Assignment#3 Q3 ReadMe

Name: Wen Sheng
E-Mail: wen.sheng@yale.edu

Operating System: Mac OS 10.10.5
Browser: Firefox 40.0.3 / Chrome 45.0 / Safari 8.0.8

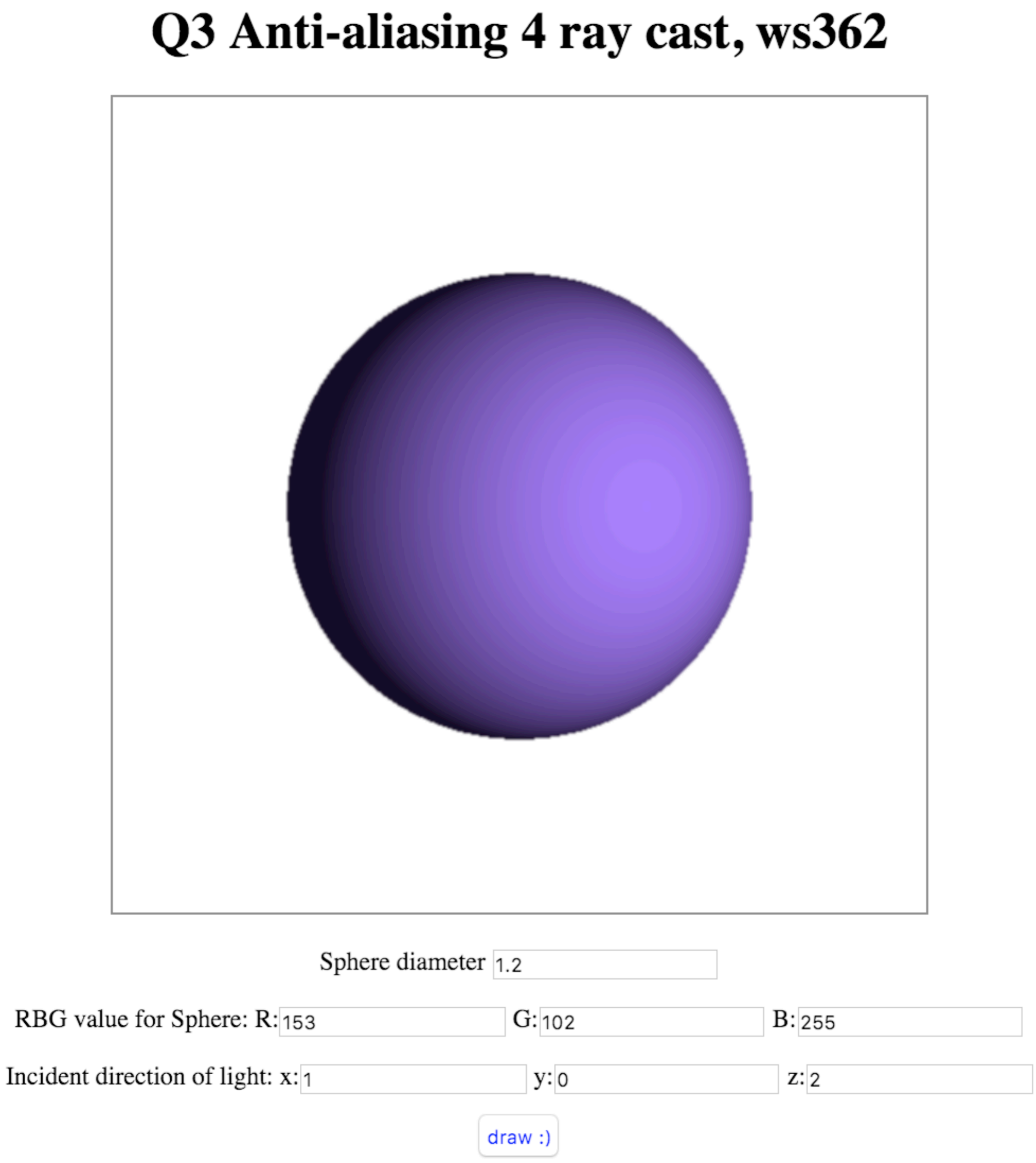
Q3 478/578 Anti-aliasing

I. Input

- Sphere Diameter:
- Light Direction
- Sphere Color

II. Output (The color of the sphere looks softer and smoother than Q2 images)

➢ Sample 1: => Diameter: 1.2, Color: (153, 102, 255), light direction:(1, 3, 3)



Q3 Anti-aliasing 4 ray cast, ws362

Sphere diameter 1.2

RBG value for Sphere: R: 153 G: 102 B: 255

Incident direction of light: x: 1 y: 0 z: 2

Q3 578 Anti-aliasing Adaptive

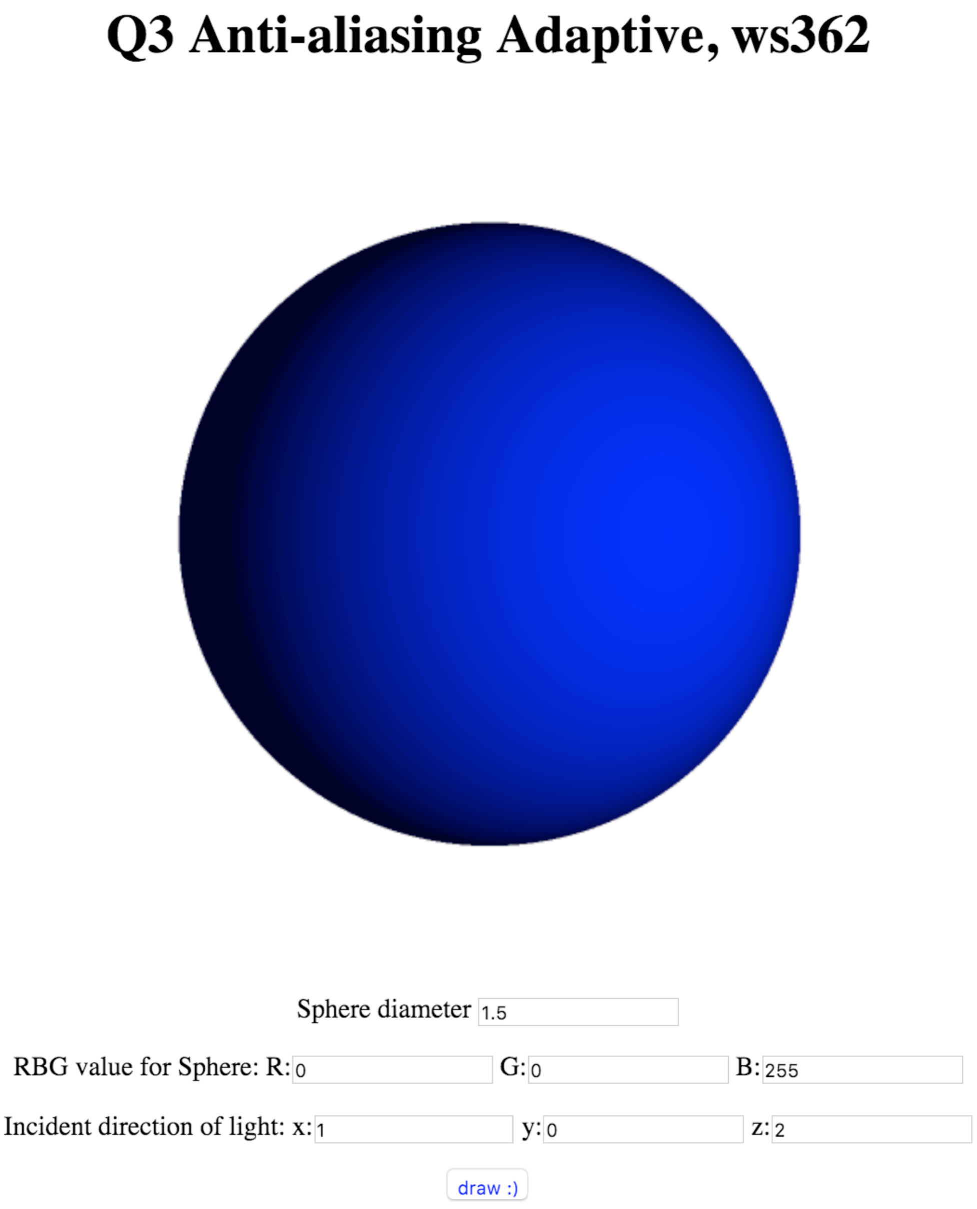
I. Input
   - Sphere Diameter:
   - Light Direction
   - Sphere Color

II. Output (The color of the sphere looks even softer and smoother than the Q3 non-adaptive one. And the edge of the sphere is less rough)

- ➢ Sample 1: => Diameter: 1.5, Color: (0, 0, 255), light direction:(1, 0, 2)

III. Algorithm:

Use the recursive method calColor() to continue check the 4 corners of each pixel. If they don’t equal, then calculate the unequal sub-square of the original pixel square by reducing its side length to half length. Keep checking the sub square until the four corners equal.



Q3 Anti-aliasing Adaptive, ws362

Sphere diameter 1.5

RBG value for Sphere: R: 0 G: 0 B: 255

Incident direction of light: x: 1 y: 0 z: 2

```
var calColor = function (depth, centerX, centerY, delta) {
  color = [];
```

```
var xcomp = Vector.scale(vpRight, ((centerX + delta) * pixelWidth) - halfWidth),
    ycomp = Vector.scale(vpUp, ((height - (centerY + delta)) * pixelHeight) - halfHeight);
ray.vector = Vector.unitVector(Vector.add3(eyeVector, xcomp, ycomp));
color.push(trace(ray, scene, 0));

xcomp = Vector.scale(vpRight, ((centerX + delta) * pixelWidth) - halfWidth);
ycomp = Vector.scale(vpUp, ((height - (centerY - delta)) * pixelHeight) - halfHeight);
ray.vector = Vector.unitVector(Vector.add3(eyeVector, xcomp, ycomp));
color.push(trace(ray, scene, 0));

xcomp = Vector.scale(vpRight, ((centerX - delta) * pixelWidth) - halfWidth);
ycomp = Vector.scale(vpUp, ((height - (centerY + delta)) * pixelHeight) - halfHeight);
ray.vector = Vector.unitVector(Vector.add3(eyeVector, xcomp, ycomp));
color.push(trace(ray, scene, 0));

xcomp = Vector.scale(vpRight, ((centerX - delta) * pixelWidth) - halfWidth);
ycomp = Vector.scale(vpUp, ((height - (centerY - delta)) * pixelHeight) - halfHeight);
ray.vector = Vector.unitVector(Vector.add3(eyeVector, xcomp, ycomp));
color.push(trace(ray, scene, 0));

if (checkColor(color[0], color[1]) && checkColor(color[2], color[1])
    && checkColor(color[2], color[3])) {
    //if (color[0] == color[1] && color[1] == color[2] && color[2] == color[3]) {
    return color[0];
} else if (depth == maxDepth) {
    var avg = {
        x: (color[0].x + color[1].x + color[2].x + color[3].x) / 4,
        y: (color[0].y + color[1].y + color[2].y + color[3].y) / 4,
        z: (color[0].z + color[1].z + color[2].z + color[3].z) / 4,
    }
    return avg;
} else {

    delta *= 0.5;
    color = [];
    color.push(calColor(depth + 1,
        centerX + delta,
        centerY + delta, delta));

    color.push(calColor(depth + 1,
        centerX + delta,
        centerY - delta), delta);

    color.push(calColor(depth + 1,
        centerX - delta,
        centerY + delta, delta));
    color.push(calColor(depth + 1,
        centerX - delta,
        centerY - delta, delta));
        var avg = {
            x: (color[0].x + color[1].x + color[2].x + color[3].x) / 4,
            y: (color[0].y + color[1].y + color[2].y + color[3].y) / 4,
            z: (color[0].z + color[1].z + color[2].z + color[3].z) / 4,
        }
        return avg;
    }
}
```