Discussion:

Module-6 Critical Thinking – Sphere Object

Alejandro Ricciardi

Colorado State University Global

CSC405: Graphics and Visualization

Professor: Dr Jennifer Marquez

September 22, 2024

Discussion:

Module-6 Critical Thinking – Sphere Object

In computer graphics, particularly in OpenGL or WebGL, spheres are typically rendered using triangles since spheres are not supported within the APIs. Primitive shapes such as triangles are used because drawing every point on a sphere is impossible, so primitive shapes are used to approximate a sphere shape. Different techniques are used to achieve this approximation. One of them is to divide the sphere by latitude and longitude sectors connected by triangle primitives forming a mesh of triangles (Ahn, n.d.). This is computed using parametric equations. Another technique is using icosahedrons. Icosahedrons are regular polyhedrons with 12 vertices and 20 equilateral triangles. The Icosahedrons are recursively dived resulting in a smoother surface. This smoother surface helps with lighting and shading computations, particularly when dealing with reflections and vector computations because the finer the subdivision is the more accurate normal are, making the surface light reflections appear more realistic. Another method that approximates a sphere shape by using a recursive subdivision is to start with a tetrahedron and subdivide each facet into smaller triangles recursively (resulting in a smaller tetrahedron) creating an approximation of a sphere (Angel & Shreiner, 2020). Below is a code example in JavaScript that implements a recursive subdivision of a tetrahedron to approximate a sphere:

Code Snippet from the ‘tetrahedron’ and the ‘triangle’ functions

```javascript
// Function to create a tetrahedron by subdividing its four triangular faces
function tetrahedron(a, b, c, d, n) {
    // Subdivide and draw the first triangular face
    divideTriangle(a, b, c, n);

    // Subdivide and draw the second triangular face
    divideTriangle(d, c, b, n);

    // Subdivide and draw the third triangular face
    divideTriangle(a, d, b, n);
```

```
  // Subdivide and draw the fourth triangular face
  divideTriangle(a, c, d, n);
}

// Recursive function to subdivide a triangle into smaller triangles
// 'a', 'b', 'c' are vertices of the triangle, and 'count' is the recursion depth
function divideTriangle(a, b, c, count) {
  //---- Base case: if count reaches zero, stop recursion and draw the triangle ----
  if (count > 0) {
    // Find the midpoints of each edge of the triangle
    var ab = normalize(mix(a, b, 0.5), true);  // Midpoint of edge AB
    var ac = normalize(mix(a, c, 0.5), true);  // Midpoint of edge AC
    var bc = normalize(mix(b, c, 0.5), true);  // Midpoint of edge BC

    //--- Recursive call
    // Recursively subdivide the triangle into four smaller triangles
    divideTriangle(a, ab, ac, count - 1);  // Subdivide triangle A-AB-AC
    divideTriangle(ab, b, bc, count - 1);  // Subdivide triangle AB-B-BC
    divideTriangle(bc, c, ac, count - 1);  // Subdivide triangle BC-C-AC
    divideTriangle(ab, bc, ac, count - 1);  // Subdivide triangle AB-BC-AC
  } else {
    // If count is zero, draw the triangle as-is (base case of recursion)
    triangle(a, b, c);
  }
}

// Function to add the triangle's vertices to the position array for rendering
function triangle(a, b, c) {
  // Push the vertices of the triangle into the positions array
  positions.push(a);
  positions.push(b);
  positions.push(c);
}
```

Even though different methods of recursive subdivision are available, all of them result in smooth surfaces that are beneficial for lighting and shading computations, particularly when dealing with reflections and vector computations resulting in accurate normal making the surface light reflections appear more realistic. Additionally, they give more control over the level of detail by adjusting the recursion depth.

References

Ahn, S. (n.d.). OpenGL sphere. *OpenGL*. Songho. https://www.songho.ca/opengl/gl_sphere.html#icosphere

Angel, E., & Shreiner, D. (2020). Chapter 6: Light and shading. *Interactive computer graphics. 8th edition*. Pearson Education, Inc. ISBN: 9780135258262