Rendering Primitives for a Virtual Holodeck

Chantelle Morkel **Prof. Shaun Bangay**
g99m0761@campus.ru.ac.za **s.bangay@ru.ac.za**

Department of Computer Science
Rhodes University
Grahamstown, South Africa

Abstract

The main objective of this research is to implement a "Star Trek"-like holodeck in a computer environment.
An experiment to create graphical primitives and images solely out of spheres is being conducted. We investigate several approaches of creating primitives using spheres, and then using these primitives to create images. Initial results of this experiment are presented and we conclude that using spheres to create primitives and images is a viable approach to creating realistic-looking three-dimensional (3D) images.

1. Introduction

A Star Trek style holodeck allows the rendering of 3D images. While the holographic hardware does not currently exist, we can explore implications for rendering algorithms by applying them to a simulated, hypothetical technology.

We assume the ability to holographically render spheres of a desired size and colour at any point within the holodeck.

To create any graphical image, it is necessary to have a variety of primitives to create the image with. Each primitive is made up solely of spheres, as opposed to the more conventional two-dimensional (2D) method of using pixels.

2. Related Work

In his paper, "The Holodeck: A Parallel Ray-caching Rendering System", Gregory Larson outlines his approach to creating a "data structure that resembles a Star Trek holodeck in form and function" [1].

Larson makes use of a grid on a "holodeck section" ("a combination of the notion of a hologram with an unobstructed region of free movement" [1]) which acts as a four-dimensional (4D) rendering target for a ray-tracing algorithm. To optimize disk and memory usage, Larson makes use of a "holodeck server", which coordinates both ray evaluation and display processing.

How it works:

a) One or more ray trace programs are used for ray tracing.
b) The holodeck server gathers and writes the ray information to a holodeck file.
c) The display driver sends requests to the holodeck server for beams that it needs to fill an image.
d) The holodeck server retrieves all the rays that are in memory, on disk and any new rays created by the ray tracing.

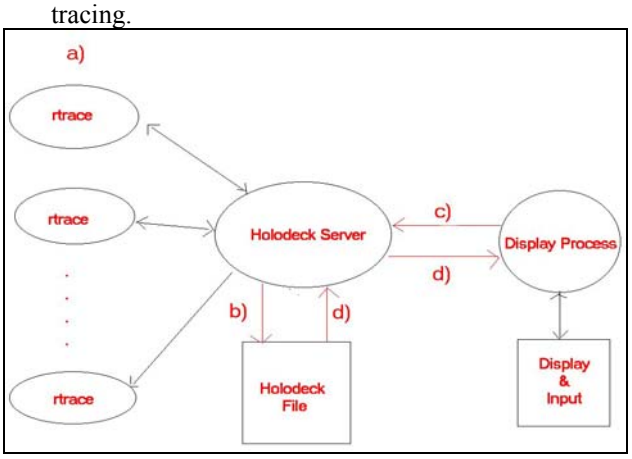

Figure 1. Adapted from [1]. This figure shows Larson's approach to his Holodeck

In their paper, "Toward the Holodeck: Integrating Graphics, Sound, Character and Story", the MRE (Mission Rehearsal Exercise) project group outlines their approach to simulating a holodeck. Having been inspired by Star Trek's Holodeck, the MRE was formed with the goal of "creating a virtual reality training environment in which different scenarios may be played out" [2]. The main use of the MRE is for military training exercises.

The simulation is hosted in a theatre or cinema type setting. The virtual characters and objects are created on computers and then projected onto a curved screen using three projectors. A sound controller triggers any sound events that are meant to coincide with what is happening in the simulation.

The trainee stands in front of the screen, which provides a 150^{o} field of vision. "People participating in the training are immersed in the sights and sounds of the setting and interact with virtual humans acting as characters in the scenario" [2].

It is also necessary to explore current holographic projection technologies. The Laser Magic group has introduced several ways of creating and projecting

holographic images. Two main ones are the 'TransScreen' and the 'HoloTank'.

"The TransScreen is comprised of a microscopic pattern of particles suspended in a dense medium that simultaneously diffracts, reflects and transmits all wavelengths of light, thus allowing an observer to both see the image and see through the screen" [3].

"The HoloTank is a 3D volumetric display, created by projecting scanned laser light or vector video graphics into a special liquid medium resulting in 3D holographic type projections. Moving laser or video images are projected into a liquid chamber (like an aquarium) in which proprietary microscopic particles are suspended" [4].

Actuality has also developed a new projection technology, that of the 'globe-shaped display'. "An image coming from a workstation is manipulated and dissected into 198 "slices". When combined, these slices make up a spherical image. The image is then projected onto a paper-thin, transparent screen, and, as the image is projected, the screen rotates at 730-revolutions per minute to create the illusion of a free-floating image" [5].

3. The Experiment

3.1 System Architecture

To implement a holodeck, it is necessary to devise a holographic technology. Our chosen technology creates images solely out of spheres. Effectively, each image is broken down into a set of primitives, which are then further broken down into a series of spheres.

Each sphere has several attributes:

- A set of coordinates (x, y, z)
- Radius (size)
- Colour (r, g, b)
- A quality value

For future enhancements:

- *A transparency value*
- *An emission value, and*
- *A reflectivity value*

To allow the user to control the quality of the image produced, we introduce the concept of the 'quality value'. The quality value of the spheres serves as a scale:

- Between 0 and 1: The spheres do not touch, which results in a rough framework of the image.
- Exactly 1: The spheres just touch, this gives a proper framework of the image.
- Greater than 1: The spheres overlap, resulting in a fine-grained framework of the image.

3.2 Image Processing

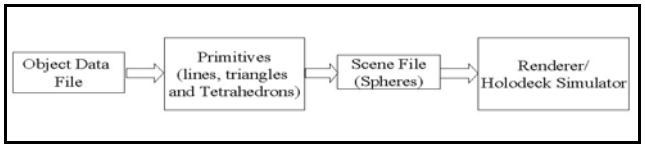

Figure 2. The process an image goes through before being displayed in the holodeck

The choice of primitive to use is dependent on the nature of the object data file (ODF). If the ODF originated from an OFF file [6] then the primitive that is most likely to be used is the triangle. Whereas, if the ODF originated from a test file, or user input, the choice of primitive is dependent on the user.

4. Implementation

As illustrated in Figure 2 above, each image is constructed using primitives that are then broken down into a series of spheres. The most common primitives are lines, triangles and tetrahedrons. Two less commonly used primitives, namely pyramids and polygons, also form part of the implementation.

Almost all the primitives implemented have three formats, namely framework, hollow with surrounding framework and solid. The exceptions are lines, triangles and 2D polygons. The solid primitive is used to create a solid-looking image, which is true to the Star Trek holodeck. The hollow image is used when faster rendering is required, it still gives the impression of a solid image, but is not. Finally, the framework is used when images are needed quickly.

4.1 Lines

The task of drawing a line made entirely out of spheres is achieved by using the following algorithm:

$dx = (point1\_x + point2\_x)/2$
$dy = (point1\_y + point2\_y)/2$
$dx = (point1\_z + point2\_z)/2$
$distance = SquareRoot(dx^2 + dy^2 + dz^2)$
$number\_of\_spheres = quality*(distance/radius)$

For 0 to number_of_spheres
Write sphere attributes to file
x +=dx/number_of_spheres
y +=dy/number_of_spheres
z +=dz/number_of_spheres
End For

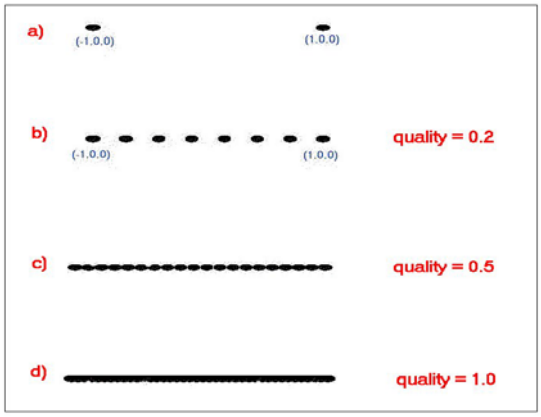

Figure 3. A line made up of spheres, drawn at different levels of quality

4.2 Triangles

4.2.1 Frame Triangles

To construct a triangle framework, it is easiest to make use of the line primitive. All that is required is to draw three lines, which results in a triangle.

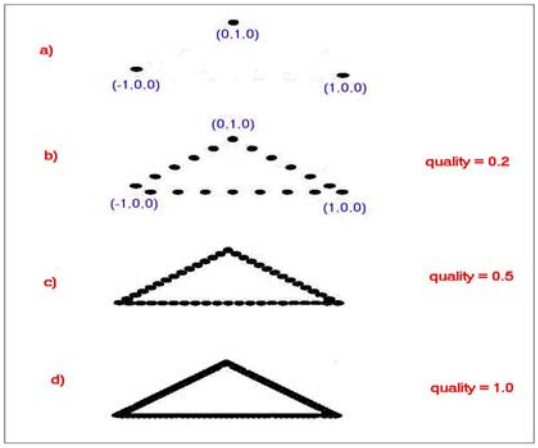

Figure 4. A triangle made up of spheres, drawn at different levels of quality

4.2.2 Filled Triangles

We examine several approaches to creating a filled triangle. The first approach draws a frame triangle, and then draws lines from the apex of the triangle to each of the sphere coordinates on the base line. At high levels of quality this approach works fairly well, but as the quality is reduced it becomes obvious that this approach is unsuitable. The reason being that the area around the apex has a heavy concentration of spheres, while the effect tapers out towards the bottom of the triangle.

The second approach focuses rather on calculating the areas of both the triangle and the sphere and recursively dividing under the first, until it is comparable with the second.

```
void drawTriangle(point1, point2, point3)
Calculate triangle_area
Calculate sphere_area

If (triangle_area > sphere_area)
    Divide triangle into four smaller ones
    Calculate new corner coordinates by taking the average
     of the x, y and z coordinates
    drawTriangle(point1, newpoint1, newpoint2)
    drawTriangle(newpoint1, point2, newpoint3)
    drawTriangle(newpoint1, newpoint2, newpoint3)
    drawTriangle(newpoint2, newpoint3, point3)
else
    Write sphere attributes to file
End If
End drawTriangle
```

This algorithm, however, doesn't allow for variation in quality levels. To overcome this problem, we change the way the algorithm works. Instead of recursing until the triangle area is less than or equal to the sphere area, the number of recursions is required beforehand. This approach allows for the level of quality to be manipulated.

```
void drawTriangle(point1, point2, point3, recursions)
Calculate triangle_area
Calculate sphere_area
recursions = quality * log4(triangle_area/sphere_area)

While (recursions > 0)
     recursions--
    Divide triangle into four smaller ones
    Calculate new corner coordinates by taking the average
     of the x, y and z coordinates
    drawTriangle(point1, newpoint1, newpoint2, recursions)
    drawTriangle(newpoint1, point2, newpoint3, recursions)
    drawTriangle(newpoint1, newpoint2, newpoint3,
                 recursions)
    drawTriangle(newpoint2, newpoint3, point3, recursions)
End While
    Write sphere attributes to file
End drawTriangle
```

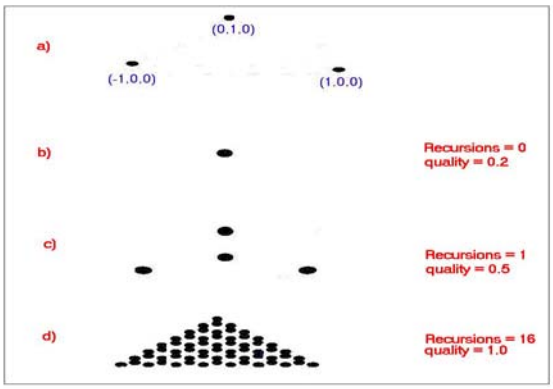

Figure 5. A filled triangle, drawn at different levels of quality. The number of recursions is shown on the side.

4.3 Tetrahedrons

4.3.1 Frame Tetrahedrons

The framework for a tetrahedron may be created using four frame triangles as shown in Figure 6.

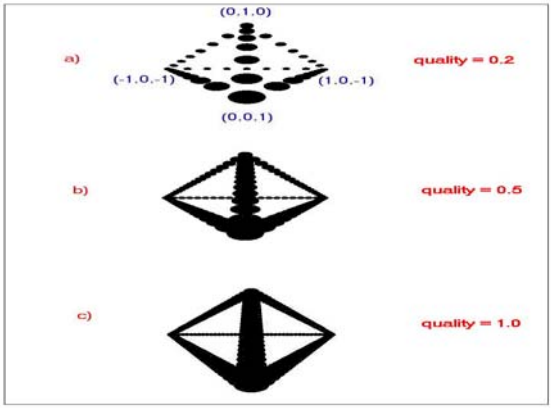

Figure 6. A tetrahedron made out of spheres, drawn at different levels of quality.

4.3.2 Hollow Tetrahedrons

Similar to the frame tetrahedron, the hollow tetrahedron is made up of four filled triangles.

4.3.3 Solid Tetrahedrons

Considering that a tetrahedron is a 3D primitive, it is difficult to find an effective way of creating a solid tetrahedron. The reason being that it is necessary to work with the volume of both the tetrahedron and the sphere that is required to fill it.

The solid tetrahedron makes use of the Pyramid primitive. A pyramid is made of two tetrahedrons, so the drawPyramid method effectively divides the pyramid into two tetrahedrons by calling drawTetrahedron twice.

```
void drawSolidTet (point1, point2, point3, point4)
Calculate tetrahedron_volume
sphere_radius = cube root (tetrahedron_volume/2) / Pi)

If (sphere_radius > default radius)
        Calculate mid-point of each line
        Write sphere attributes to file

        drawPyramid(midpt3_pt4, midpt1_pt4, midpt2_pt4,
                    midpt2_pt3, midpt1_pt3)
        drawPyramid(midpt1_pt2, midpt1_pt4, midpt2_pt4,
                    midpt2_pt3, midpt1_pt3)

        drawSolidTet(point1, midpt1_pt2, midpt1_pt3,
                    midpt1_pt4)
        drawSolidTet(point2, midpt1_pt2, midpt2_pt3,
                    midpt2_pt4)
        drawSolidTet(point3, midpt1_pt3, midpt2_pt3,
                    midpt3_pt4)
        drawSolidTet(point4, midpt1_pt4, midpt2_pt4,
                    midpt3_pt4)
End If
End drawSolidTet
```

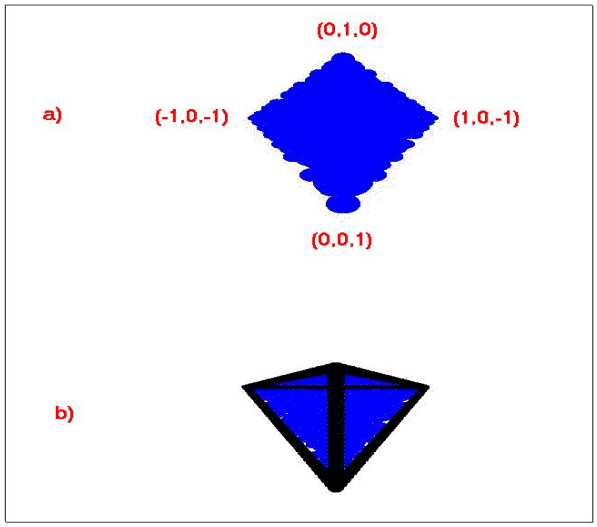

Figure 7. a) Filled tetrahedron, without framework, b) Same tetrahedron with framework

4.4 Other Primitives

The tetrahedron is in fact the basic component for both the pyramid and 3D polygon. The pyramid and polygon are created by calling the drawTetrahedron twice and four times respectively.

5. Results

Figures 3, 4, 5, 6 and 7 illustrate that our suggested method of creating primitives out of spheres does in fact work. Which means that there is a viable alternative to the conventional method of creating primitives out of pixels.

Considering that images in a holodeck are necessarily 3D, it is useful to have primitives that are already 3D due to the nature of spheres.

In Section 3.2, mention is made of the ability to use primitives to convert OFF files into 3D images made of spheres. Most OFF files contain the coordinates of images that are made out of triangulated faces. This allows for easy conversion to triangulated faces that are made solely out spheres. Figure 8 illustrates an image that originated from an OFF file. More images can be seen in Figures 9 and 10.

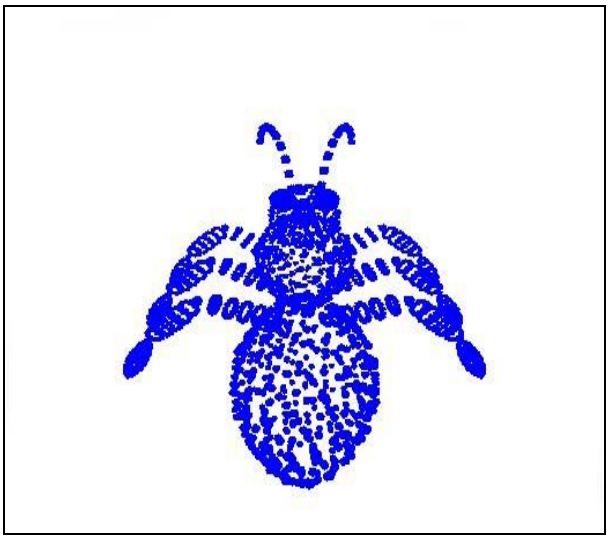

Figure 8. An ant, created from an OFF file (using frame triangles), quality = 0.5

Figure 8 serves as conclusive proof that it is indeed possible to incorporate individual sphere-based primitives into a 3D image.

Another aspect, in which we can see interesting results, is that of filling the primitives. Most of our approaches work very effectively in filling the primitives, which in turn results in solid-looking images that are in likeness of the original object.

We conclude that our experiment of creating primitives and images out of spheres has been an overall success. The main objective of creating a "Star Trek"-like holodeck is in the final stages of completion, and once all outstanding work has been completed, will be finished.

Figure 9. A helicopter, created from an OFF file (using frame triangles), quality = 1.0

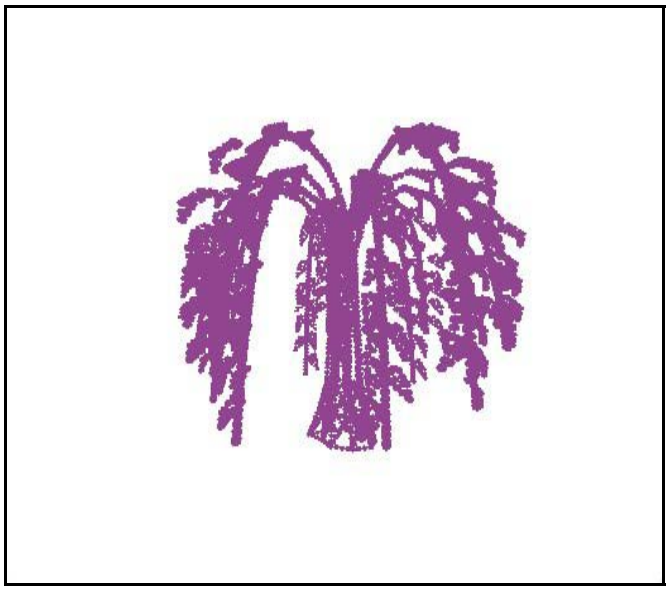

Figure 10. A palm tree, created from an OFF file (using frame triangles), quality = 1.0

6. Outstanding Work

The actual holodeck is completed. Current work includes shading, and is aimed at improving the overall 'look' of the images within the holodeck.

7. References

[1] Larson, G. W. *The Holodeck: A Parallel Ray-caching Rendering System*, Silicon Graphics

[2] Swartout, W., Hill, R., Gratch, J., Johnson, W. L., Kyriakakis, C., LaBore, C., Lindheim, R., Marsella, S., Miraglia, D., Moore, B., Morie, J., Rickel, J., Thi Ú baux, M., Tuch, L., Whitney, R., and Douglas J. *Toward the Holodeck: Integrating Graphics, Sound, Character and Story*, In Proceedings of Fifth International Conference on Autonomous Agents, pp. 409-416, May 2001. New York: ACM Press.

[3] Laser Magic Productions, *The TransScreen,* Available at: http://www.laser-magic.com/transscreen.html

[4] Laser Magic Productions, *The HoloTank,* Available at: http://www.laser-magic.com/holotank.html

[5] Berger, M., *Actuality's globe-shaped display* http://cssvc.pcworld.compuserve.com/computing/cis/article/0,aid,102372,00.asp

[6] Rost, A. J., *OFF - A 3D Object File Format*, Availabe at: http://www.dcs.ed.ac.uk/home/mxr/gfx/3d/OFF.spec