubic interpolation.

The *Evaluation Routines* perform the specific texture algorithm for each texture type. There are seven evaluation routines: **bozo**, which access the iso-values of noise and sets a specific color for each region which delimits the iso-values; **bumpmap**, which performs the bump mapping technique; **map**, which performs the texture mapping technique for user defined texture maps;**mapRGB**, which performs the texture mapping technique for not user defined texture maps; **marble**, which performs marble-like color patterns; **noise**, as defined in [Perlin (1985)]; **wood**, which performs wood-like grain pattern.

For the **bozo**, **marble**, **noise** and **wood** functions it is necesssary the prior calling of the **initializenoise** function, since these functions made use of the noise array which is initialized by the **initializenoise** function.

Besides the file's name where the texture map is defined, the **map** and **mapRGB** functions need as an input parameter the number of times the texture map will be replicated over the object's surface. The **bumpmap** function needs the partial derivatives of the parametric function which describes the object being rendered. The user can controll as well the number of times the bump map will be replicated over the object's surface.

The *Finalization Routines* close the previous opened texture occurrences and atualize the current occurrence identification number. This is done since there is a limit on the number of possible different texture occurrences. A full explanation of LIBTEX functions can be found in [Walter (1992a)].

2.3 Implementation Notes

The LIBTEX memory management is being done static and that is the reason why there is a maximum number of occurrences. Also, the different maps are not being read into the memory since the total size of the different possible maps at a given time could easily overflow the memory space. Thus, the time required for the texture evaluation is directed related to this constraint.

The different maps - bump map, user defined texture map - and the color files are all written as a colection of **unsigned char** values. Each map has a header which express the map number of rows and lines. On figure 3, for example, the *chessboard.map* file has a header of four bytes expressing that the file has 8 rows and 8 lines.

The LIBTEX routines are written in standard C language and have been used within Sun workstations environment. The source code shall be linked together with the application program which is supposed to use the LIBTEX routines.

3 Conclusions

We have presented an application independent group of routines named LIBTEX which allows, in a easy way, the inclusion of texture effects in any photorealistic rendering system. The use of LIBTEX gives an easy and fast access to a wide range of different texture techniques: texture mapping, bump mapping and some solid texture effects.

Some experiences with the use of LIBTEX have demonstrated that the time required to add a texture effect may vary from some minutes to a few hours, depending on the basic rendering technique, on the shape complexity of the objects and on the complexity of the scene. When the objects have an easy parametrization like a sphere for example, the texture effects can be added in a matter of minutes.

LIBTEX is easily expandible due to its group of routines characteristic. New texture techniques can be included as well as improvements on the already implemented routines can be made effective. In the near future we plan to overcome the following restrictions of LIBTEX:

- the necessity that the objects be represented in parametric coordinates,i.e, there is not an automatic mapping from the geometric object space to the parametric texture space;

- the point sampling. Sometimes this kind of sampling can result in high aliasing artifacts which are undesirable;

- the static memory allocation routines, i.e., the number of different occurrences is limited and the different maps are not read into memory.

4 Acknowledgements

This work was developed while the author was as an associated researcher at the Graduate Program on Computer Science at "Universidade Federal do Rio Grande do Sul" in Porto Alegre - Brazil . The support of CNPq is gratefully acknowledged. The author would like to express his thanks also to his adviser at UFRGS, Prof. Anatólio Laschuk.

5 References

[Blinn (1978)] BLINN, J. F. Simulation of Wrinkled Surfaces. **Computer Graphics**, New York, v.12, n.3, p.286-292, Aug. 1978.

[Carey-Greenberg (1985)] CAREY, R.J. & GREENBERG, D.P. Textures for Realistic Image Synthesis. **Computer & Graphics**, Oxford, v.9, n.2, p.125-138, Apr. 1985.

[Catmull (1974)] CATMULL, Edwin E. **A Subdivision Algorithm for Computer Display of Curved Surfaces**. Utah:Department of Computer Science, University of Utah, 1974. (PhD dissertation, Technical Report UTEC-CSc-74-133).

[Catmull (1975)] CATMULL, E.E. Computer Display of Curved Surfaces. In: CONFERENCE ON COMPUTER GRAPHICS, PATTERN RECOGNITION and DATA STRUCTURES, May 14-16, 1975, Los Angeles. **Proceedings**...New York: IEEE, 1975. p.11-17.

[Peachey (1985)] PEACHEY, Darwyn R. Solid Texturing of Complex Surfaces. **Computer Graphics**, New York, v.19, n.3, p.279-286, July 1985.

[Perlin (1985)] PERLIN, K. An Image Synthesizer. **Computer Graphics**, New York, v.19, n.3, p.287-296, July 1985.

[Walter (1991)] WALTER, Marcelo. **A Obtenção de Texturas na Síntese de Imagens Realísticas num Ambiente Limitado**. Porto Alegre:CPGCC da UFRGS, Jan. 1991. (M.Sc. Thesis)

[Walter (1992)] WALTER, Marcelo. **A Cross Indexed Guide to Texture Literature**. Porto Alegre:CPGCC da UFRGS, Jun. 1992. (Research Report)

[Walter (1992a)] WALTER, Marcelo. **LIBTEX - Uma Biblioteca para Síntese de Texturas em Imagens de Computação Gráfica**. Porto Alegre:CPGCC da UFRGS, July 1992. (Research Report)