computer Graphics: Computer Rendering Algorithms

Abstract

One concern of the computer graphics community has been the efficiency of rendering algorithms.

Buyers faced with a variety of hardware graphics accelerators would like to know the average display rate of each machine. In such fields as ray tracing, researchers continue to explore which is the fastest way to find the closest intersection point for arayanda set of primitives. The pyramid. The problem faced by these other people involved in computer graphics is a lack of standards.

Within computer graphics, the field of predictive rendering is concerned with those methods of image synthesis that yield results that do not only look real but are also radiometrically correct renditions of nature, i.e. which are accurate predictions of what a real scene would look like under given lighting conditions.

In order to guarantee the correctness of the results obtained by such techniques, three stages of such a rendering system have to be verified with particular care: the light reflection models, the light transport simulation, and the perceptually based calculations used at display time.

1. Introduction

In recent years, a lot of work has been published in the field of photorealistic computer graphics concerning more accurate rendering algorithms, more detailed descriptions of surfaces, more realistic tone mapping algorithms, and many other improvements for the process of rendering photorealistic images. Unfortunately, there have been comparatively few attempts on verifying all these algorithms in practice. In the field of photorealistic rendering, it should be common practice to compare a rendered image to measurements obtained from a real?world scene.

This is currently the only practicable way to prove the correctness of the implementation of a true global illumination algorithm. However, most rendering algorithms are just verified by visual inspection of their results.

throughout this report.I will dive deeper in different rendering algorithms and their methodologies.

2. Rendering Algorithms

Rendering is the process of converting the abstract representations of geometric entities into the appropriate color values in a rendered image (e.g. a rendered image of a cube below).

2.1. Scan-line:

The most commonly used rendering algorithm is the scan-line algorithm. Any digital image is composed of a 2D grid of the smallest picture elements called pixels. A scan line is each row of pixels. The can-line algorithm looks at each pixel, one after the other, scan line by scan line, and calculates the color that pixel should be rendered. The color of each pixel is computed using the color and other surface characteristics of the surface visible from the camera (i.e, the surface closest to the camera), the lights in the scene, and the position of the camera. If the surface closest to the camera is transparent, the color of the pixel is computed using the surface characteristics of the next closest surface as well. With the scan-line algorithm, light is never refracted.

2.2. Ray-tracing:

The ray-tracing algorithm traces the origins of the imaginary light ray that arrives at the camera through each pixel in the image plane. To achieve this, a ray is cast back into the object space to determine whether the ray was reflected or absorbed by a surface, was refracted by a transmissive medium, or originated directly from a light source. The path of a ray may be divided into two when part of light is reflected by a surface while another part travels through the surface. Ray-tracing is time-consuming. Use a scan-line algorithm-based renderer as much as you can and use ray-tracing only when it is absolute 'must.' For instance, you can use environmental mapping, instead of ray-tracing when you want an object to reflect the surrounding environment.

The following images show the difference between the scan-line algorithm and ray tracing algorithm. The 'interGlass' material in Maya's Shader Library was applied to a sphere, and the two images were rendered by switching renderer, without changing any attributes of the material or the light.

Scan-line Raytracing

2.3. Tessellation:

Tessellation is a process of converting non-polygonal objects into meshes, which are nets of interconnected triangles, prior to rendering. The finer the meshes are the more accurate appearances of the rendered objects are but the slower the rendering is.

2.4.Radiosity

Despite its ability to create highly impressive images, ray tracing has been criticized for its slowness and its emphasis on direct reflection and transmission. Looking for a technique that would more accurately render environments characterized more by diffuse reflection, Don Greenberg and his collaborators at Cornell devised the radiosity method of image synthesis in the mid-1980s.

The basic principles of radiosity are as follows:

- the system looks solely at the light/energy balance in a closed environment
- a closed environment is one in which all energy emitted or reflected from a given surface is accounted for by reflection and/or absorption by other surfaces

- it is possible to define a surface radiosity value: the rate at which energy leaves the surface:

radiosity = sigma (emissions + reflections + transmissions)

3. Volume rendering via 3D textures

When 3D textures became available on graphics workstations their potential benefit in volume rendering applications was soon recognized. The basic idea is to interpret the voxel array as a 3D texture defined over 0 1 3 and to understand 3D texture mapping as the trilinear interpolation of the volume data set at an arbitrary point within this domain.

At the core of the algorithm, multiple planes parallel to the image plane are clipped against the parametric texture domain (see Figure 1) and sent to the geometry processing unit. The hardware is then exploited for interpolating 3D texture coordinates issued at the polygon vertices and for reconstructing the texture samples by tri linearly interpolating within the volume. Finally, pixel values are blended appropriately into the frame buffer in order to approximate the continuous volume rendering integral.

3.1 Gradientless shading:

Now we turn our attention to tetrahedral grids most familiar in CFD simulation, which have also recently shown their importance in adaptive refinement strategies. Since most grid types that provide the data at unevenly spaced sample points can be quite easily converted into this representation, our technique, in general, is potentially attractive to a wide area of different applications.

Caused by the irregular topology of the grids to be processed the intrinsic problem showing up in direct volume rendering is to find the correct visibility ordering of the involved primitives. Different ways have been proposed to attack this problem, e.g. by improving sorting algorithms [23, 28, 5], by using space partitioning strategies [27], by taking advantage of hardware-assisted polygon rendering [20, 23, 31, 25] and by exploiting the coherence within

cutting planes in object space [6, 21]. Similar to the marching cubes algorithm, the marching tetrahedra approach suffers from the same limitations, i.e., the large amount of generated triangles.

In each tetrahedron (hereafter termed the volume primitive or cell) we have a linear range in the material distribution and therefore a constant gradient. The affine interpolation function f x y z

a bx cy dz which defines the material distribution within one cell is computed by solving the system of equations

for the unknowns a b c locations xi yi zi.

and d. fi is the function values given at

Now the partial derivatives, bc and d, provide the gradient components of each cell. Gradients

at the vertices are computed by simply averaging all contributions from different cells. These are stored in addition to the vertex coordinates and the scalar material values, the latter ones given as one-component color indices into an RGB? lookup table.

Conclusion

In this research, I have presented many different ideas to exploit basic/advanced features offered by modern high-end graphics workstations through standard APIs like OpenGL in volume rendering applications.

In particular, real-time rendering of shaded iso-surfaces has been made possible for Cartesian and tetrahedral grids avoiding any polygonal representation.

References

1. K. Akeley. Reality Engine Graphics. ACM Computer Graphics, Proc. SIG- GRAPH ’93, pages 109–116, July 1993.
2. T.J. Cullip and U. Neumann. Accelerating Volume Reconstruction with 3D Texture Hardware. Technical Report TR 93-027, University of North Carolina, Chapel Hill N.C., 1993.
3. Alias: Is it Fake or Foto?, http://www.alias.com/eng/etc/fakeorfoto/, Feb. 2006.