/\* Place an eye point at (0,0,0).

For each pixel (i, j) in 0...(nx-1), 0...(ny-1) make a ray in direction

(x,y,z) = (-1 + 2\*(i/(nx-1), -1 + 2\*(j/(ny-1)), -1). Use (nx, ny) = (256, 256).

Place a sphere with radius 2 at (0, 0, -3)

When a ray misses the sphere, set the pixel to color (red, green, blue) = (0, 0.2\*(1+y), 0.1),

where y is the y component of the ray direction.

When the ray hits the sphere, compute the surface normal (nx, ny, nz) and use the color

(red, green, blue) = ((nx + 1)/2, (ny+1)/2, (nz+1)/2).

\*/

import javax.imageio.ImageIO;

import java.io.File;

import java.io.IOException;

import java.awt.image.BufferedImage;

public class RayTracing {

/\*\* This method checks id a ray from the origin hits the sphere

\*

\* @param ray

\* @param center

\* @param radius

\* @return true if the ray hits the sphere

\*/

public static boolean doesHitSphere(vector ray, vector center, double radius) {

boolean answer = false;

double lengthRay = vector.vectorLength(ray);

// System.out.println("ray length " + lengthRay);

double lengthCenter = vector.vectorLength(center);

// System.out.println("center length " + lengthCenter);

double cosRayCenter = (ray.getX()\*center.getX()+ray.getY()\*center.getY()+ray.getZ()\*center.getZ())/(lengthRay\*lengthCenter);

double d = lengthCenter\*Math.sqrt(1-cosRayCenter\*cosRayCenter);

if (d<=radius) {

answer = true;

}

return answer;

}

/\*\* This method finds normal to the surface of sphere

\*

\* @param ray

\* @param center

\* @return

\*/

public static vector normal(vector ray, vector center, double radius) {

double lengthRay = vector.vectorLength(ray);

double lengthCenter = vector.vectorLength(center);

double cosRayCenter = (ray.getX()\*center.getX()+ray.getY()\*center.getY()+ray.getZ()\*center.getZ())/(lengthRay\*lengthCenter);

double d = lengthCenter\*Math.sqrt(1-cosRayCenter\*cosRayCenter);

double t = (Math.sqrt(lengthCenter\*lengthCenter - d\*d) - Math.sqrt(radius\*radius - d\*d))/lengthRay;

vector hit = new vector (t\*ray.getX(), t\*ray.getY(), t\*ray.getZ());

vector normal = new vector(hit.getX()-center.getX(), hit.getY()-center.getY(), hit.getZ()-center.getZ());

return normal;

}

public static void main(String[] args) {

BufferedImage image = new BufferedImage (256, 256, BufferedImage.TYPE\_INT\_RGB);

try {

for (int i = 0; i < 255; i++) {

for (int j = 0; j < 255; j++) {

double red, green, blue;

vector ray = new vector(2.0\*i/255.0 -1.0, 2.0\*j/255.0 - 1.0, -1.0);

// System.out.println("i = "+i+ "; j = " +j+". "+ ray.getX()+" "+ray.getY()+" "+ray.getZ());

vector center = new vector(0,0,-3);

if (doesHitSphere(ray, center, 2)) {

vector norm = normal(ray, center, 2);

// System.out.println("i = "+i+ "; j = " +j+". "+ norm.getX()+" "+norm.getY()+" "+norm.getZ());

if (norm.getX()>norm.getY() && norm.getX()>norm.getZ()) {

norm.setX(1);

norm.setY(norm.getY()/norm.getX());

norm.setZ(norm.getZ()/norm.getX());

}

else if (norm.getY()>norm.getZ() && norm.getY()>norm.getZ()) {

norm.setY(1);

norm.setX(norm.getX()/norm.getY());

norm.setZ(norm.getZ()/norm.getY());

}

else {

norm.setZ(1);

norm.setY(norm.getY()/norm.getZ());

norm.setX(norm.getX()/norm.getZ());

}

red = (norm.getX() + 1.0)\*255.0/2.0;

green = (norm.getY()+1.0)\*255.0/2.0;

blue = (norm.getZ()+1.0)\*255.0/2.0;

// System.out.println("Hit");

}

else {

red = 0;

green = 0.2\*(1+ray.getY());

blue = 0.1;

// System.out.println("Miss");

}

int r = (int)red;

int g = (int)green;

int b = (int)blue;

int rgb = ((r&0x0ff)<<16)|((g&0x0ff)<<8)|(b&0x0ff);

image.setRGB(j,i,rgb);

}

}

ImageIO.write(image, "jpg", new File("/Users/danaadylova/Desktop/sphereRayTracing.jpg"));

} catch (IOException e) {

System.out.println("Exception occured :" + e.getMessage());

}

// vector r = new vector(0,0,-1);

// vector c = new vector(0,0,-1);

// System.out.println(normal(r,c, 2).getX()+" "+ normal(r,c,2).getY()+" " + normal(r,c,2).getZ()+" ");

}

}