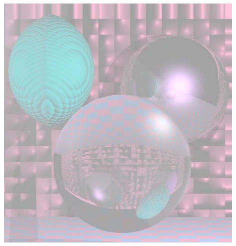
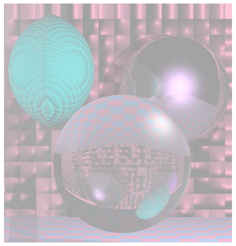


CS 4300 Computer Graphics

Prof. Harriet Fell
Fall 2012
Lecture 29 – November 14, 2012

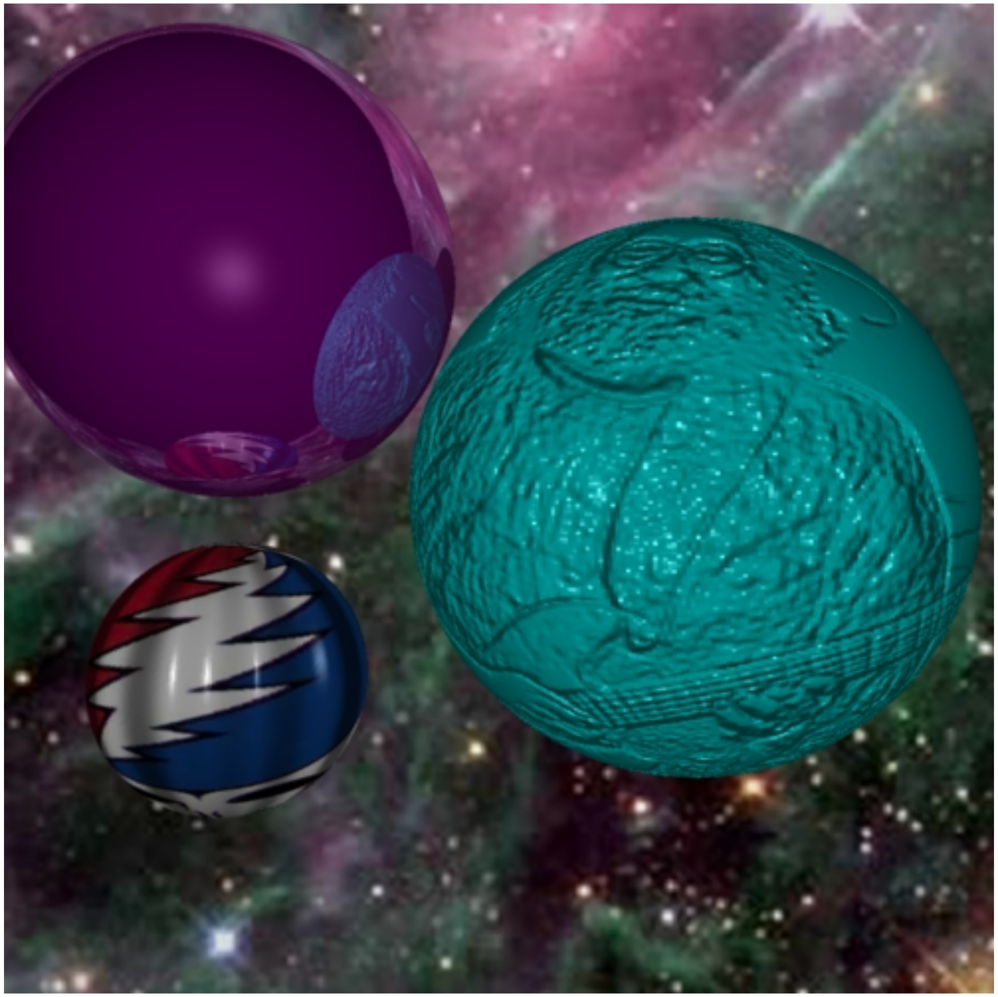


CS 4300
Computer Graphics

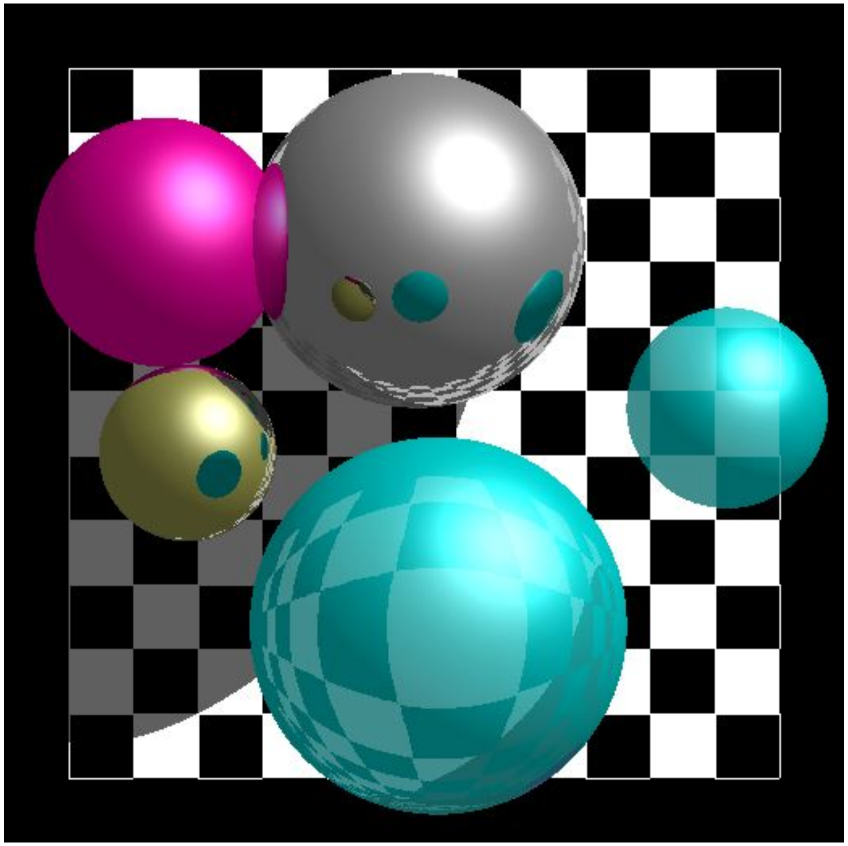
Prof. Harriet Fell
Fall 2012
Lecture 28 – November 8, 2012

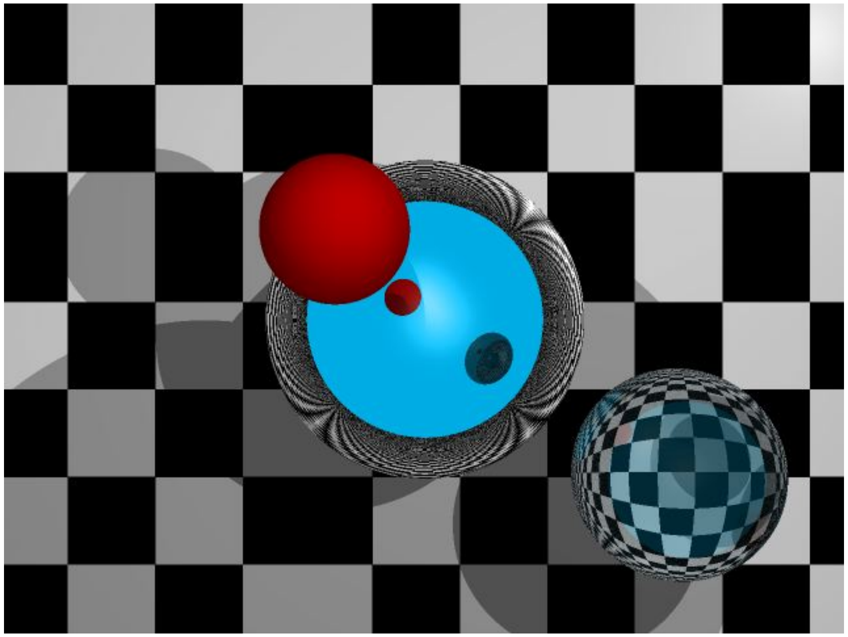
Bump Map from an Image

Victor Ortenberg

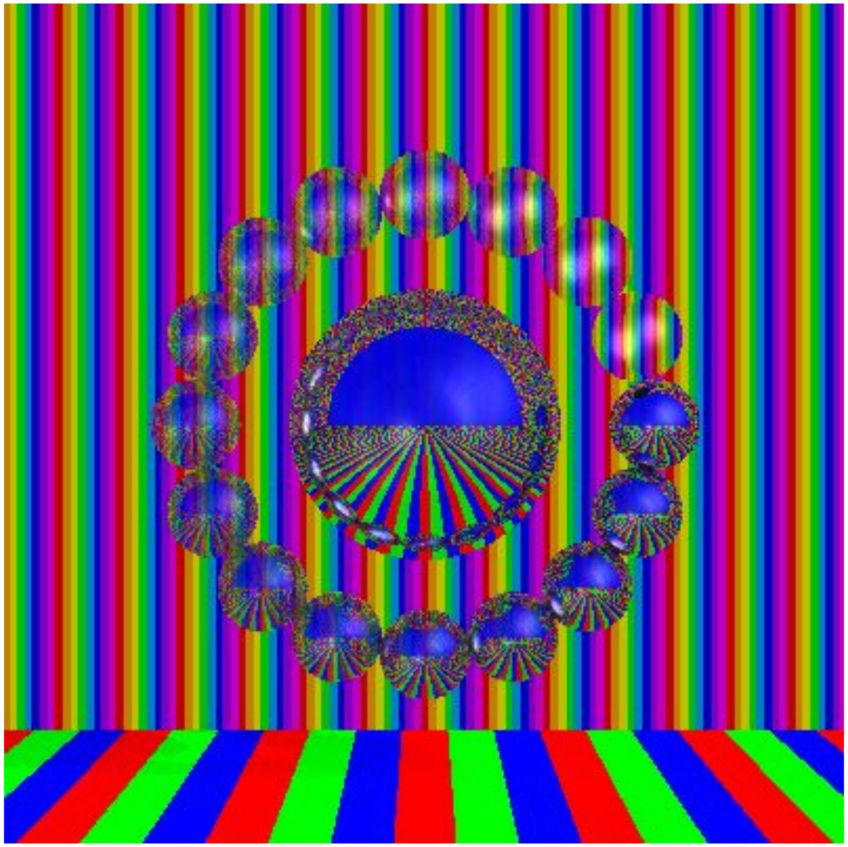


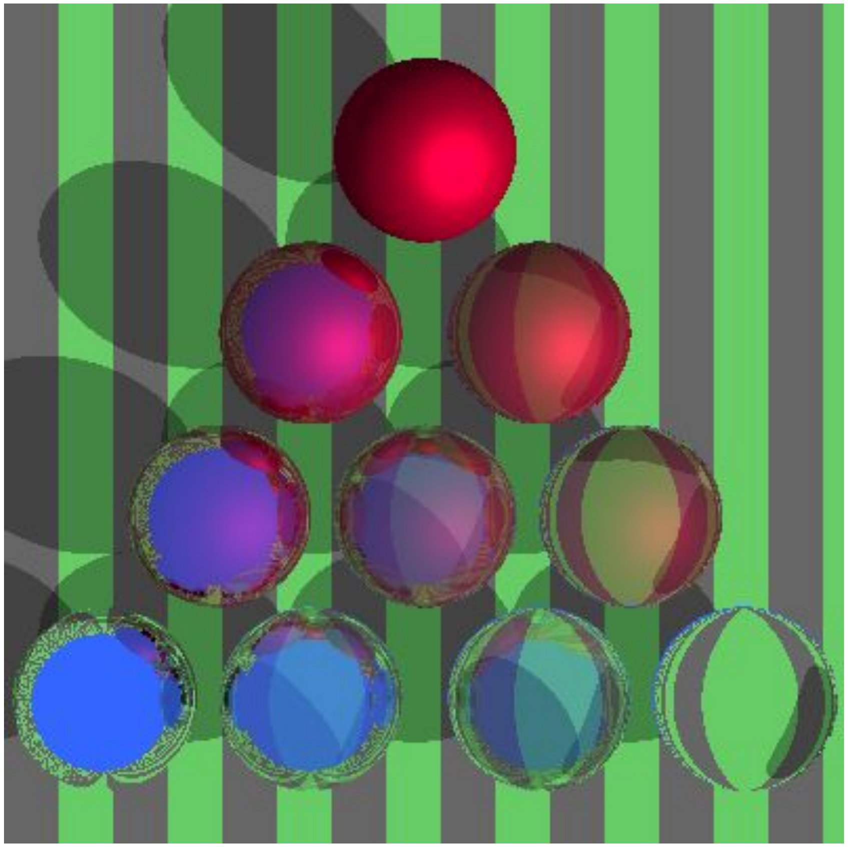
Simple Textures on Planes Parallel to Coordinate Planes



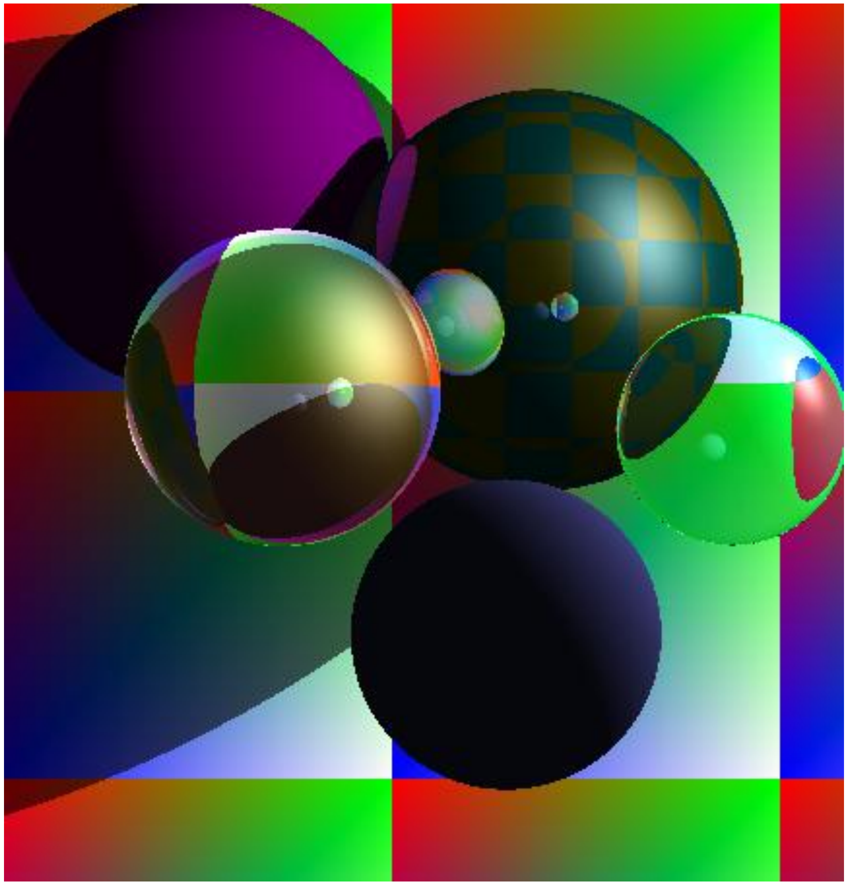


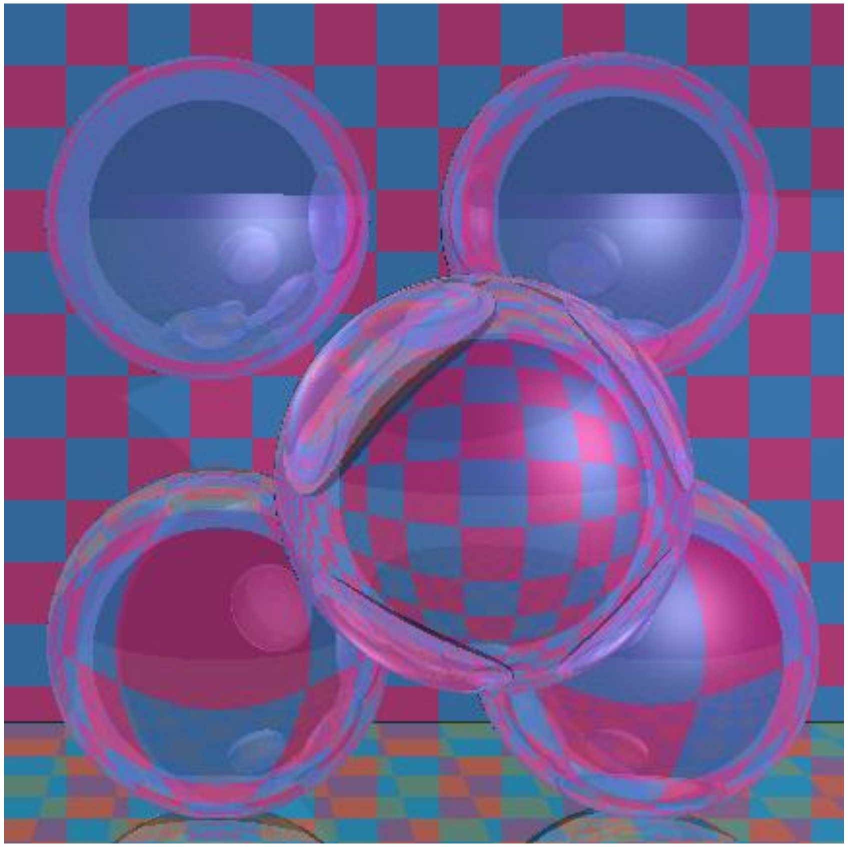
Stripes

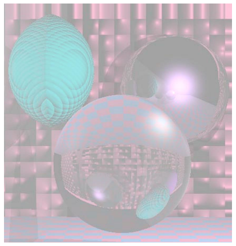




Checks



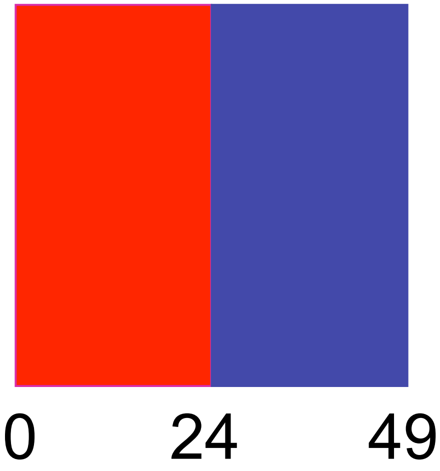




Stripes and Checks

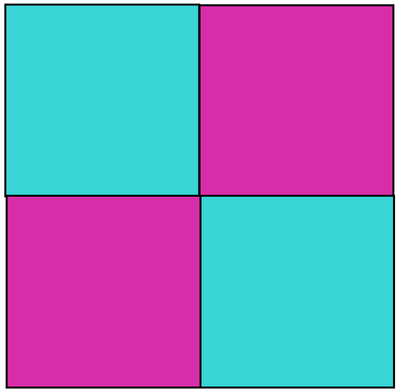
Red and Blue Stripes

```
if ((x % 50) < 25) color = red
else color = blue
```



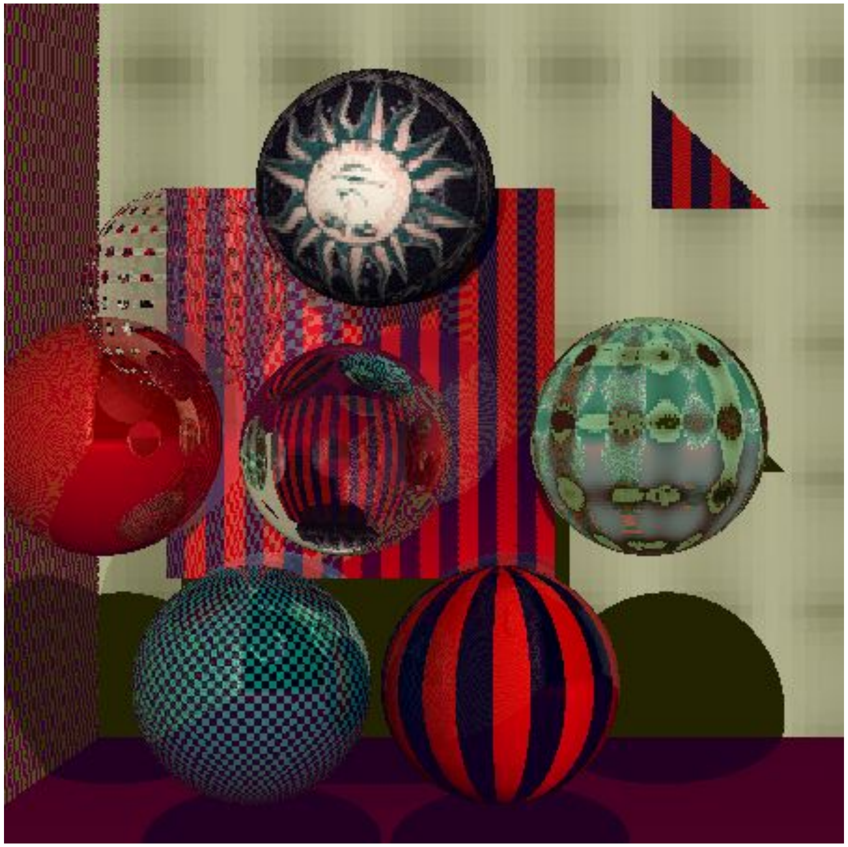
Cyan and Magenta Checks

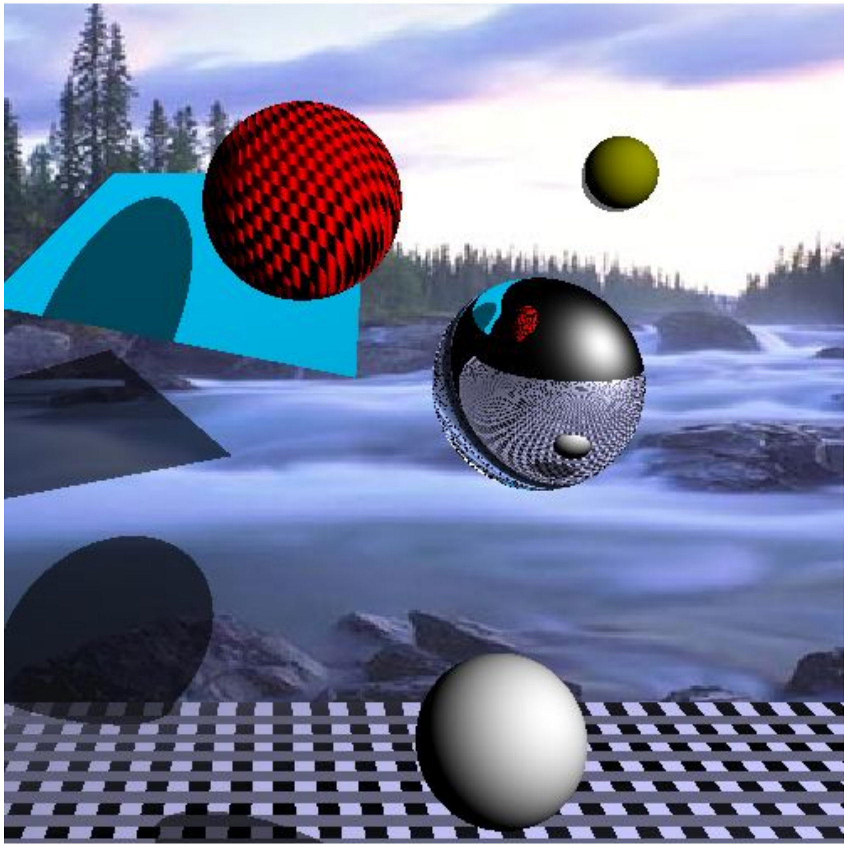
```
if (((x % 50) < 25 && (y % 50) < 25)) ||
    (((x % 50) >= 25 && (y % 50) >= 25)))
        color = cyan
    else color = magenta
```



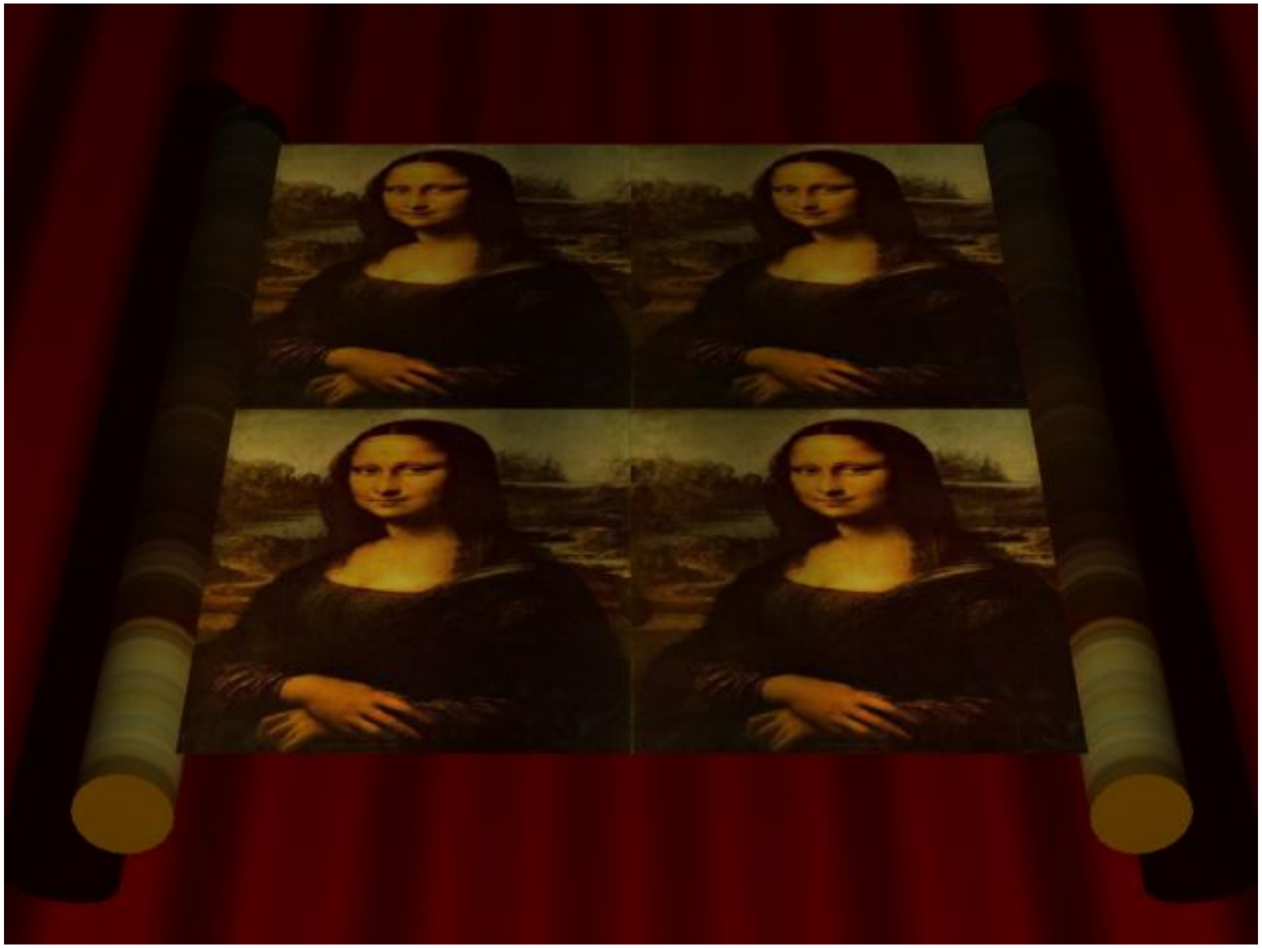
What happens when you cross x = 0 or y = 0?

Stripes, Checks, Image

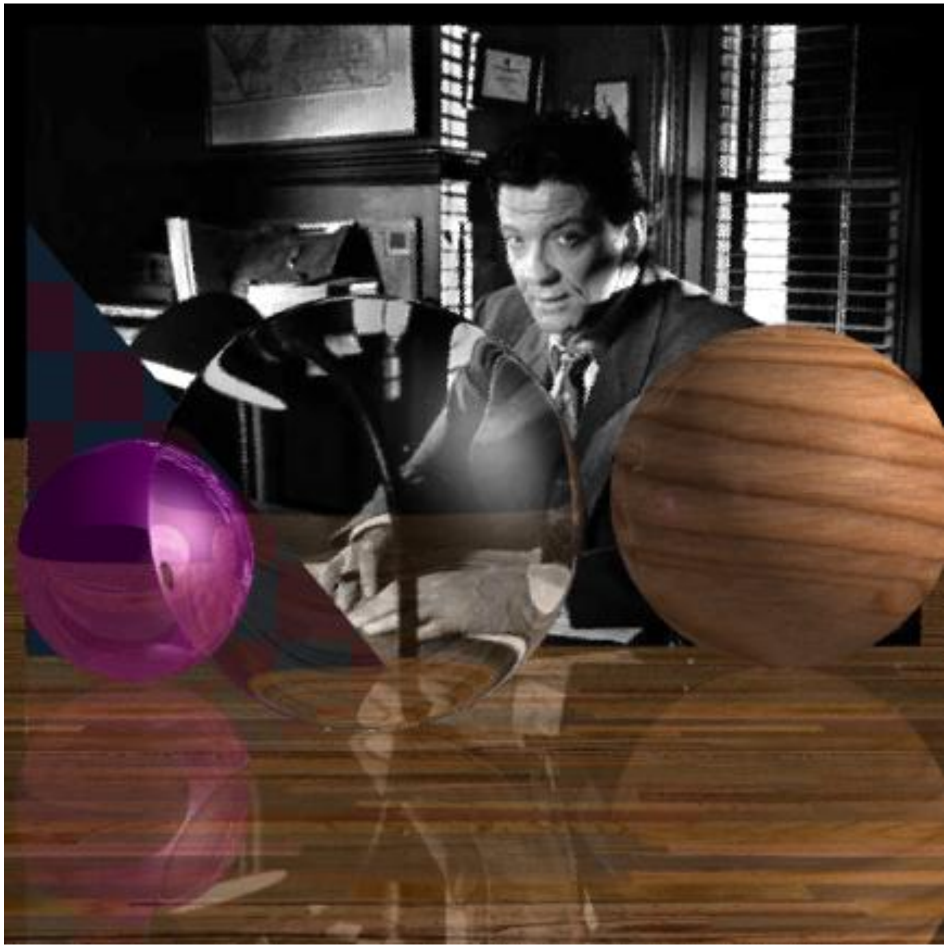




Mona Scroll



Textures on 2 Planes

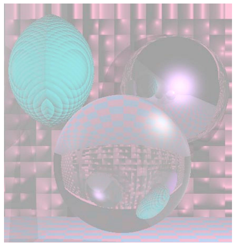


Mapping a Picture to a Plane

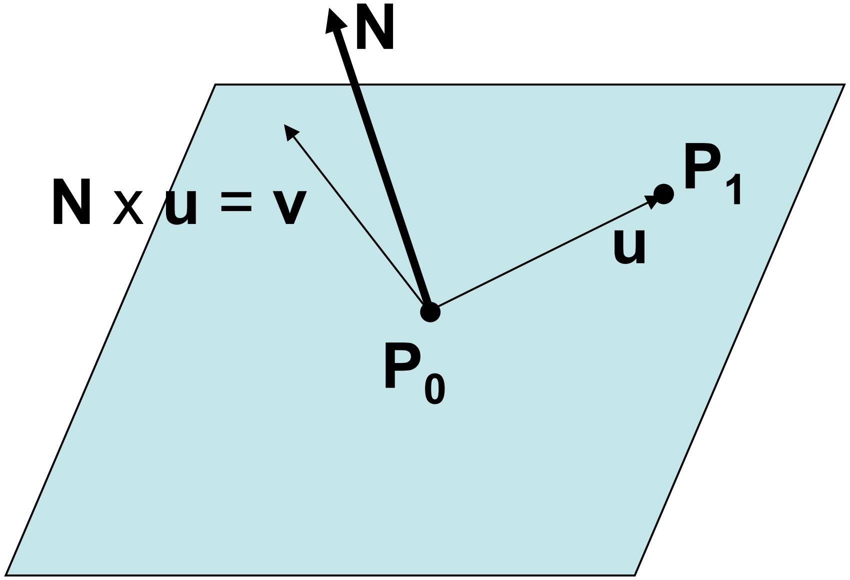
- Use an image in a ppm file.
- Read the image into an array of RGB values.

  Color myImage[width][height]
- For a point on the plane (x, y, d)

  theColor(x, y, d) = myImage(x % width, y % height)
- How do you stretch a small image onto a large planar area?



Other planes and Triangles



Given a normal and 2 points on the plane:

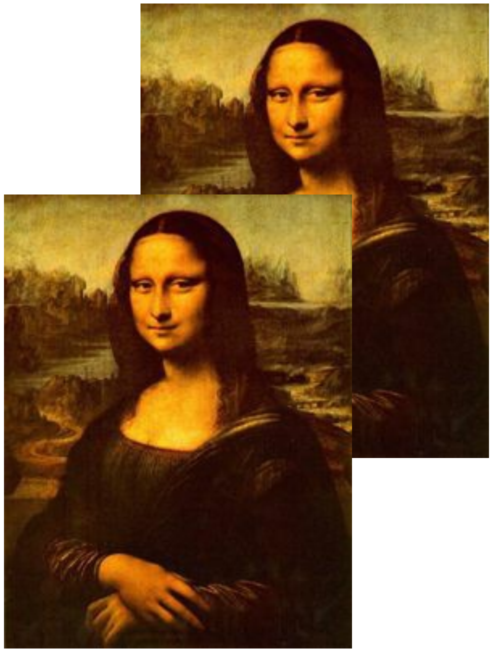
Make **u** from the two points.

$\mathbf{v} = \mathbf{N} \times \mathbf{u}$

Express **P** on the plane as

$\mathbf{P} = \mathbf{P}_0 + a\mathbf{u} + b\mathbf{v}$.

Image to Triangle - 1



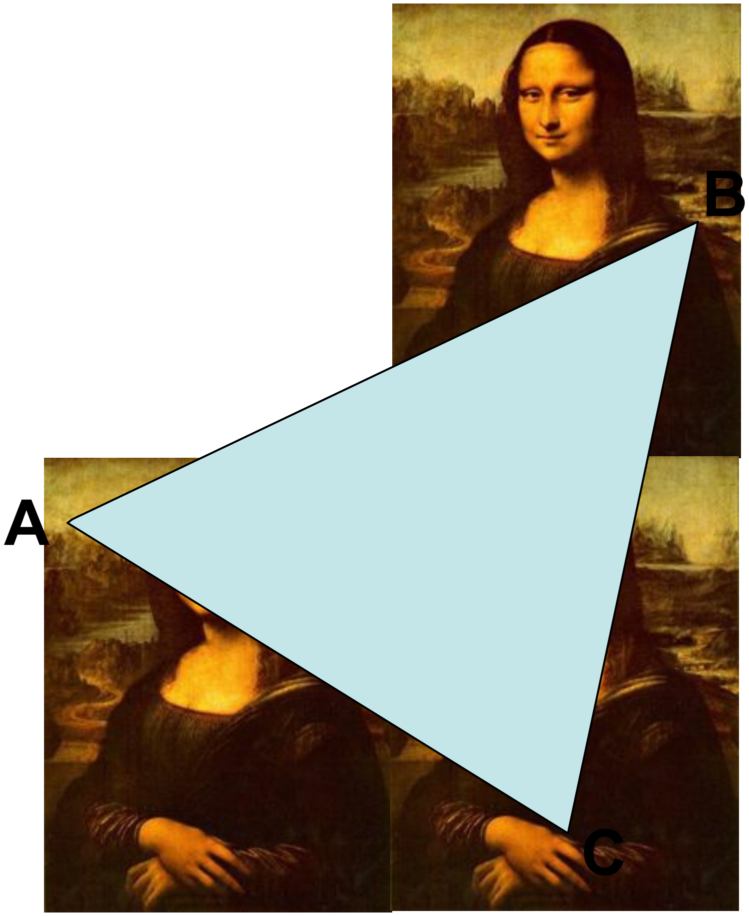


Image to Triangle - 2

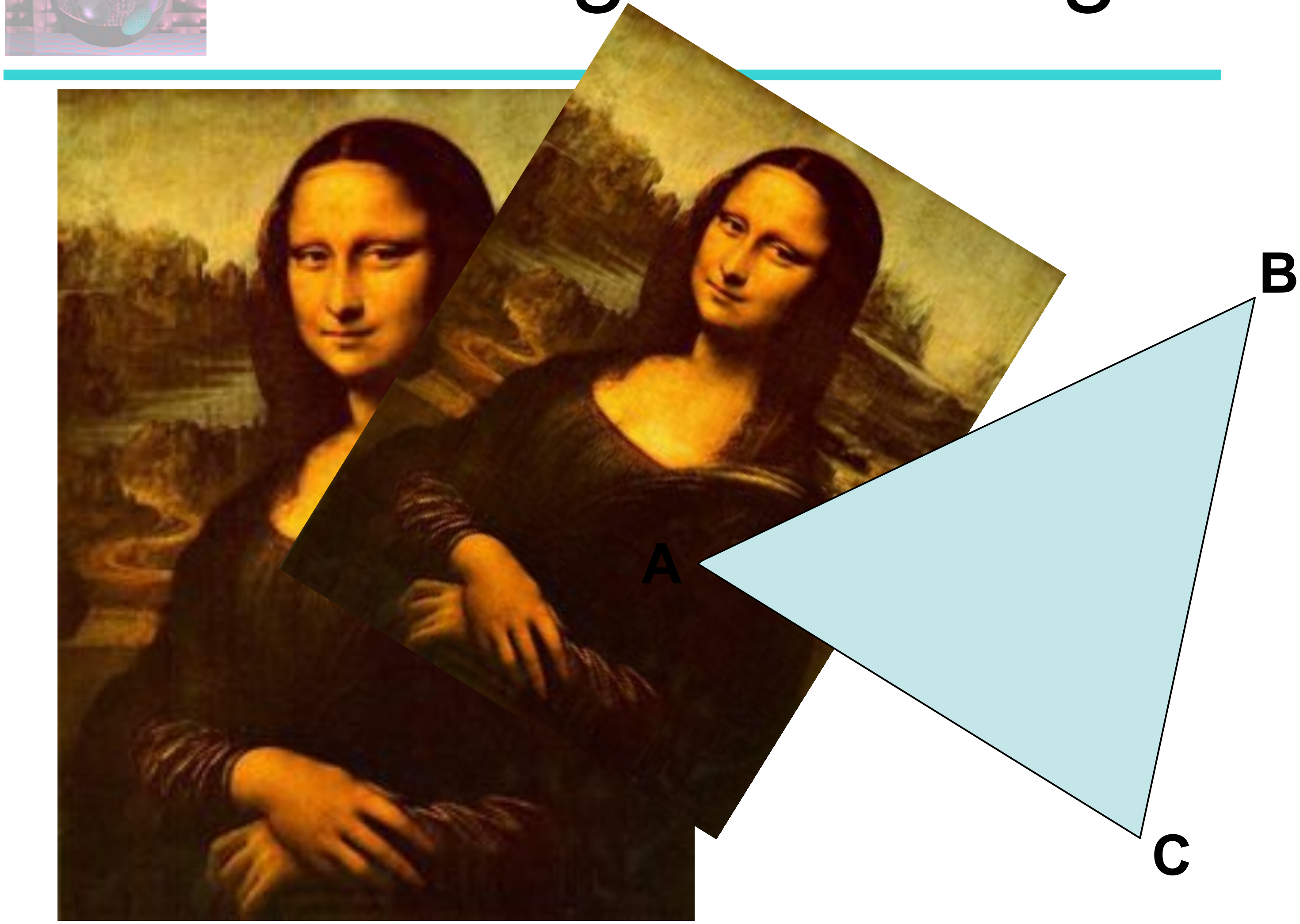
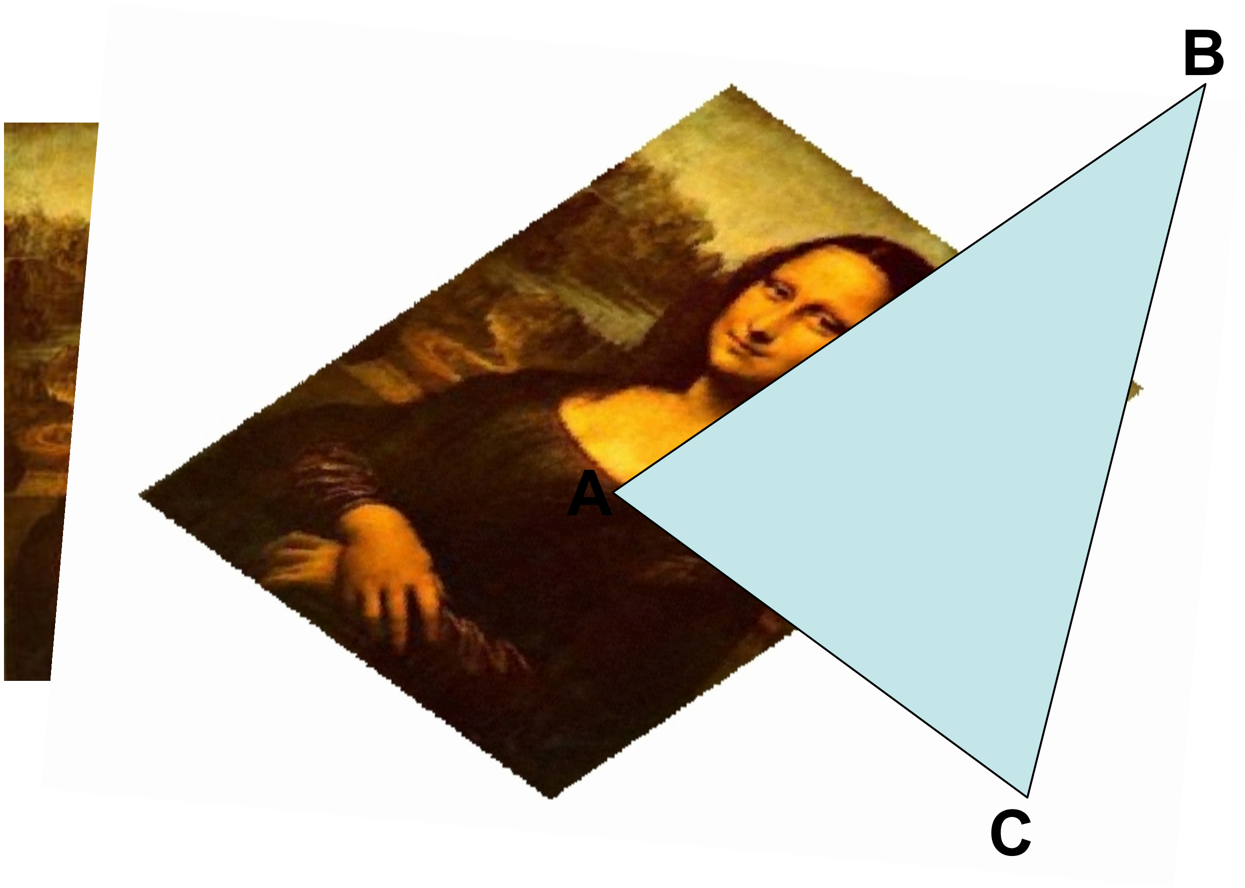
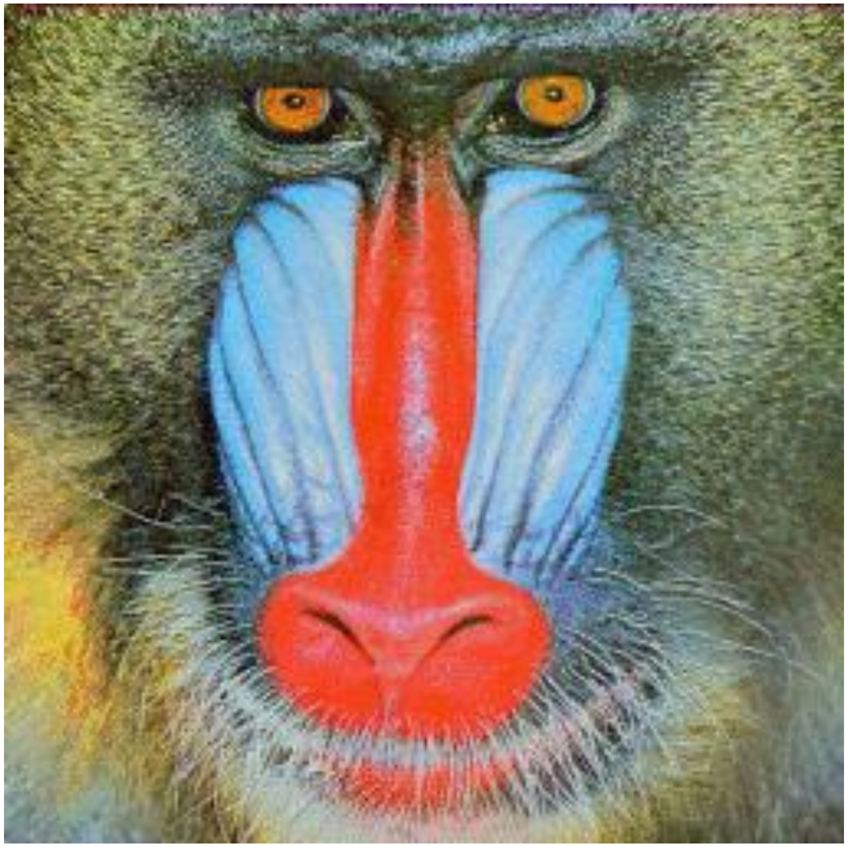
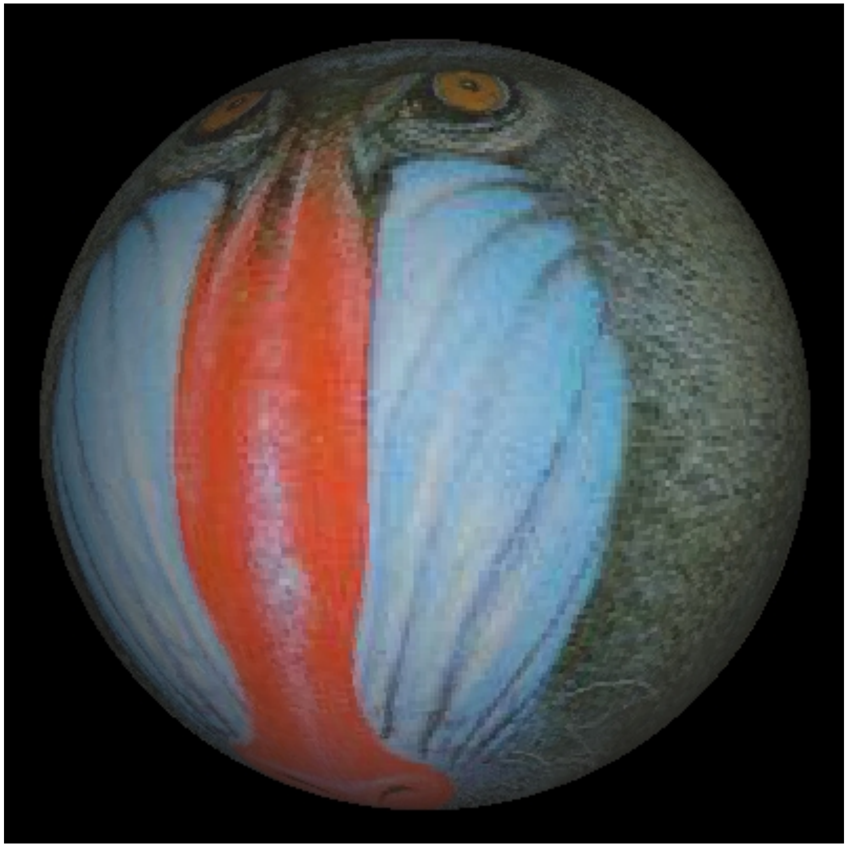


Image to Triangle - 3

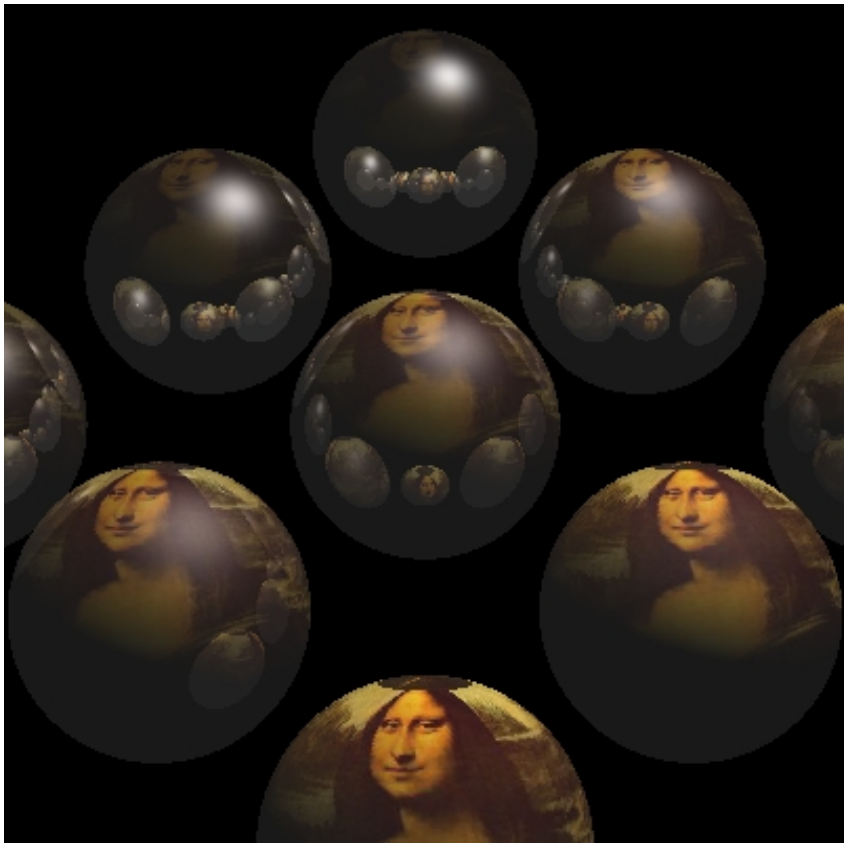


Mandrill Sphere

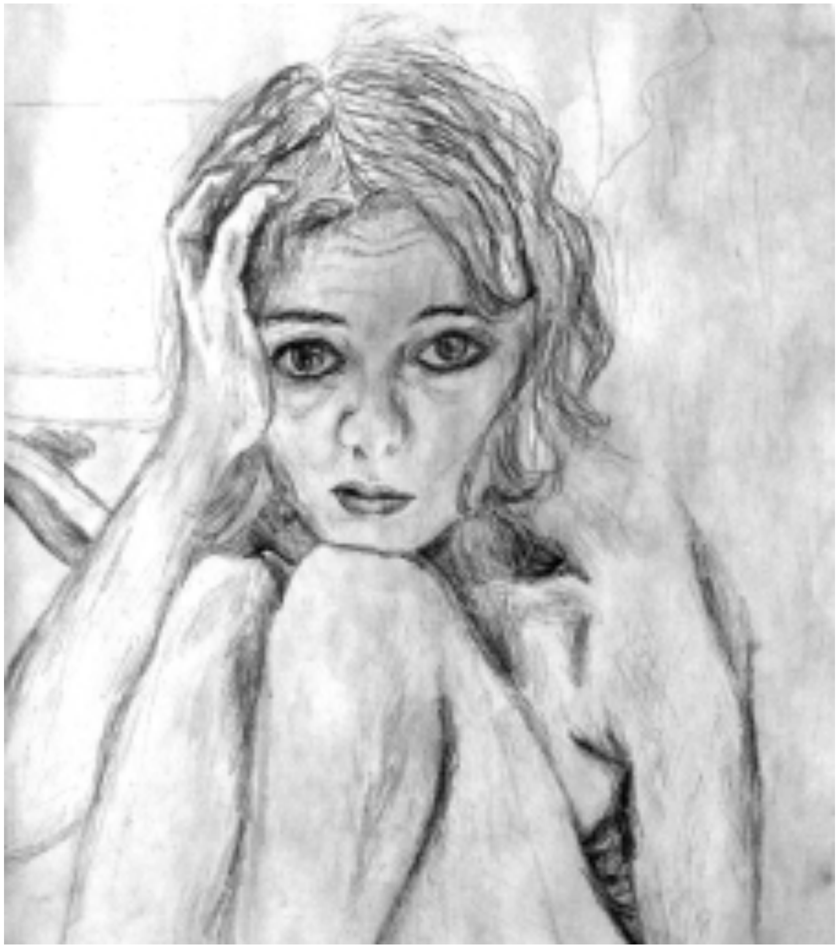


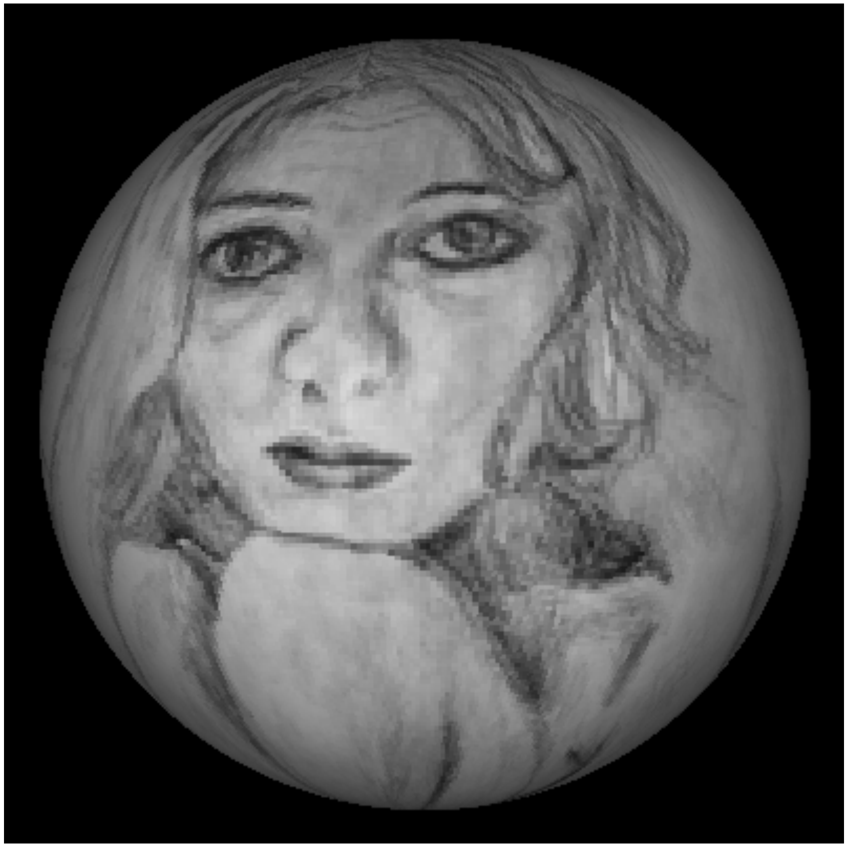


Mona Spheres

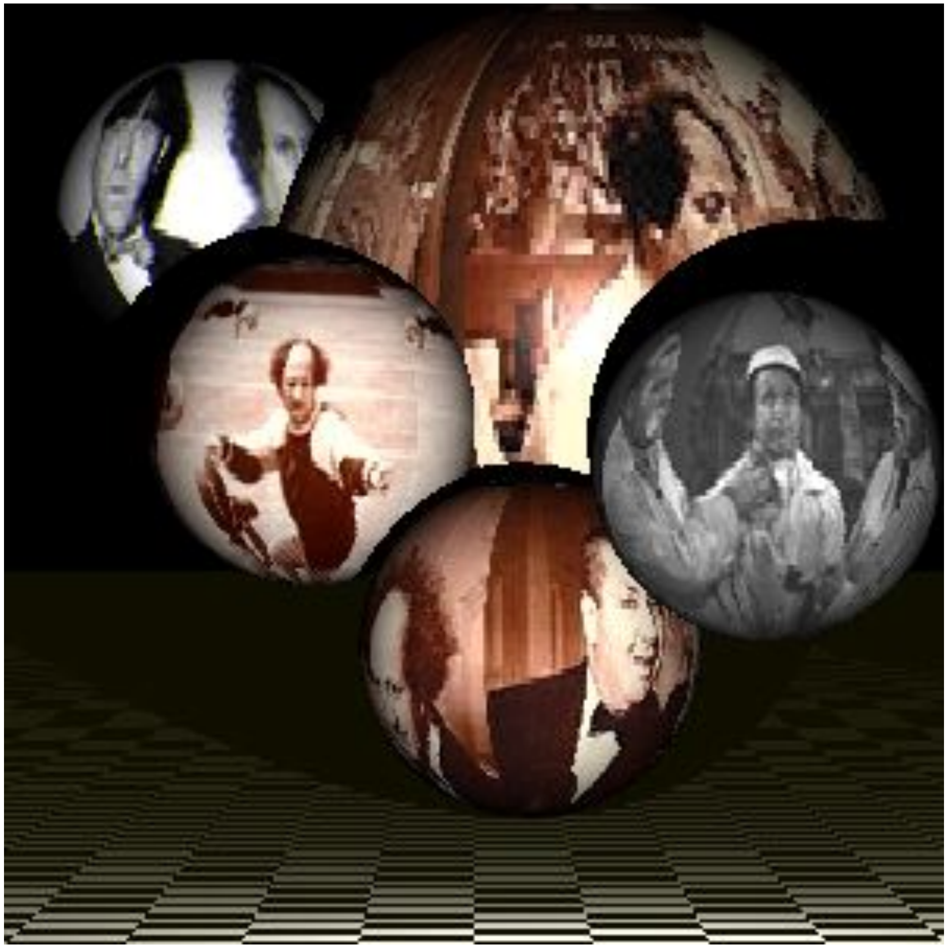


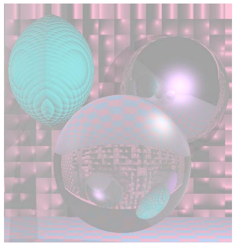
Tova Sphere



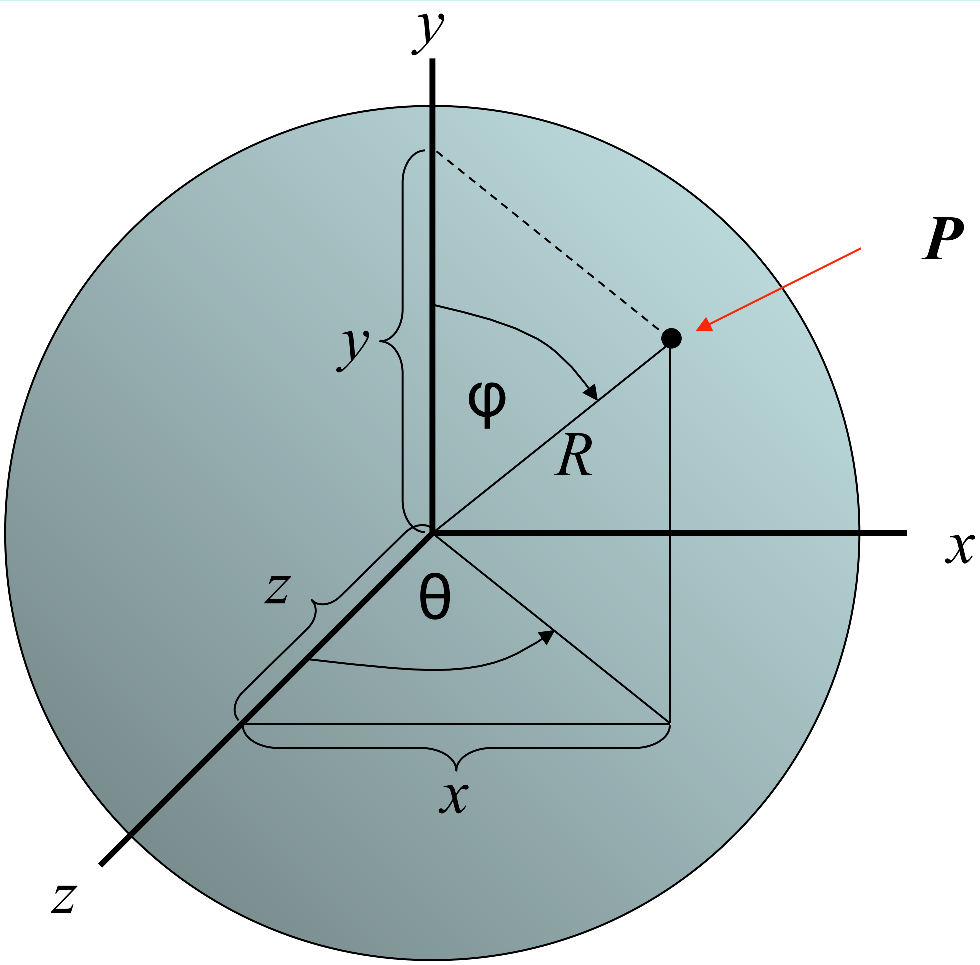


More Textured Spheres





Spherical Geometry



$\boldsymbol{P} = (x, y, z)$

$$\varphi = \arccos(y/R)$$

$$\theta = \arctan(x/z)$$

```
// for texture map – in lieu of using sphere color
double phi, theta;     // for spherical coordinates
double x, y, z;    // sphere vector coordinates
int h, v;       // ppm buffer coordinates
Vector3D V;

        V = SP - theSpheres[hitObject].center;
        V.Get(x, y, z);
        phi = acos(y/theSpheres[hitObject].radius);
        if (z != 0) theta = atan(x/z); else phi = 0;   // ???
        v = (phi)*ppmH/pi;
        h = (theta + pi/2)*ppmW/pi;

        if (v < 0) v = 0; else if (v >= ppmH) v = ppmH - 1;
        v = ppmH -v -1; //v = (v + 85*ppmH/100)%ppmH;//9
        if (h < 0) h = 0; else if (h >= ppmW) h = ppmW - 1;
        h = ppmW -h -1;      //h = (h + 1*ppmW/10)%ppmW;

        rd  = fullFactor*((double)(byte)myImage[h][v][0]/255); clip(rd);
        gd  = fullFactor*((double)(byte)myImage[h][v][1]/255); clip(gd);
        bd  = fullFactor*((double)(byte) myImage[h][v][2]/255); clip(bd);
```