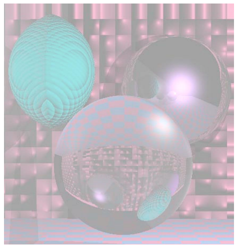


CS5310
Graduate Computer Graphics

Prof. Harriet Fell

Spring 2011

Lecture 5 – February 16, 2011

Comments

- “NOTHING else” means nothing else.
- Do you want your pictures on the web?
  - If not, please send me an email.

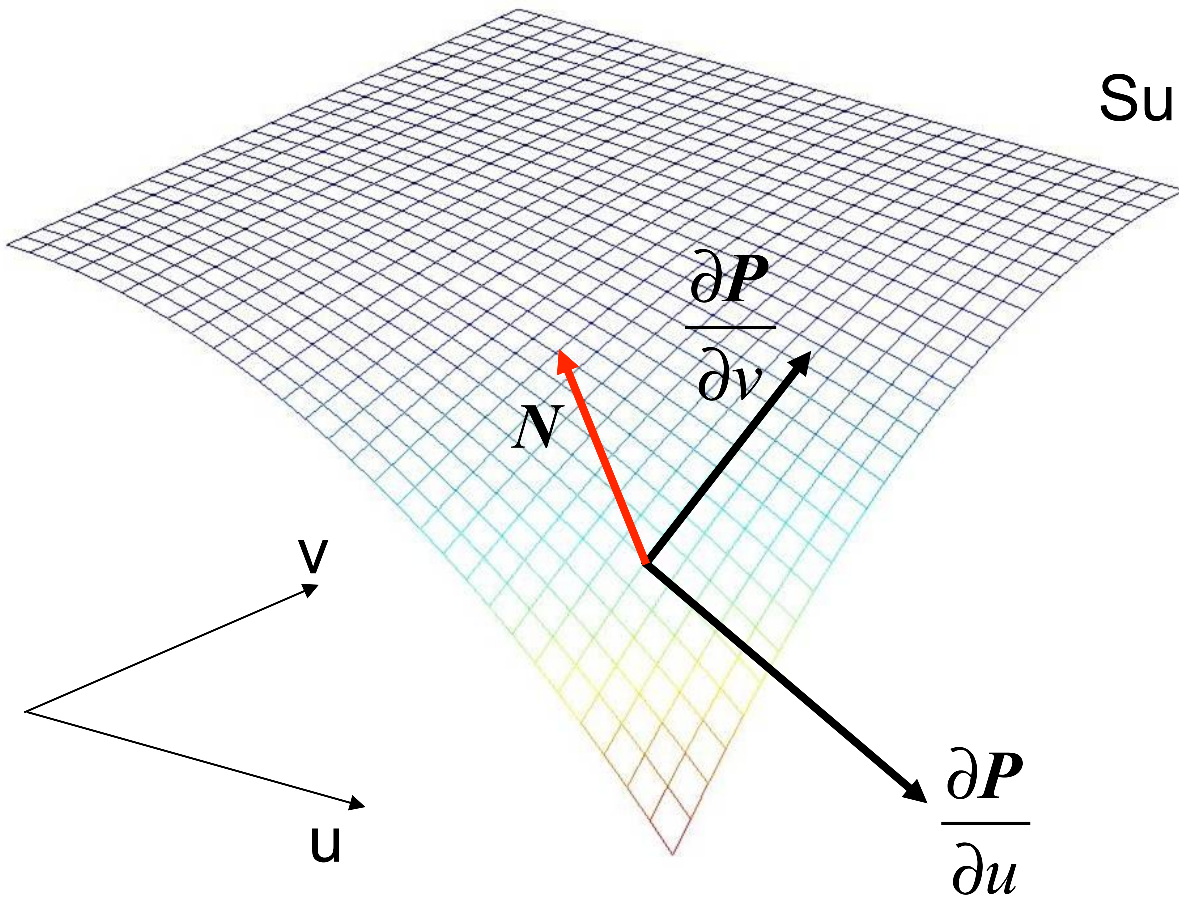
Today's Topics

- Bump Maps
- Texture Maps

- 2D-Viewport Clipping
  - Cohen-Sutherland
  - Liang-Barsky

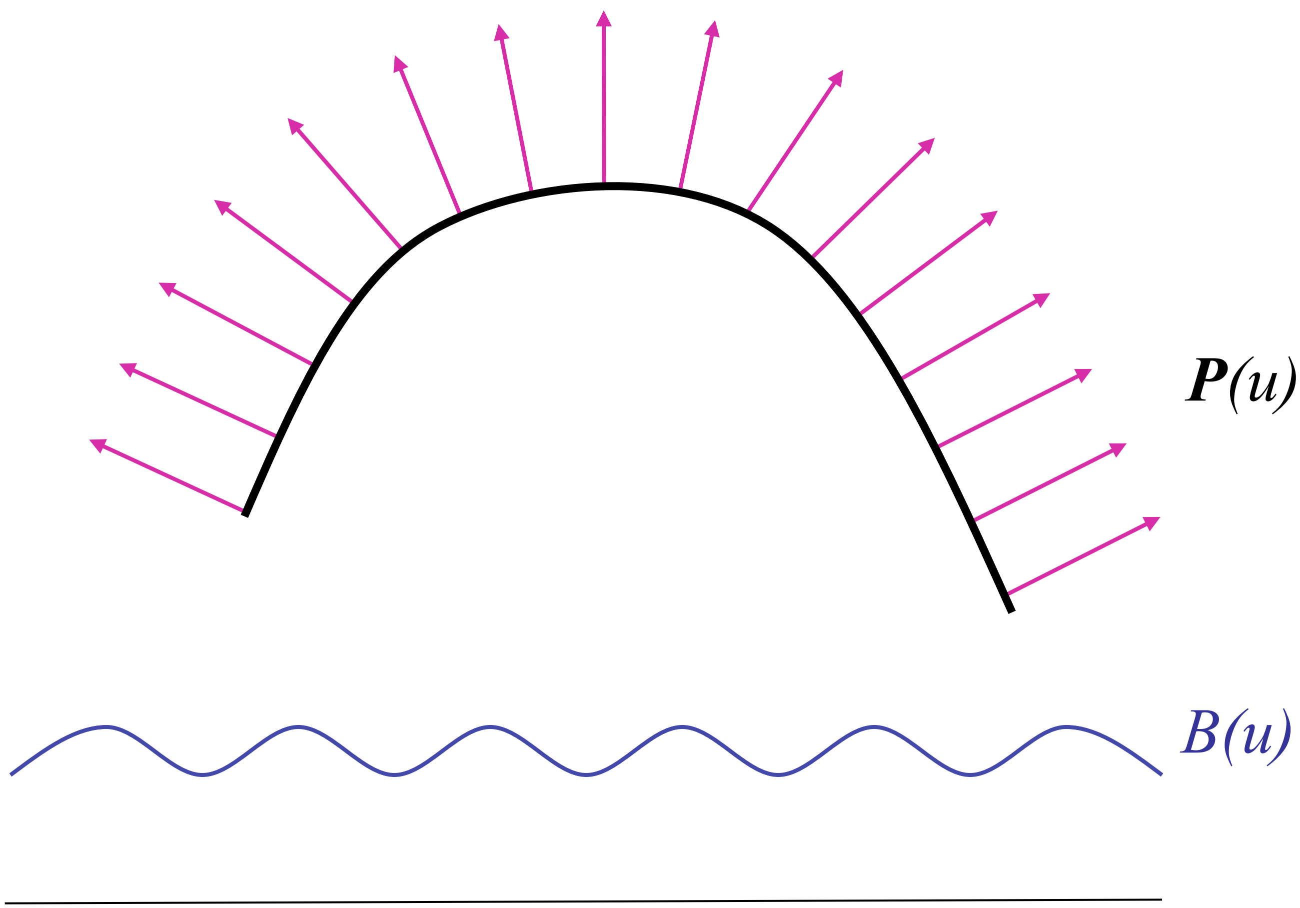
Bump Maps - Blinn 1978

Surface $\boldsymbol{P}(u, v)$

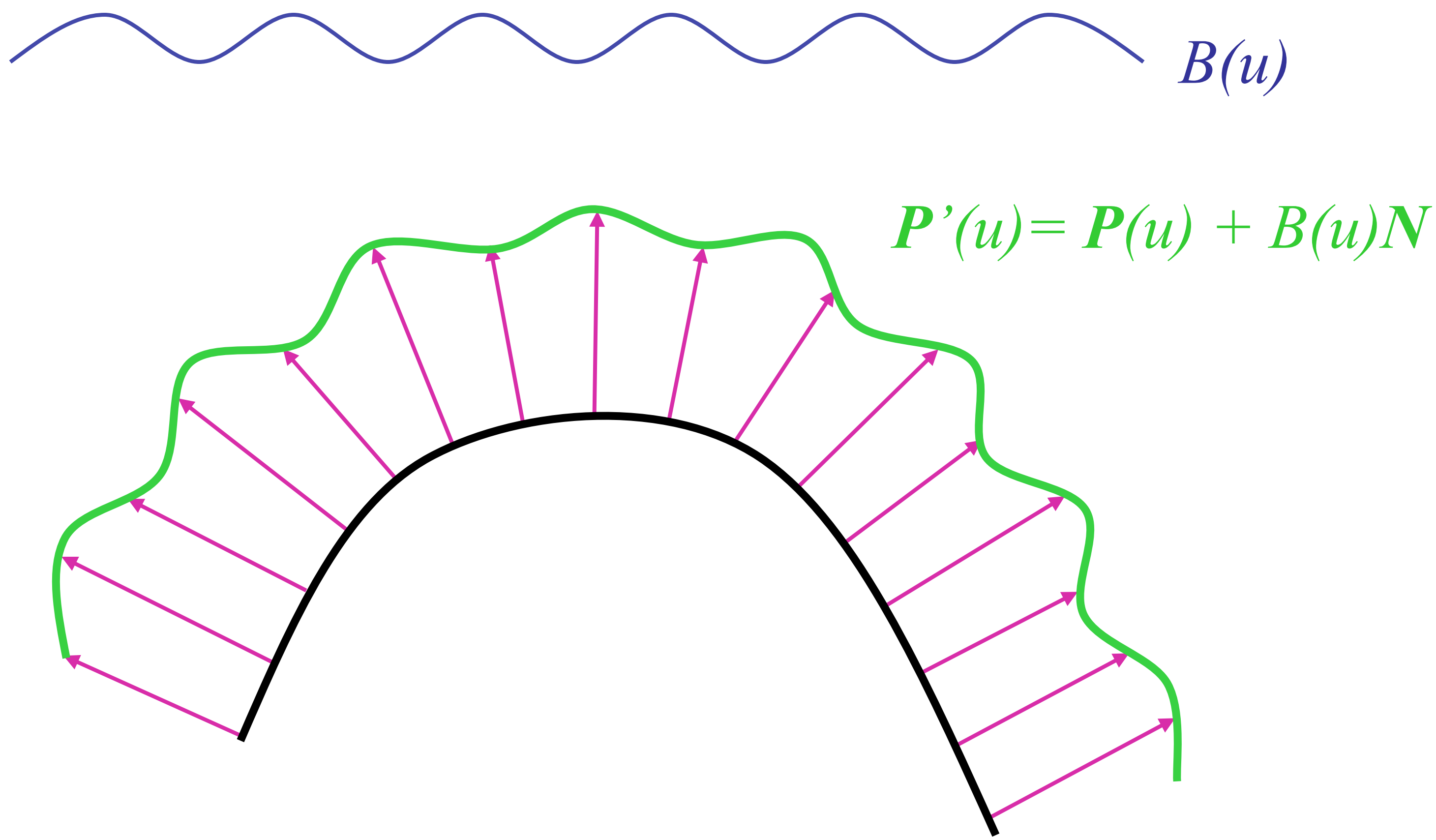


$$\boldsymbol{N} = \frac{\partial \boldsymbol{P}}{\partial u} \times \frac{\partial \boldsymbol{P}}{\partial v}$$

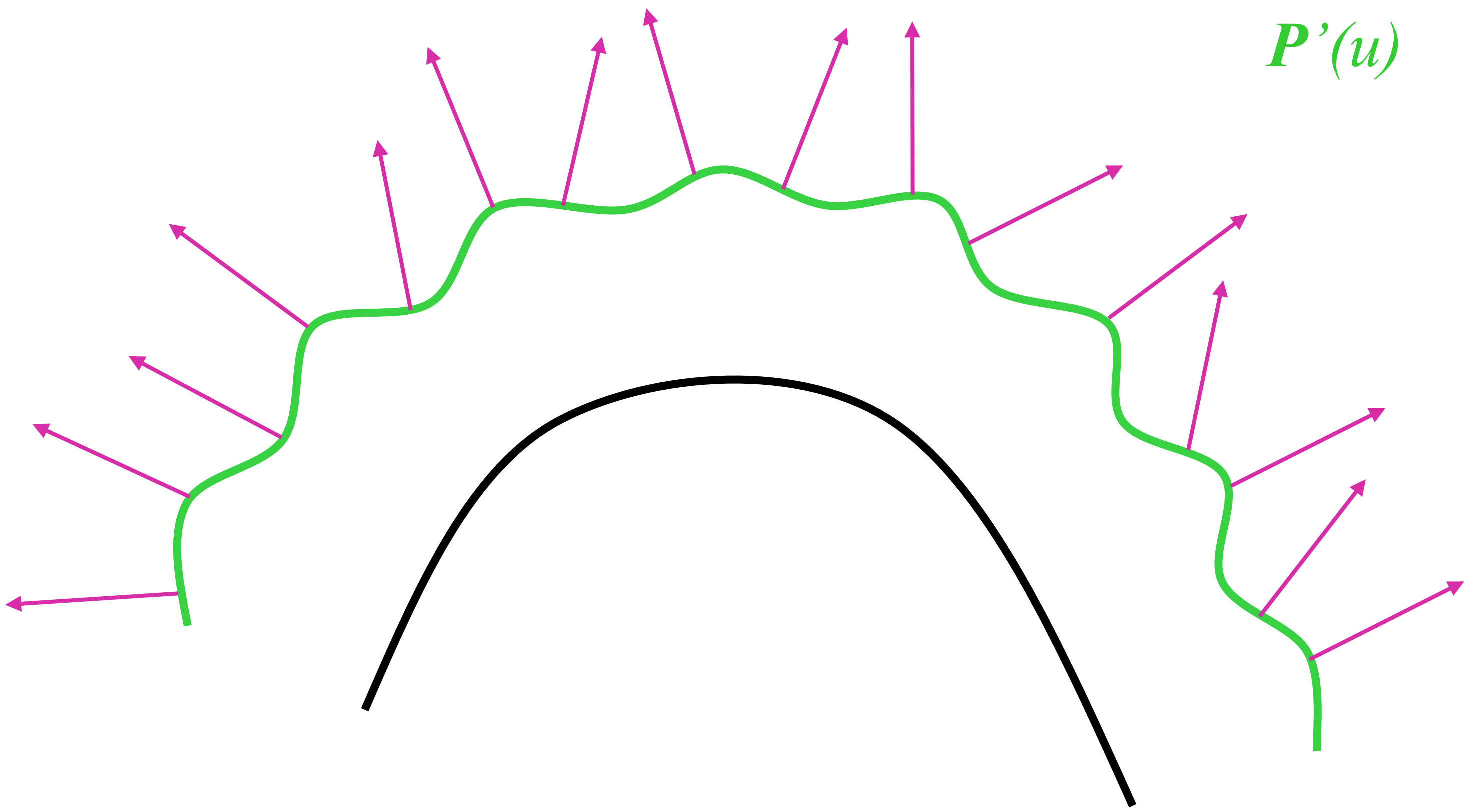
One dimensional Example

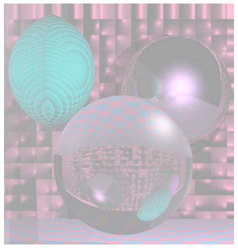


The New Surface



The New Surface Normals





Bump Maps - Formulas

A parametric Surface $\left(x(u,v), y(u,v), z(u,v)\right) = \boldsymbol{P}(u,v)$

$$\boldsymbol{N} = \frac{\partial \boldsymbol{P}}{\partial u} \times \frac{\partial \boldsymbol{P}}{\partial v}$$

The new surface $\boldsymbol{P}'(u,v) = \boldsymbol{P}(u,v) + B(u,v)\boldsymbol{N}$

$$\boldsymbol{N}' = \boldsymbol{P}'_u \times \boldsymbol{P}'_v$$

$$\boldsymbol{P}'_u = \boldsymbol{P}_u + B_u\boldsymbol{N} + B(u,v)\boldsymbol{N}_u$$

$$\boldsymbol{P}'_v = \boldsymbol{P}_v + B_v\boldsymbol{N} + B(u,v)\boldsymbol{N}_v$$

The New Normal

$$\boldsymbol{N}' = \left(\boldsymbol{P}_u + B_u\boldsymbol{N} + B(u,v)\boldsymbol{N}_u\right) \times \left(\boldsymbol{P}_v + B_v\boldsymbol{N} + B(u,v)\boldsymbol{N}_v\right)$$

$$= \boldsymbol{P}_u \times \boldsymbol{P}_v + B_v\boldsymbol{P}_u \times \boldsymbol{N} + \cancel{B(u,v)\boldsymbol{P}_u \times \boldsymbol{N}_v}$$

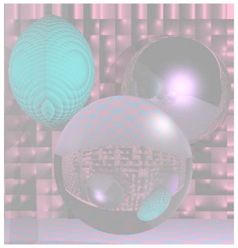
$$+B_u\boldsymbol{N} \times \boldsymbol{P}_v + \cancel{B_uB_v\boldsymbol{N} \times \boldsymbol{N}} + \cancel{B_uB(u,v)\boldsymbol{N} \times \boldsymbol{N}_v}$$

$$+\cancel{B(u,v)\boldsymbol{N}_u \times \boldsymbol{P}_v} + \cancel{B(u,v)B_v\boldsymbol{N}_u \times \boldsymbol{N}} + \cancel{B(u,v)^2\boldsymbol{N}_u \times \boldsymbol{N}_v}$$

This term is 0.

These terms are small if $B(u, v)$ *is small*.

We use $$\boldsymbol{N}' = \boldsymbol{P}_u \times \boldsymbol{P}_v + B_v\boldsymbol{P}_u \times \boldsymbol{N} + B_u\boldsymbol{N} \times \boldsymbol{P}_v$$



Tweaking the Normal Vector

$$
\begin{aligned}
\boldsymbol{N}' &= \boldsymbol{P}_u \times \boldsymbol{P}_v + B_v \boldsymbol{P}_u \times \boldsymbol{N} + B_u \boldsymbol{N} \times \boldsymbol{P}_v \\
&= \boldsymbol{N} + B_v \boldsymbol{P}_u \times \boldsymbol{N} + B_u \boldsymbol{N} \times \boldsymbol{P}_v
\end{aligned}
$$

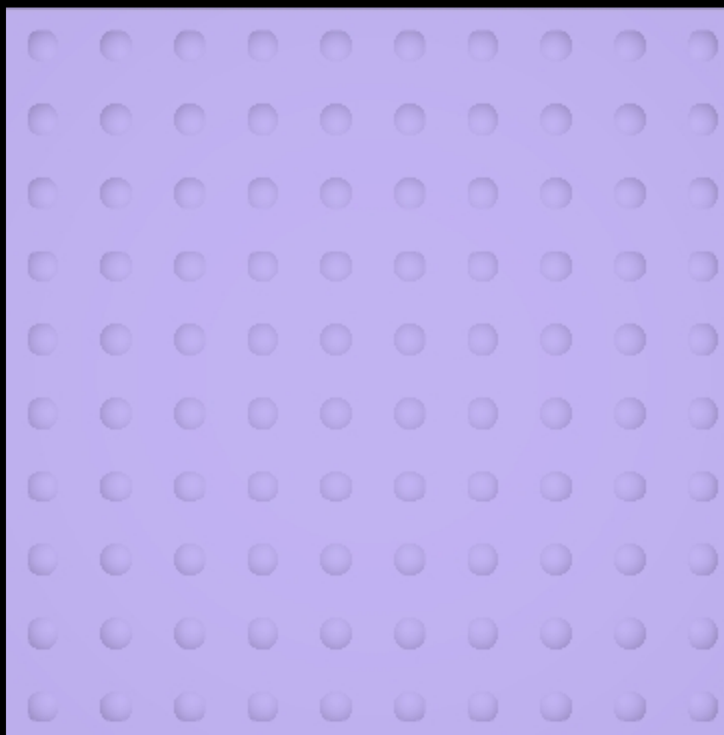
$\boldsymbol{A} = \boldsymbol{N} \times \boldsymbol{P}_v \qquad \boldsymbol{B} = \boldsymbol{N} \times \boldsymbol{P}_u$

$\boldsymbol{D} = B_u \boldsymbol{A} - B_v \boldsymbol{B}$ is the difference vector.

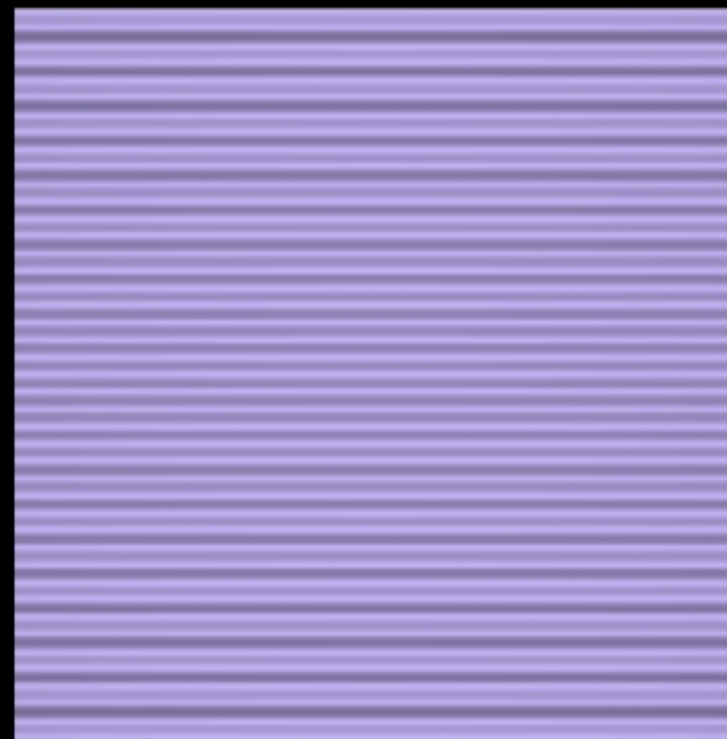
$$\boldsymbol{N}' = \boldsymbol{N} + \boldsymbol{D}$$

$\boldsymbol{D}$ lies in the tangent plane to the surface.

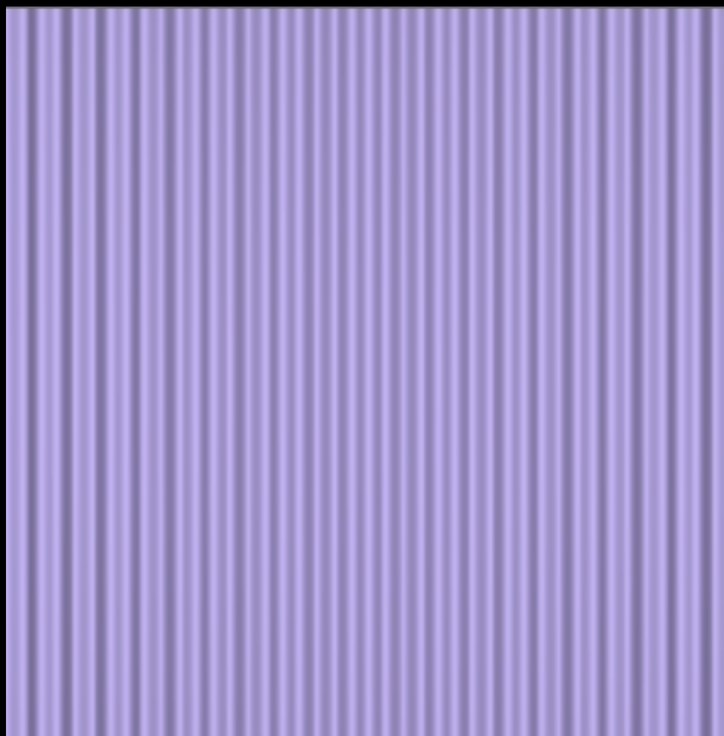
Plane with Spheres



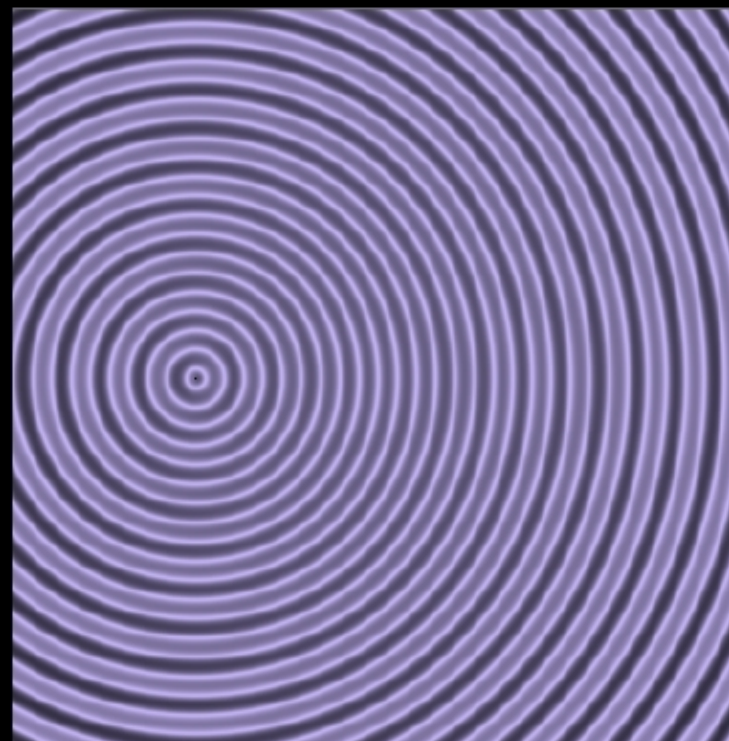
Plane with Horizontal Wave



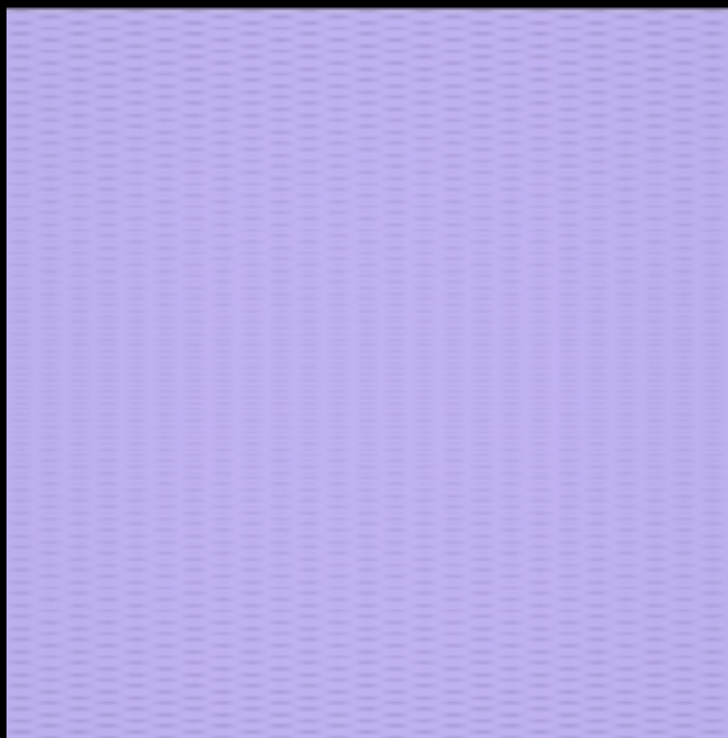
Plane with Vertical Wave



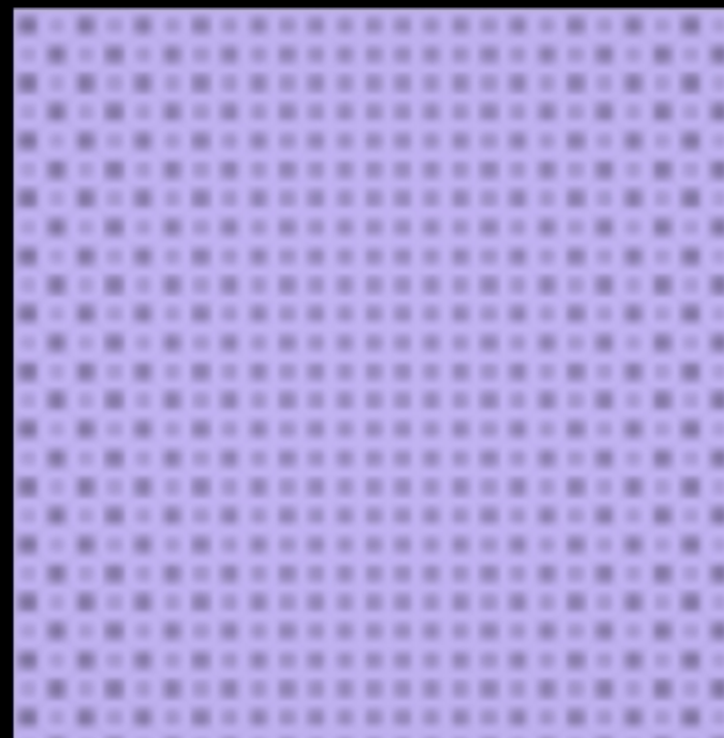
Plane with Ripple



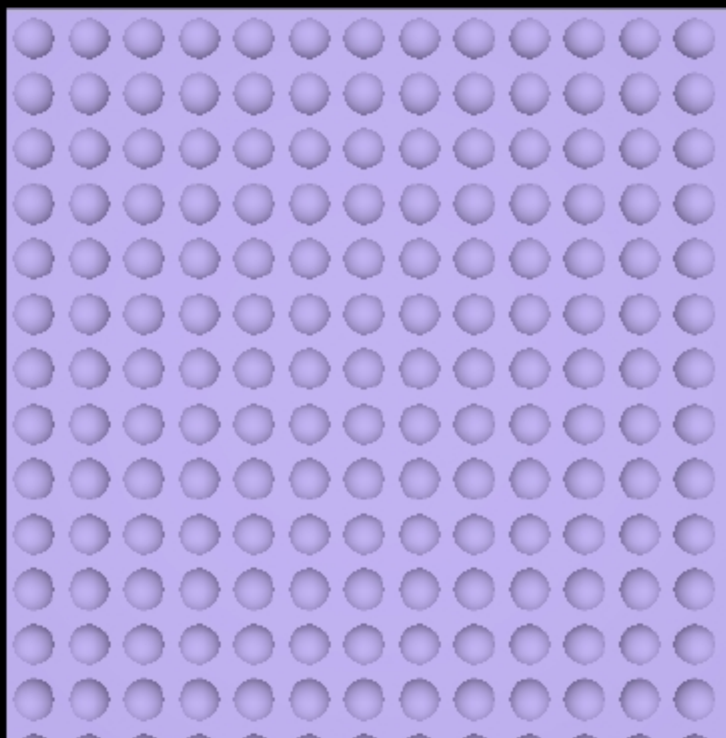
Plane with Mesh



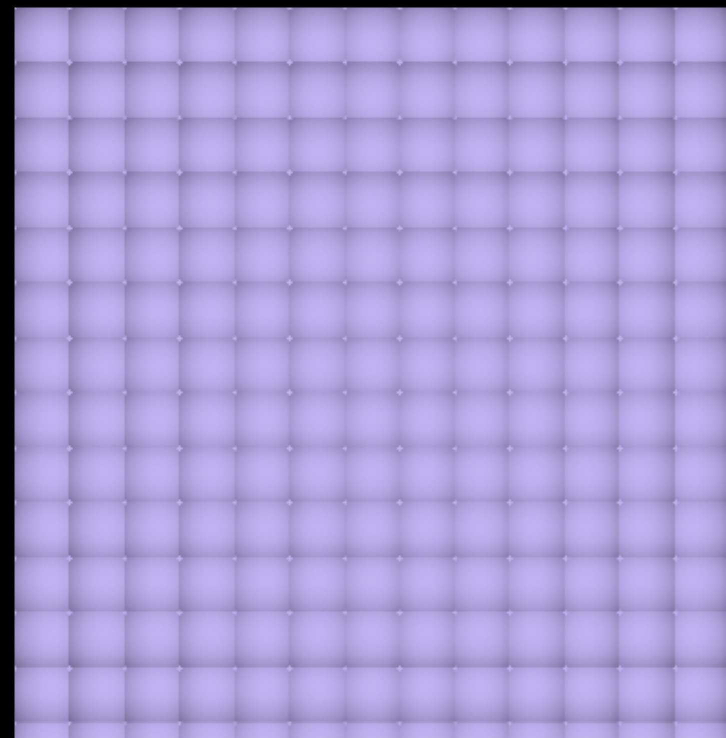
Plane with Waffle



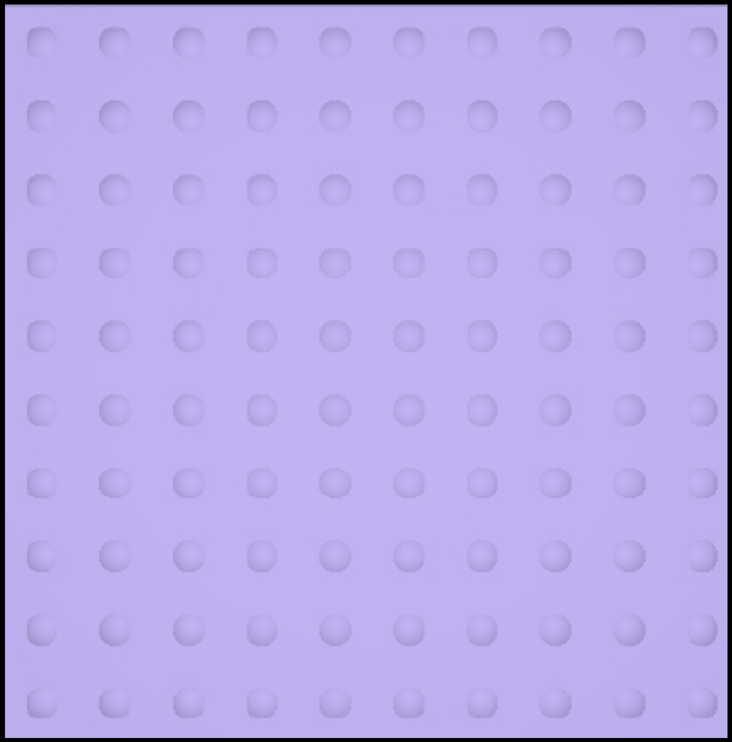
Plane with Dimples

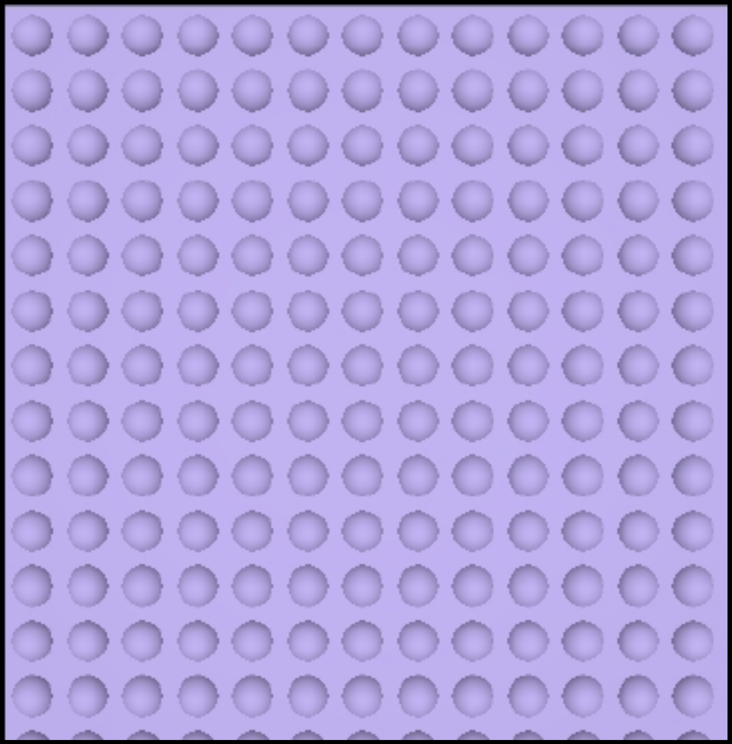


Plane with Squares

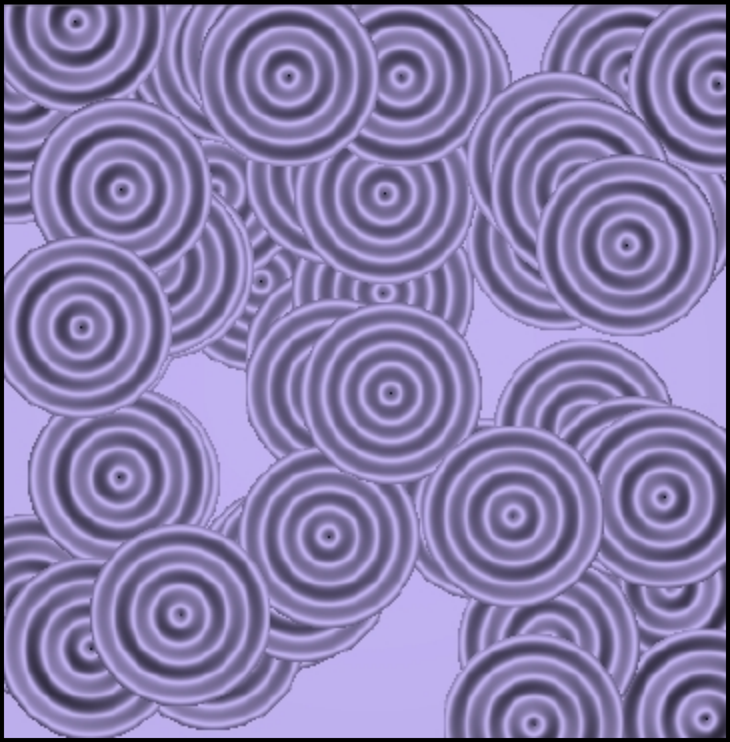


Dots and Dimples

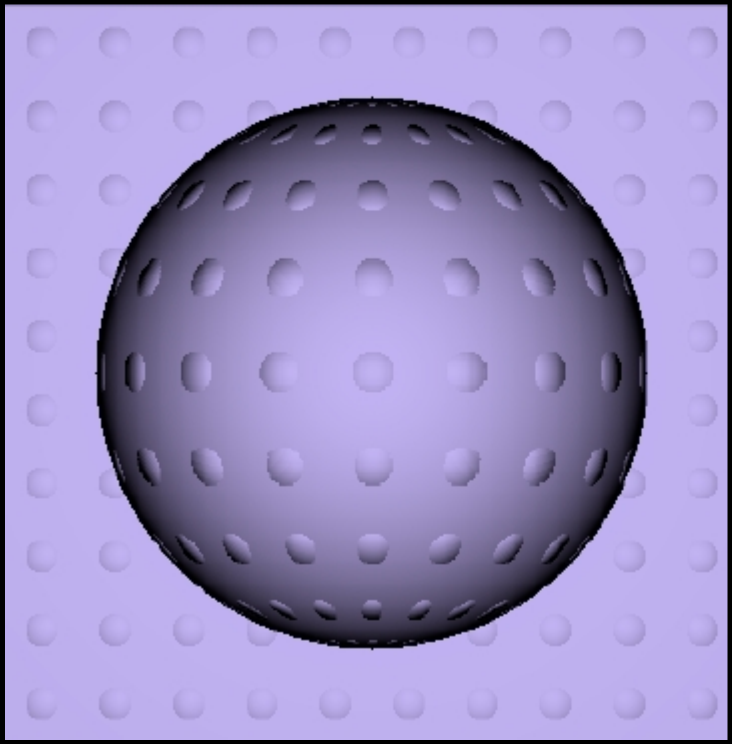




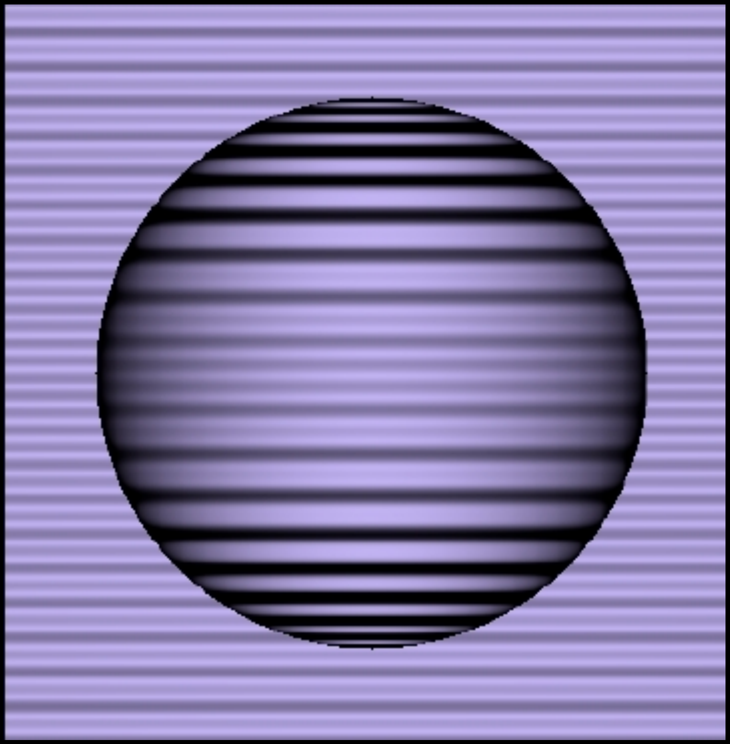
Plane with Ripples



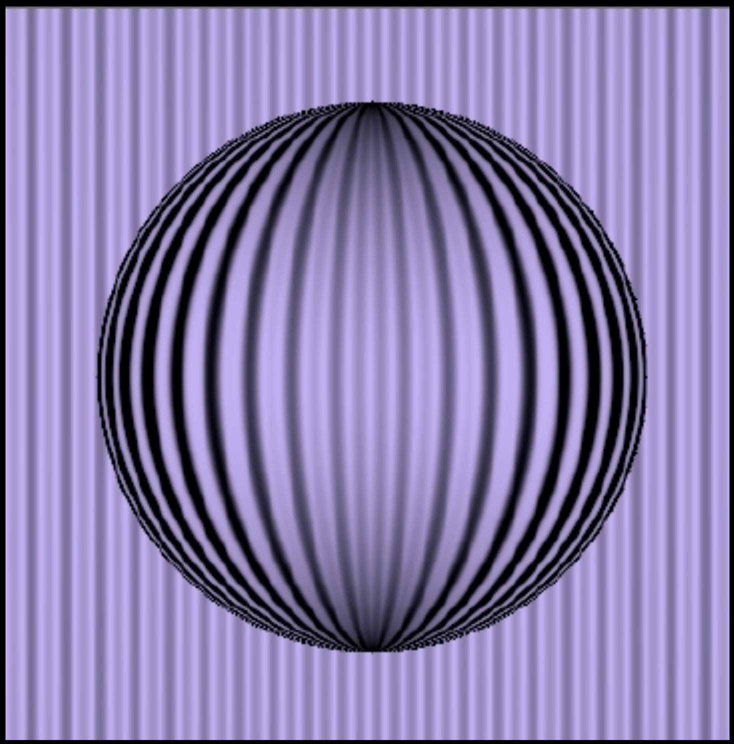
Sphere on Plane with Spheres



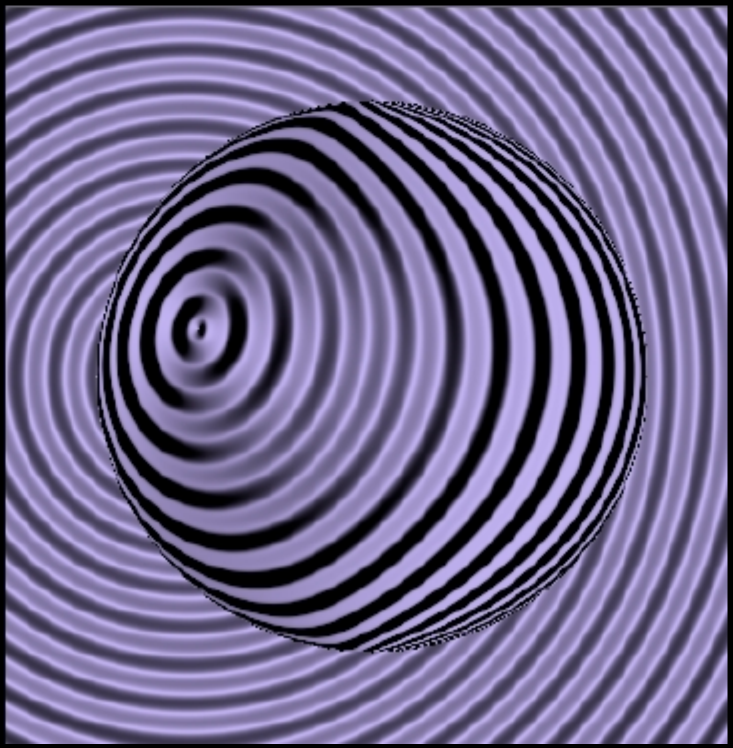
Sphere on Plane with Horizontal Wave



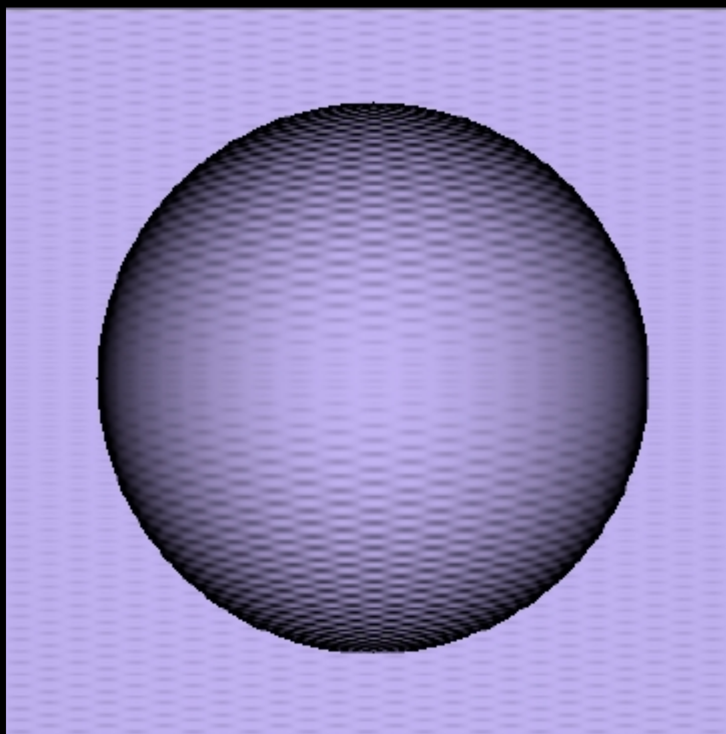
Sphere on Plane with Vertical Wave



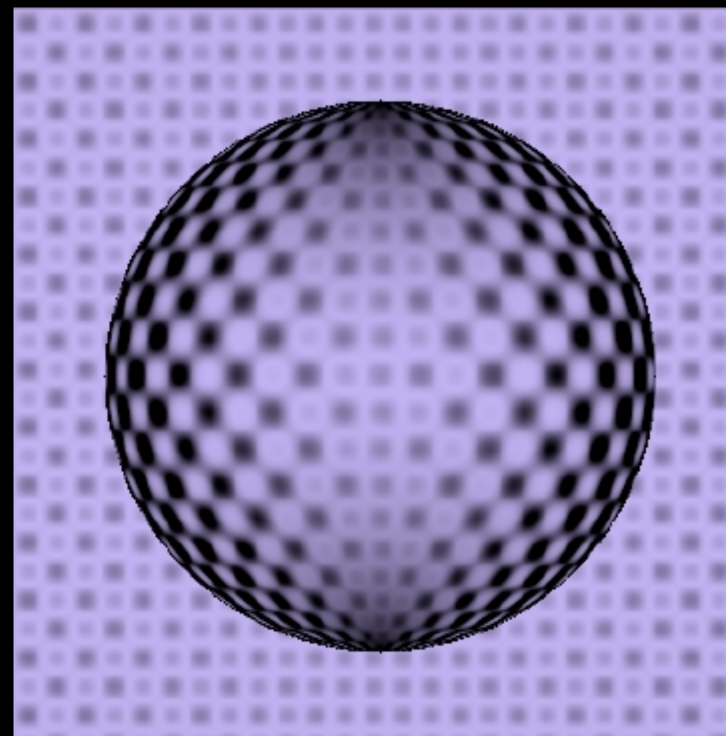
Sphere on Plane with Ripple



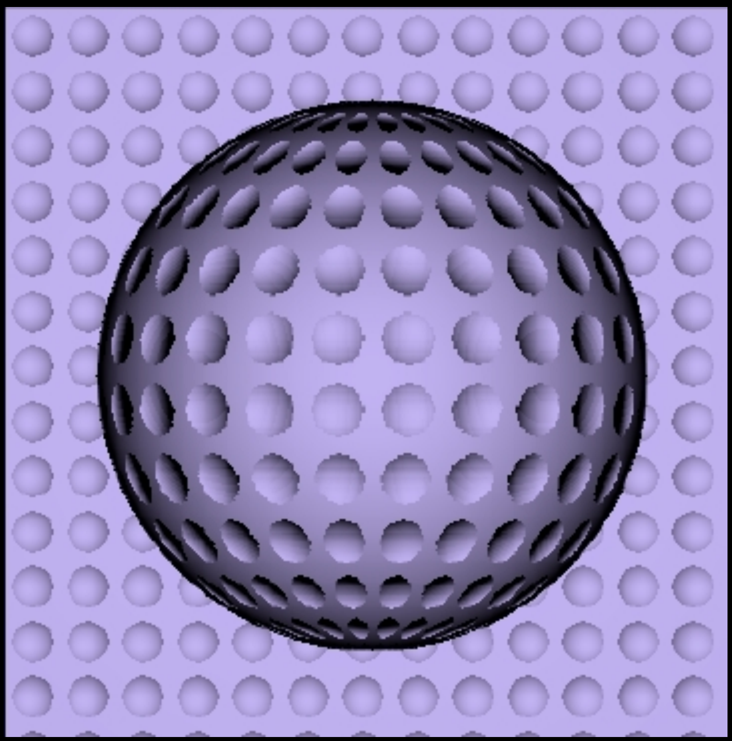
Sphere on Plane with Mesh



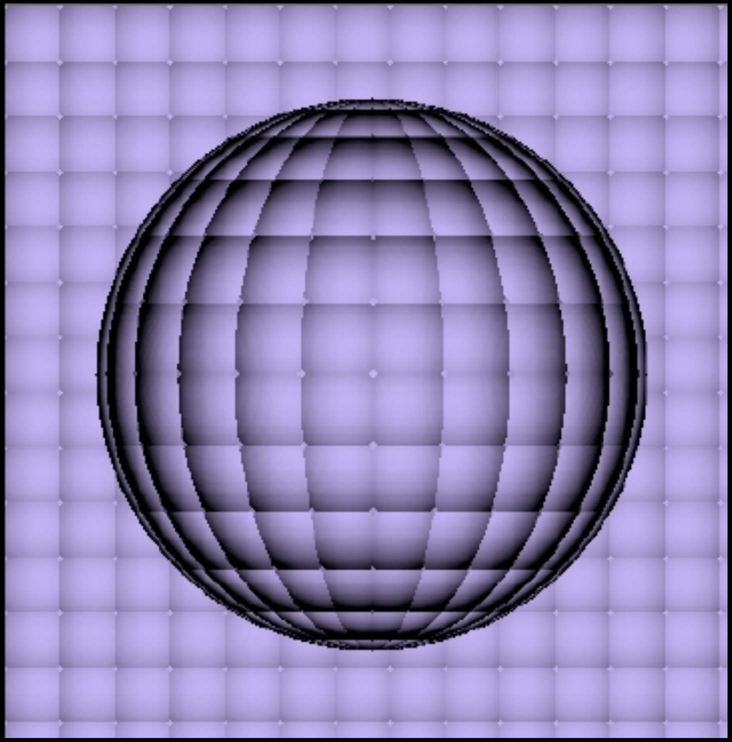
Sphere on Plane with Waffle



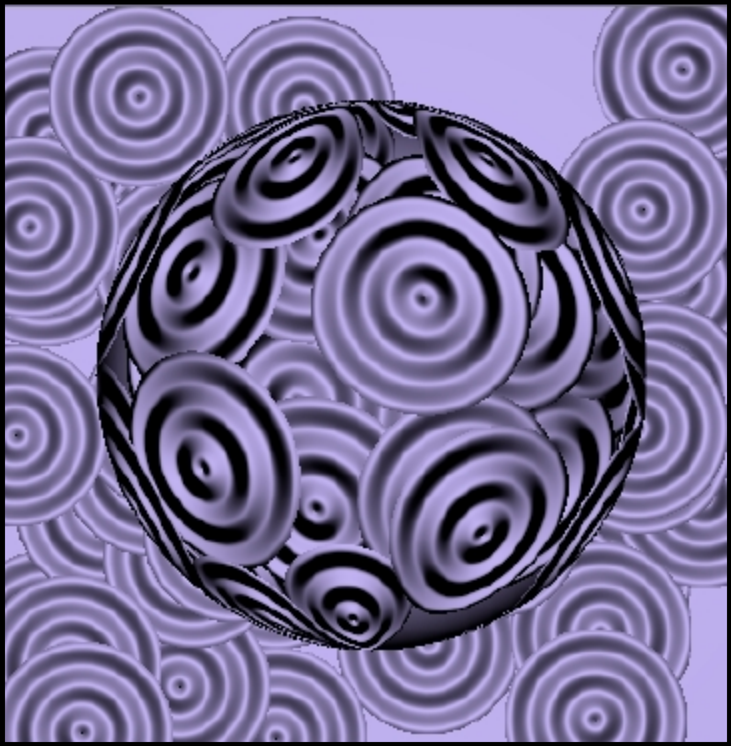
Sphere on Plane with Dimples



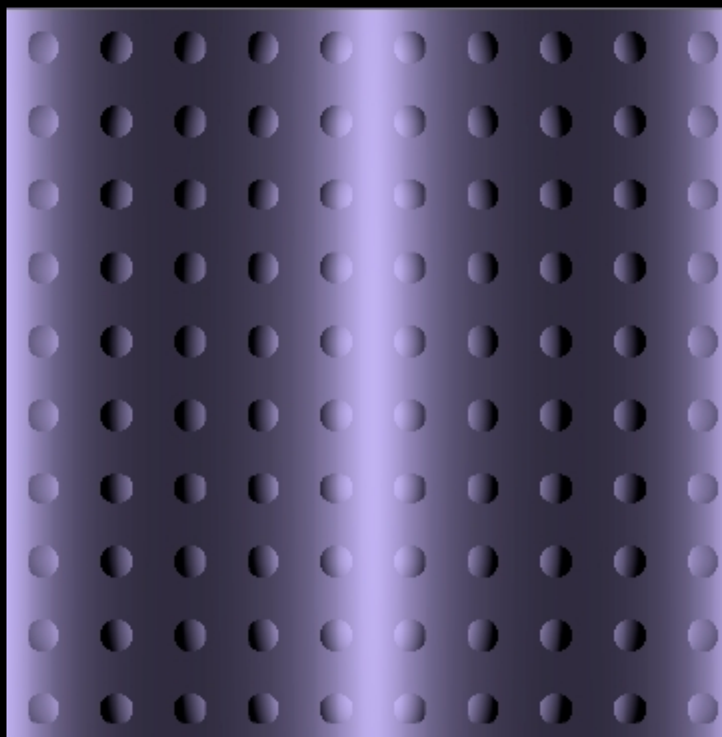
Sphere on Plane with Squares



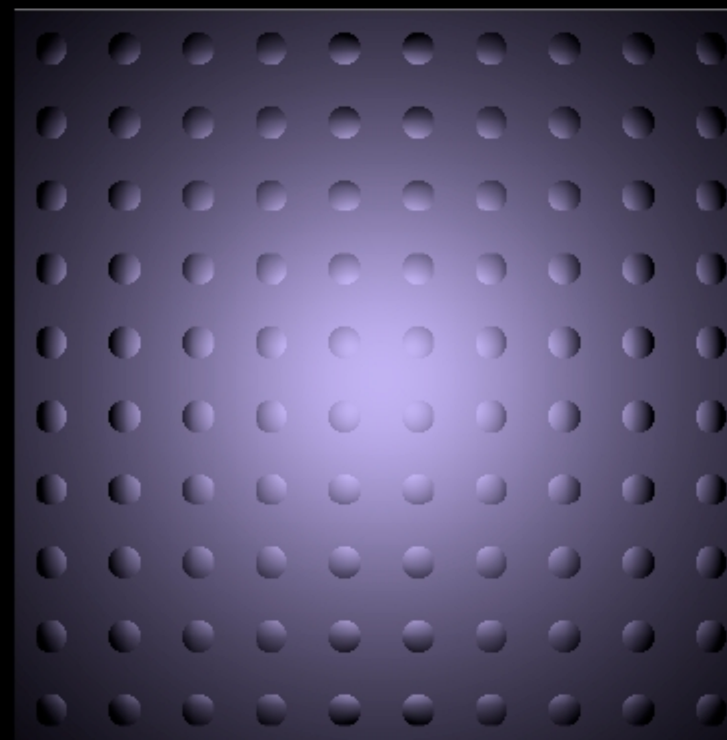
Sphere on Plane with Ripples



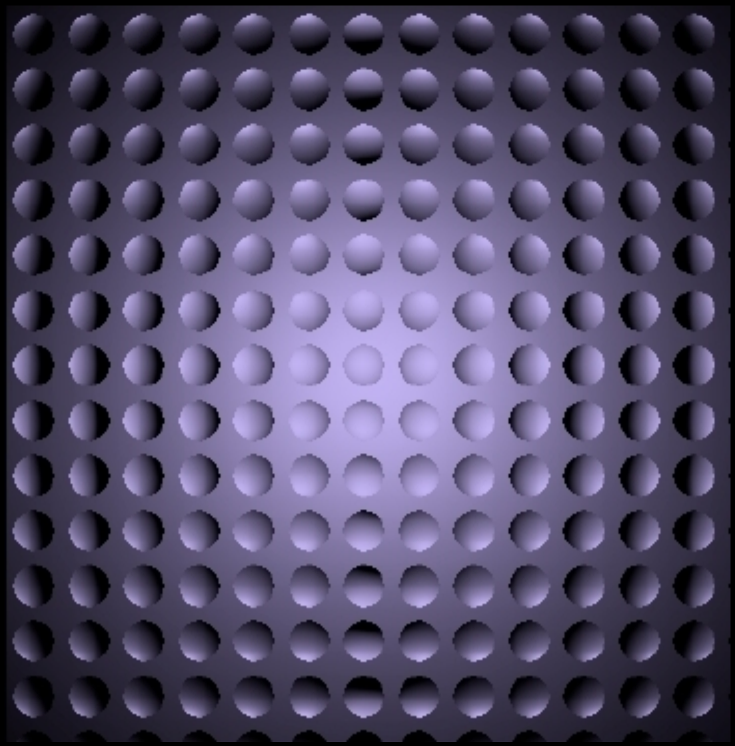
Wave with Spheres



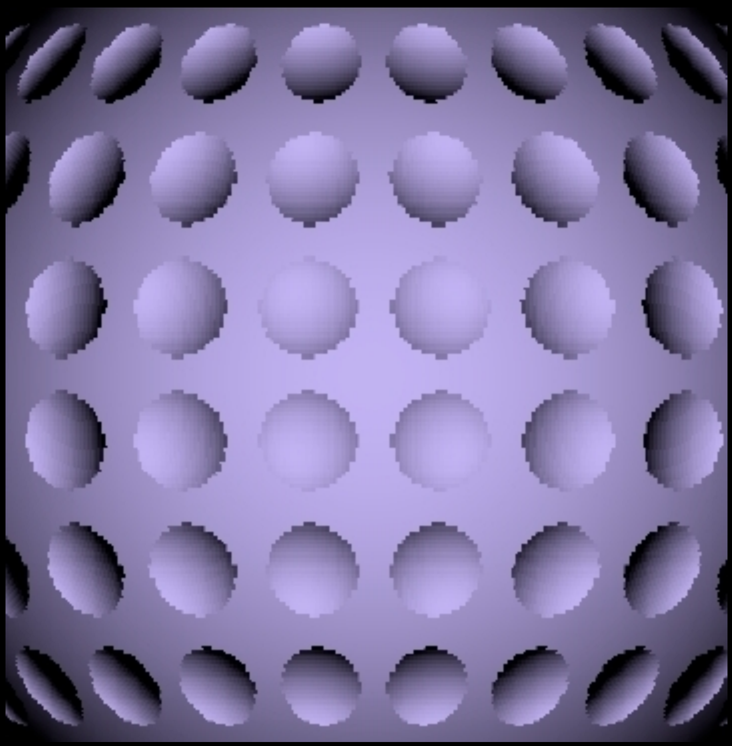
Parabola with Spheres



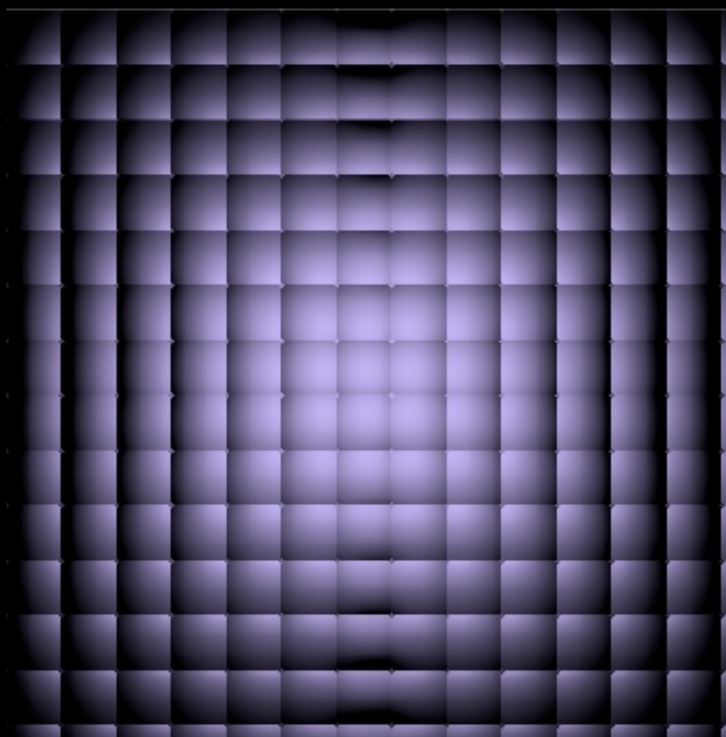
Parabola with Dimples



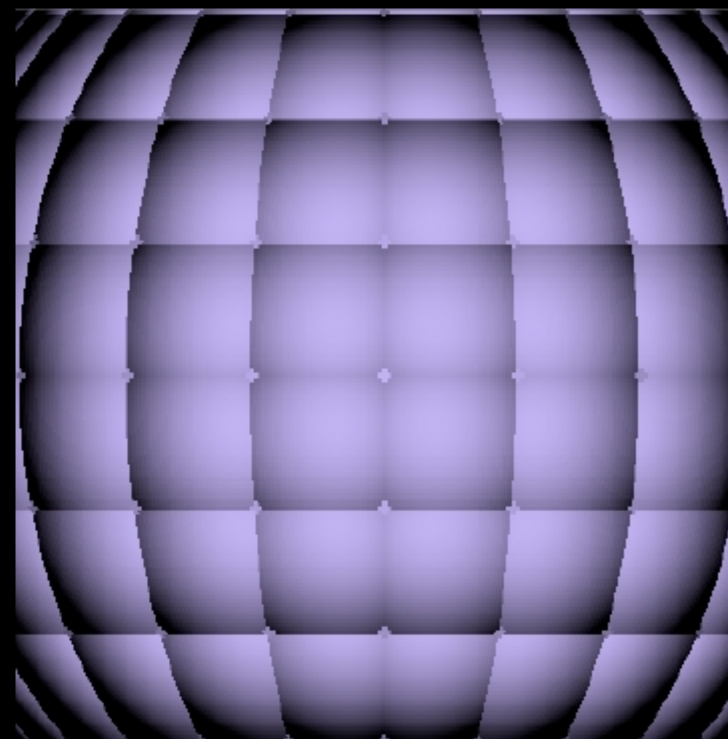
Big Sphere with Dimples



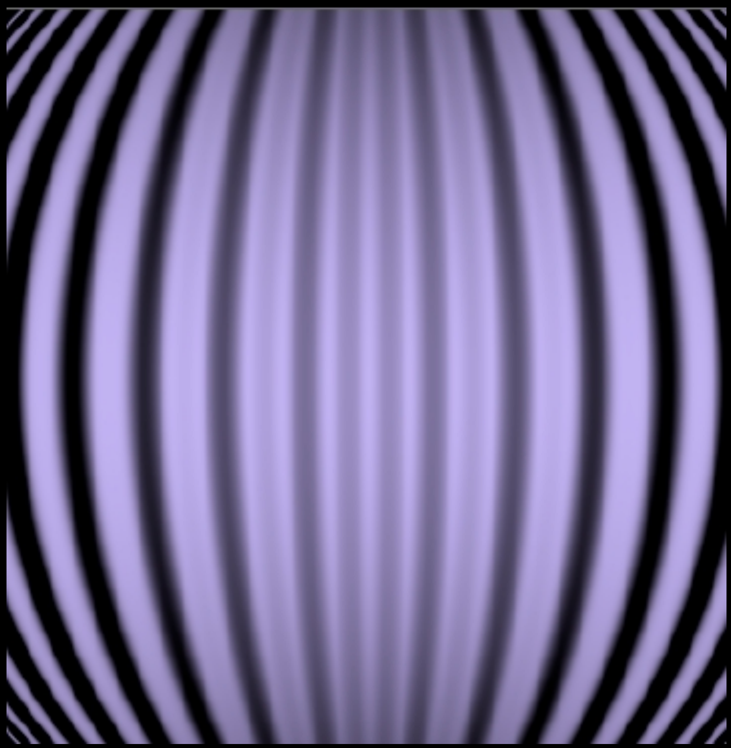
Parabola with Squares



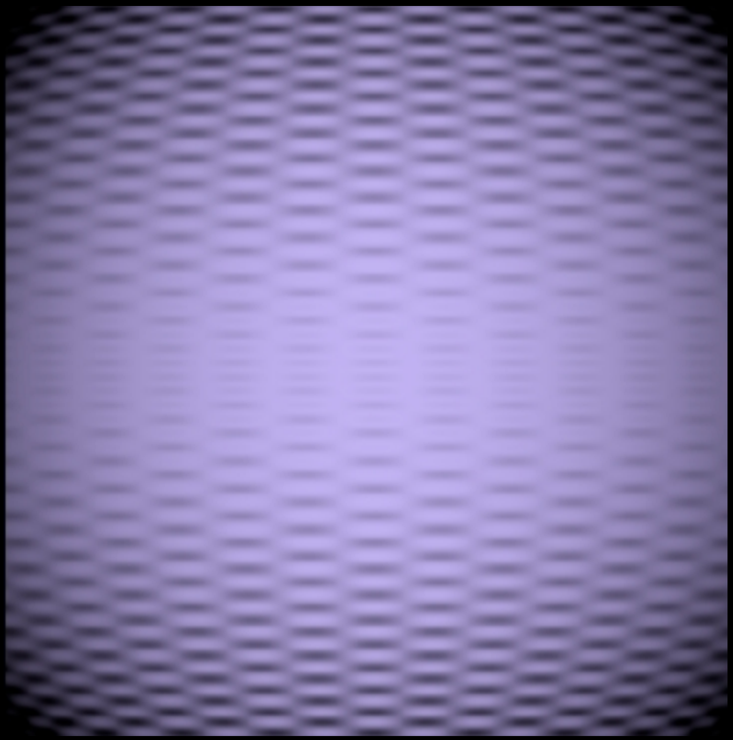
Big Sphere with Squares



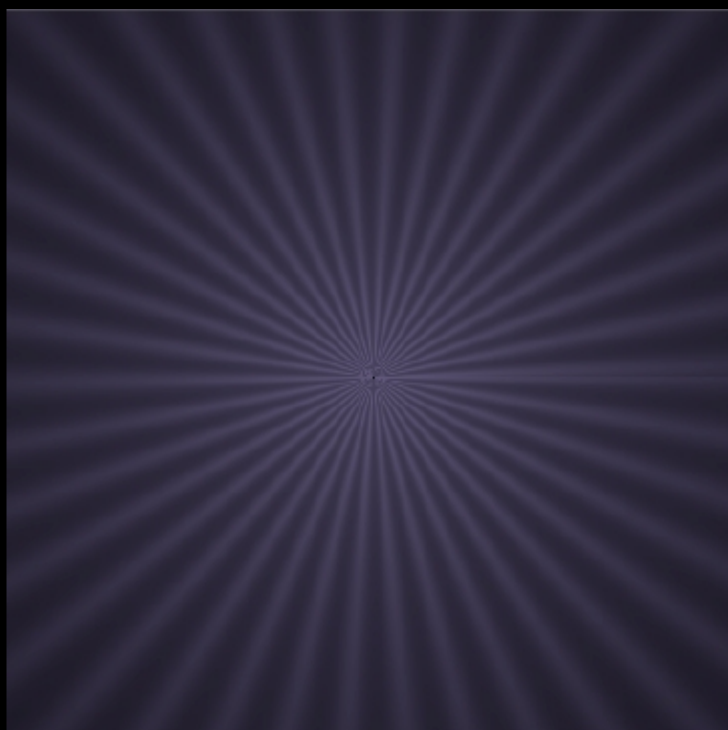
Big Sphere with Vertical Wave



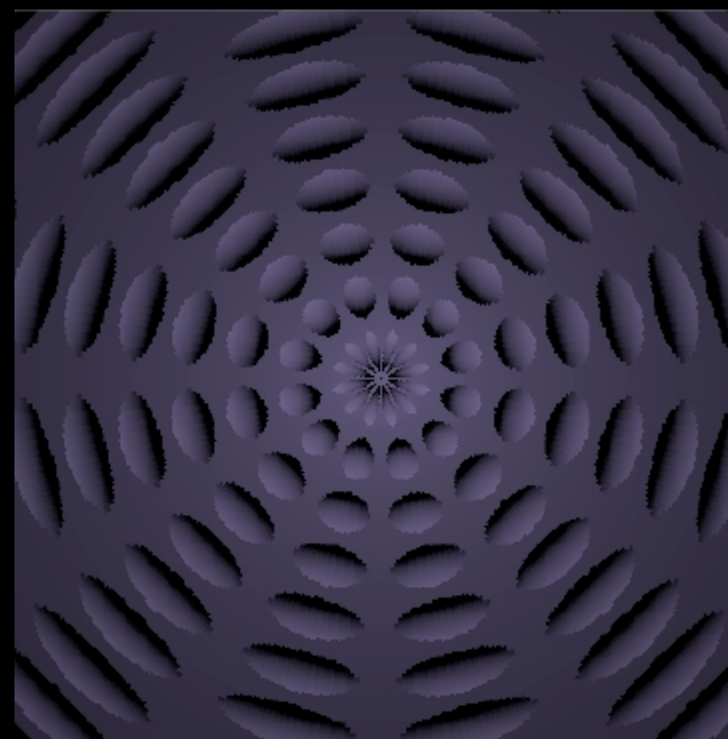
Big Sphere with Mesh



Cone Vertical with Wave

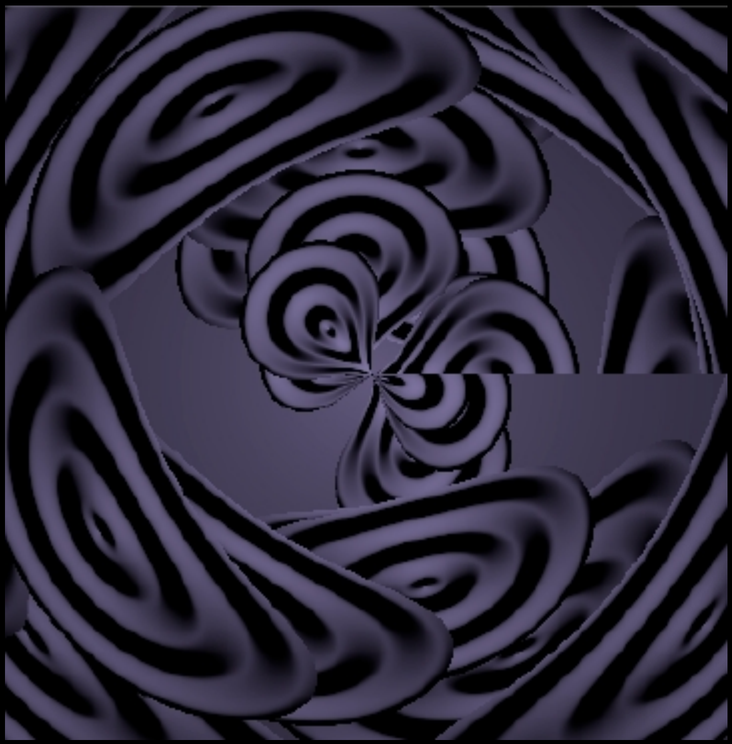


Cone with Dimples

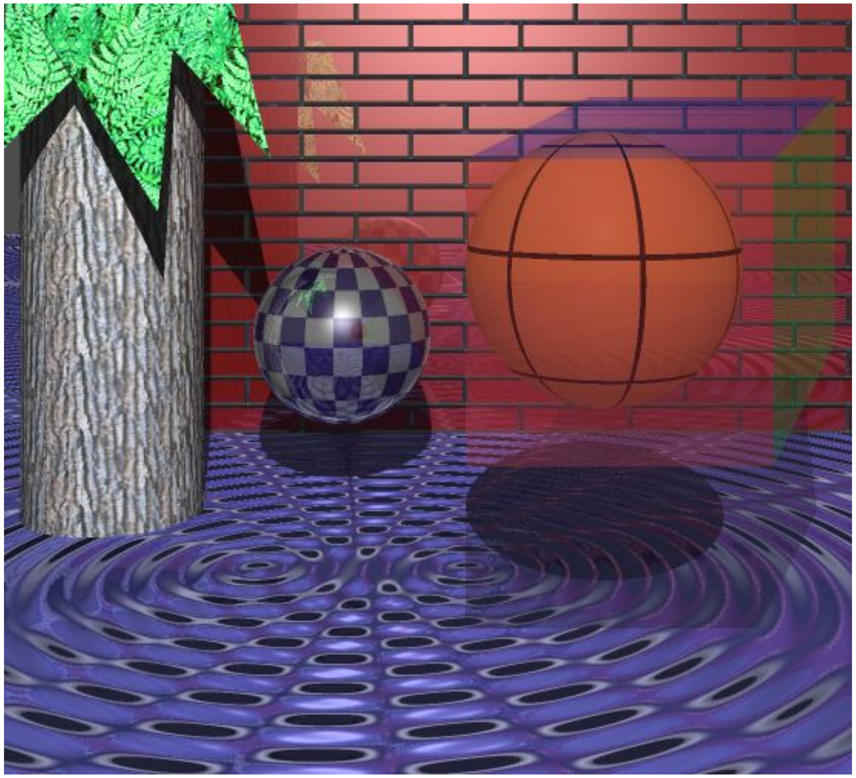


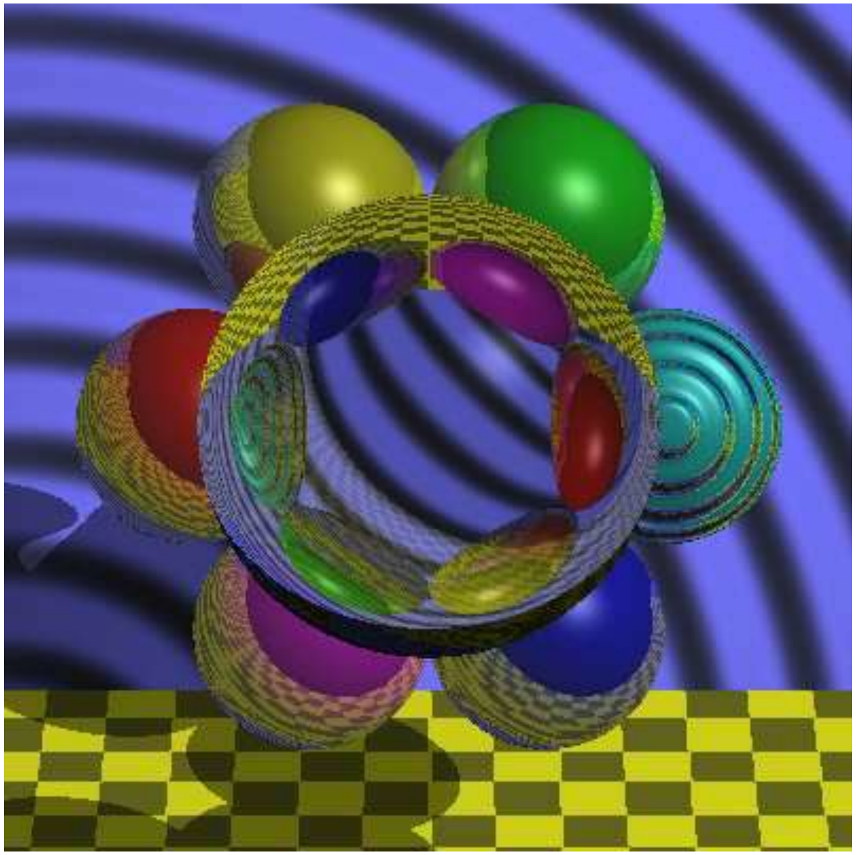
Cone with Ripple

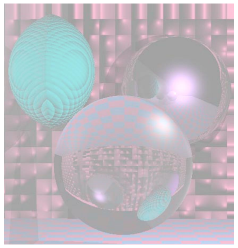
Cone with Ripples



Student Images







Bump Map - Plane

```
x = h - 200;
y = v - 200;
z = 0;

        N.Set(0, 0, 1);
        Du.Set(-1, 0, 0);
        Dv.Set(0, 1, 0);
        uu = h;
        vv = v;
        zz = z;
```

Bump Map Code – Big Sphere

```
radius = 280.0;
z = sqrt(radius*radius - y*y - x*x);
N.Set(x, y, z);
N = Norm(N);

Du.Set(z, 0, -x);
Du = -1*Norm(Du);
Dv.Set(-x*y, x*x +z*z, -y*z);
Dv = -1*Norm(Dv);

vv = acos(y/radius)*360/pi;
uu = (pi/2 +atan(x/z))*360/pi;
zz = z;
```

Bump Map Code – Dimples

```
Bu = 0; Bv = 0;
iu = (int)uu % 30 - 15;
iv = (int)vv % 30 - 15;
r2 = 225.0 - (double)iu*iu - (double)iv*iv;
if (r2 > 100) {
        if (iu == 0) Bu = 0;
        else Bu = (iu)/sqrt(r2);
        if (iv == 0) Bv = 0;
        else Bv = (iv)/sqrt(r2);
}
```

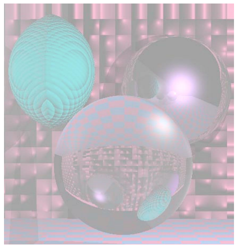


Image as a Bump Map

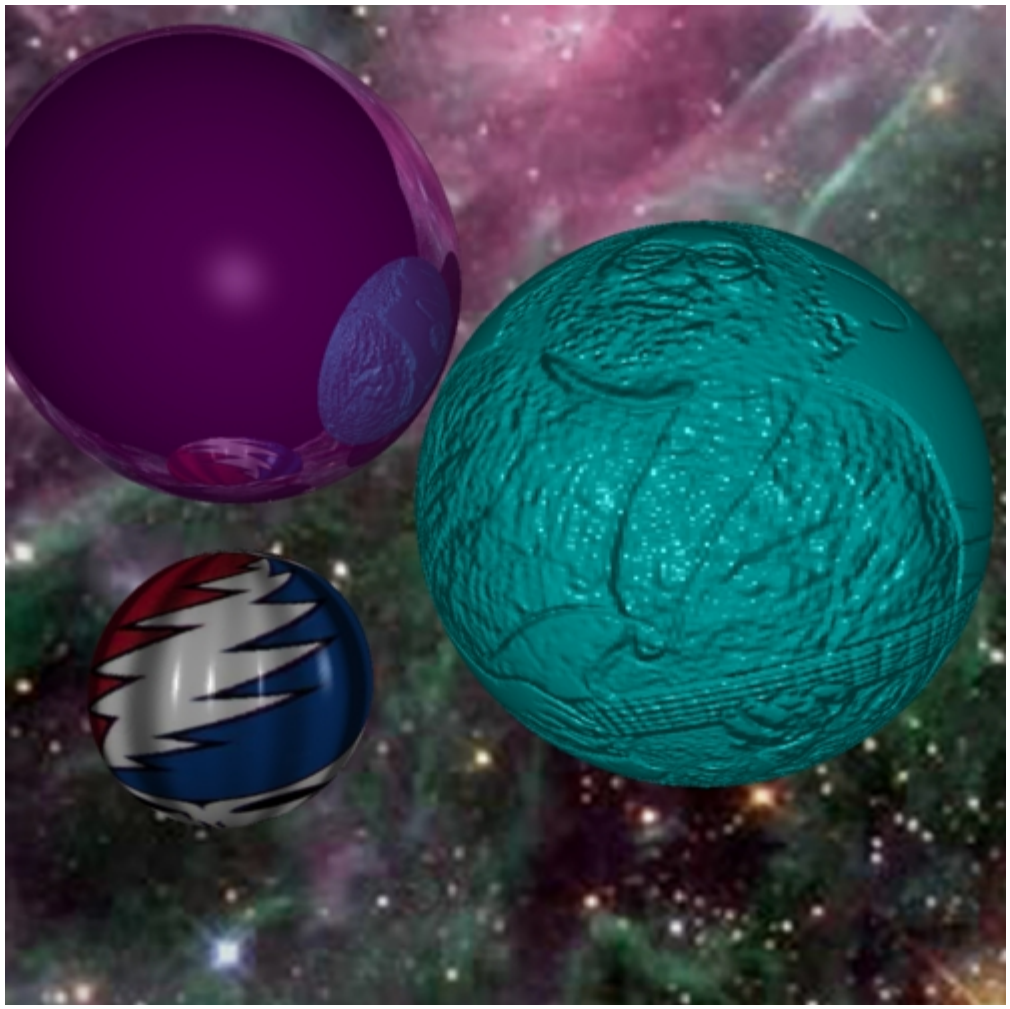
A bump map is a gray scale image; any image will do. The lighter areas are rendered as raised portions of the surface and darker areas are rendered as depressions. The bumping is sensitive to the direction of light sources.

http://www.cadcourse.com/winston/BumpMaps.html

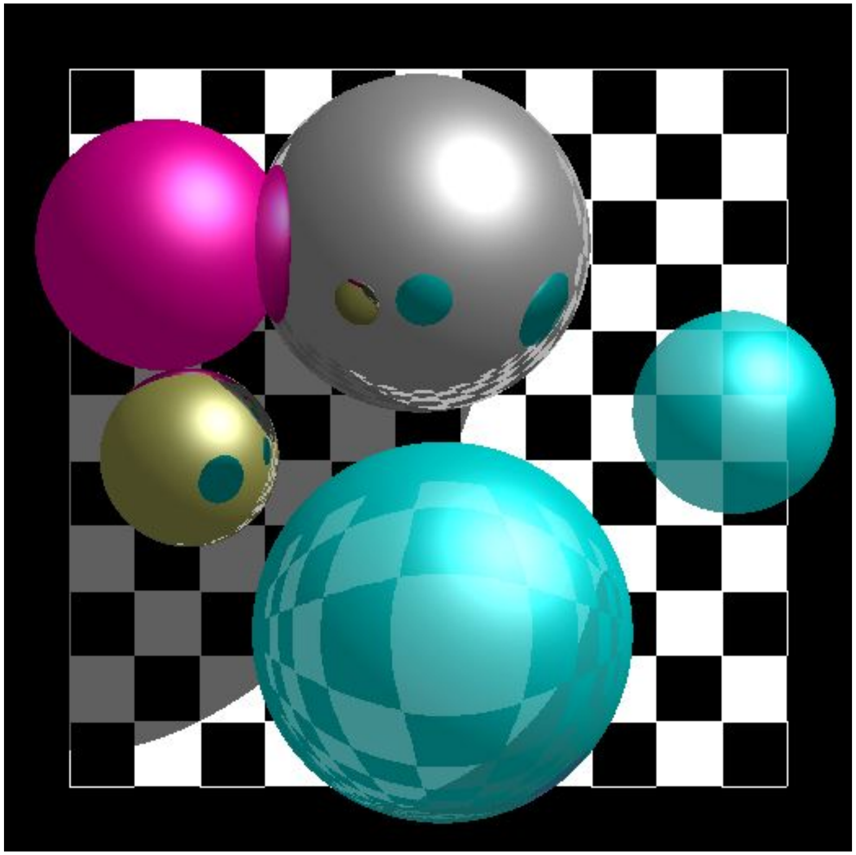
Time for a Break

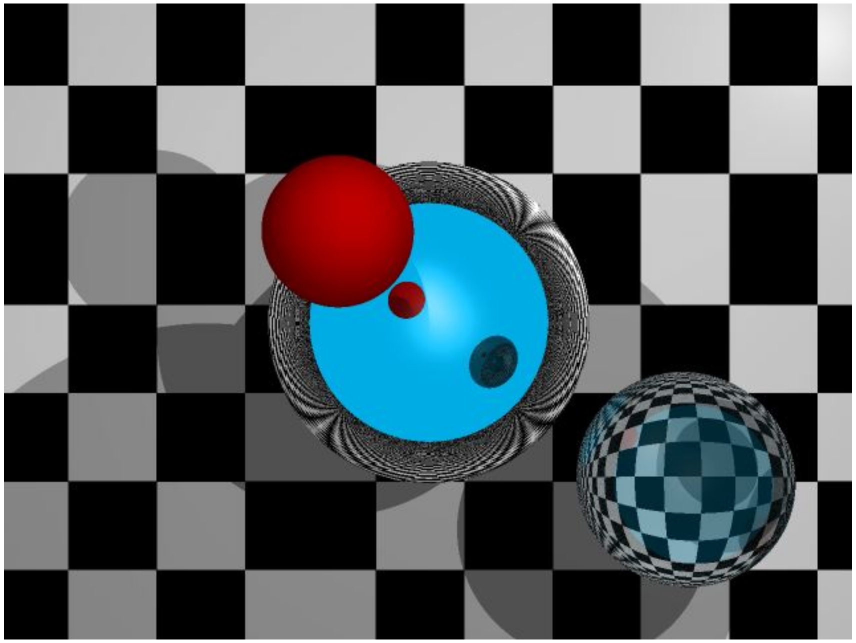
Bump Map from an Image

Victor Ortenberg

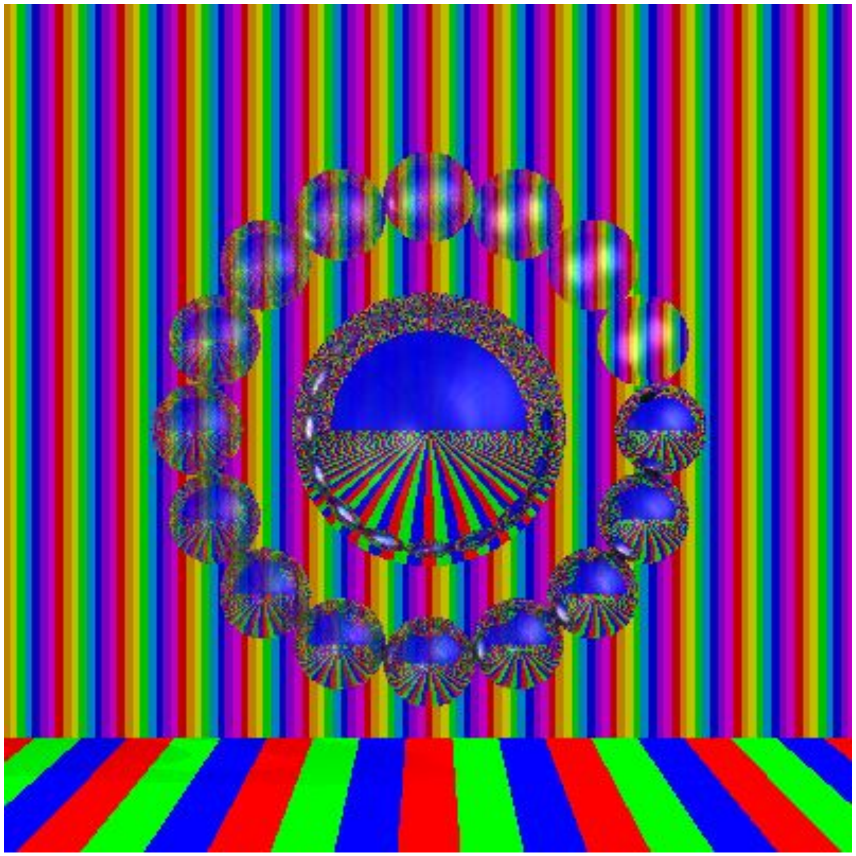


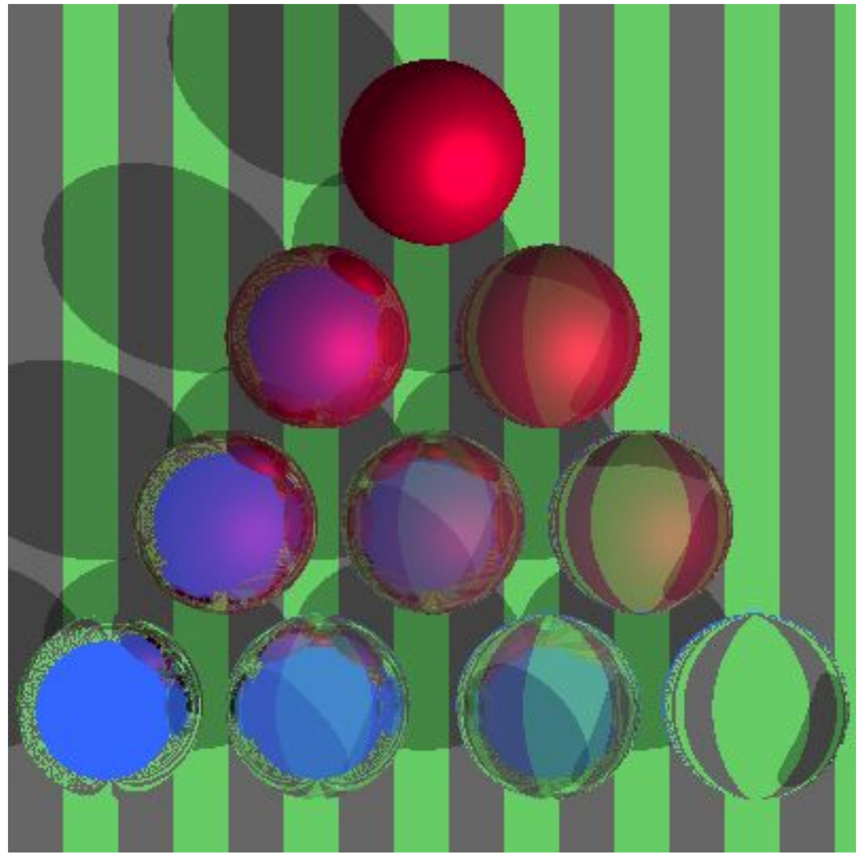
Simple Textures on Planes Parallel to Coordinate Planes



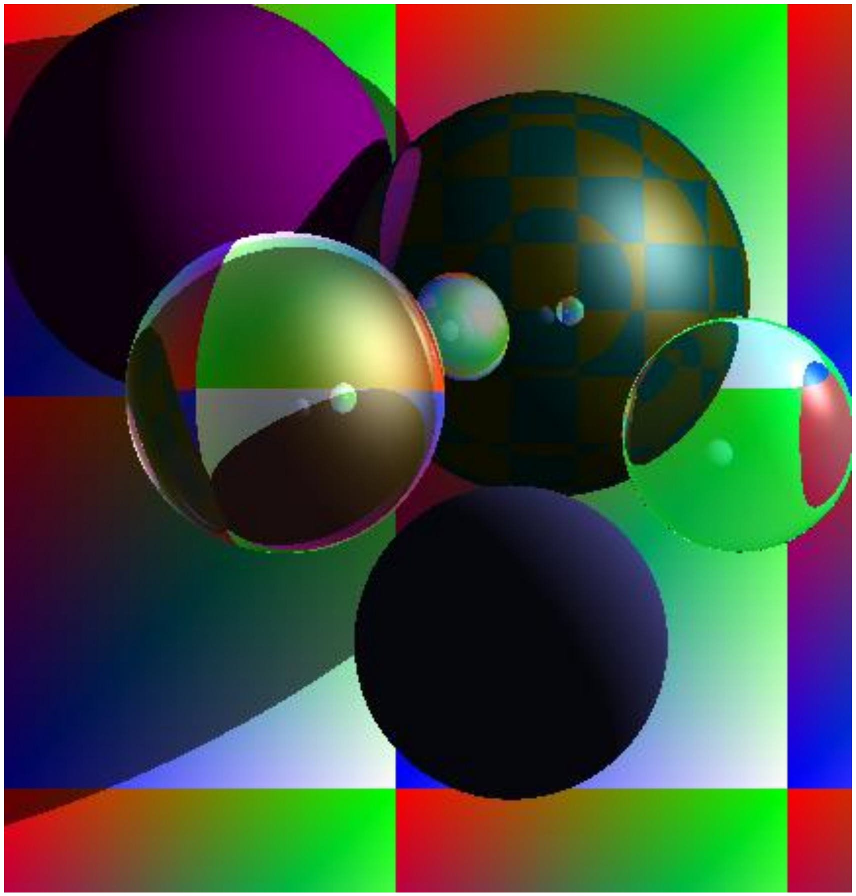


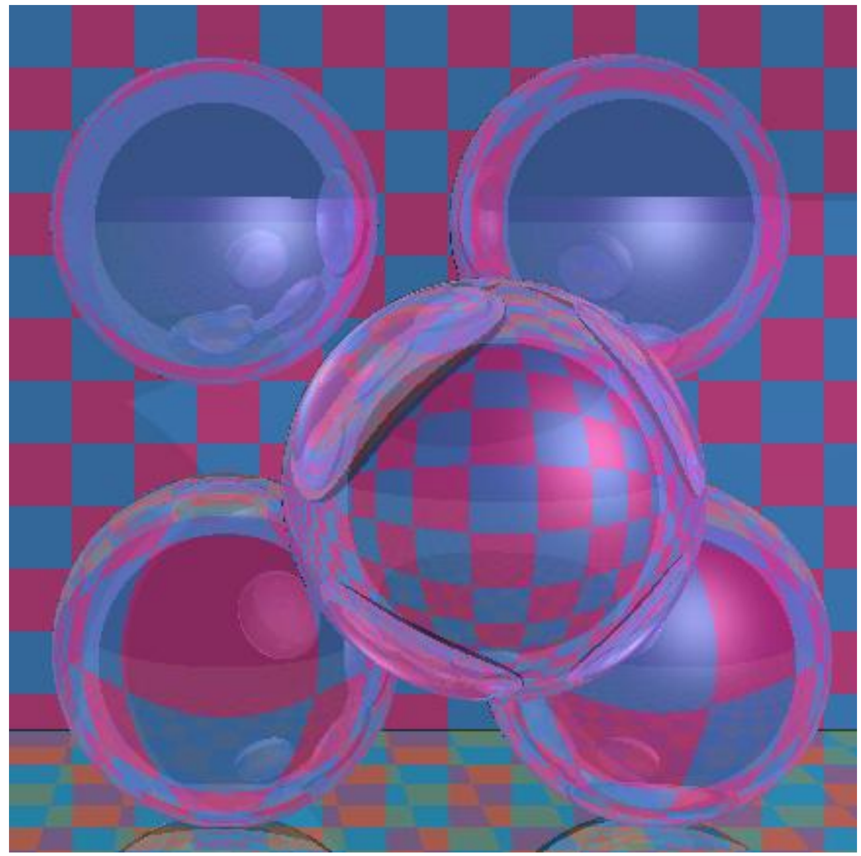
Stripes





Checks



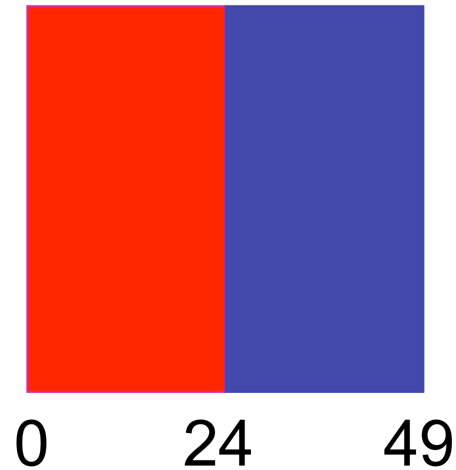


Stripes and Checks

Red and Blue Stripes

if ((x % 50) < 25) color = red

else color = blue



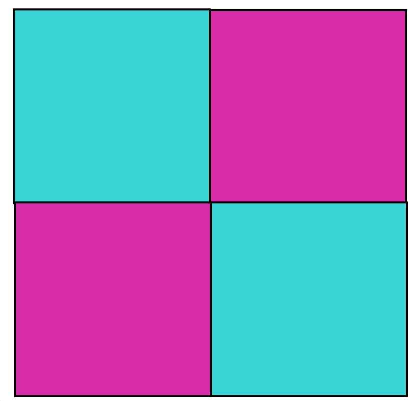
Cyan and Magenta Checks

if (((x % 50) < 25 && (y % 50) < 25)) ||

(((x % 50) >= 25 && (y % 50) >= 25)))

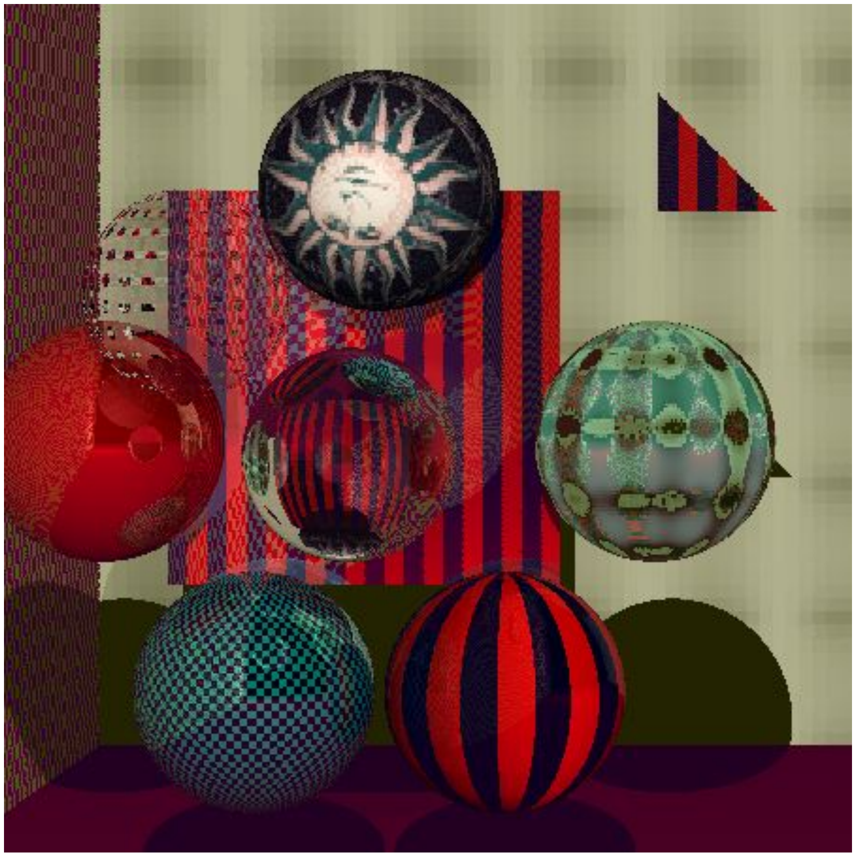
color = cyan

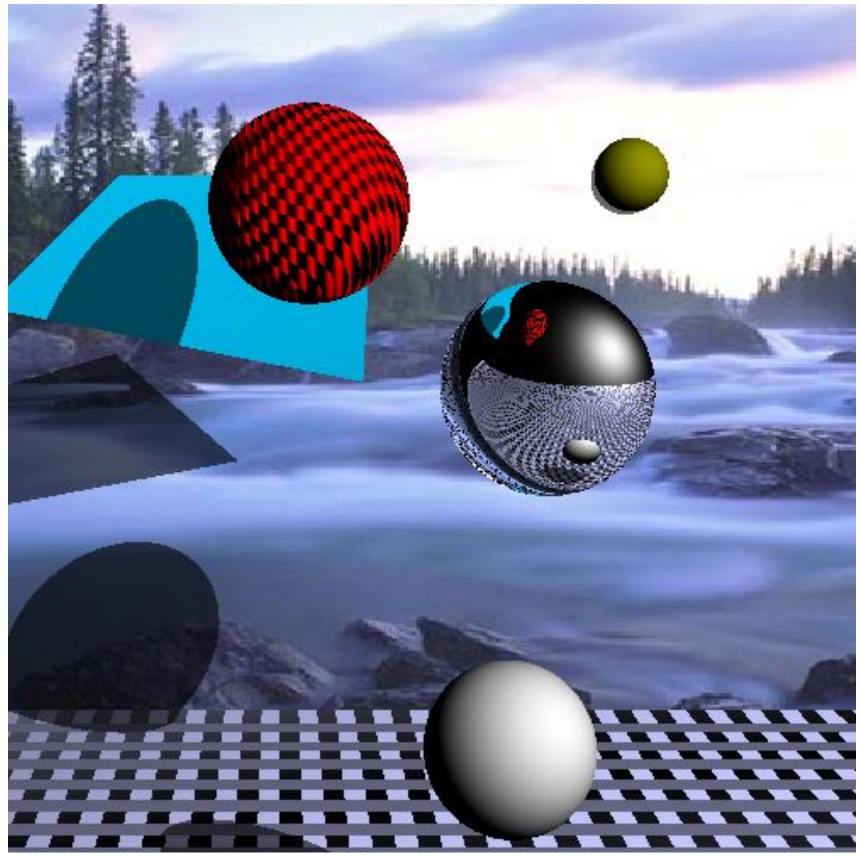
else color = magenta



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

Stripes, Checks, Image





Mona Scroll

Textures on 2 Planes

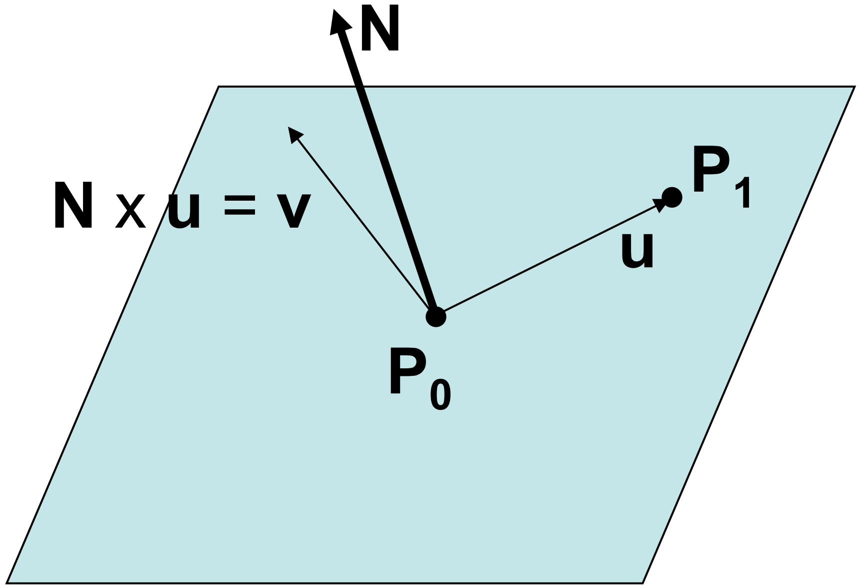
Mapping a Picture to a Plane

- 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.

v = **N** x **u**

Express **P** on the plane as

$\mathbf{P} = \mathbf{P_0} + a\mathbf{u} + b\mathbf{v}.$

Image to Triangle - 1

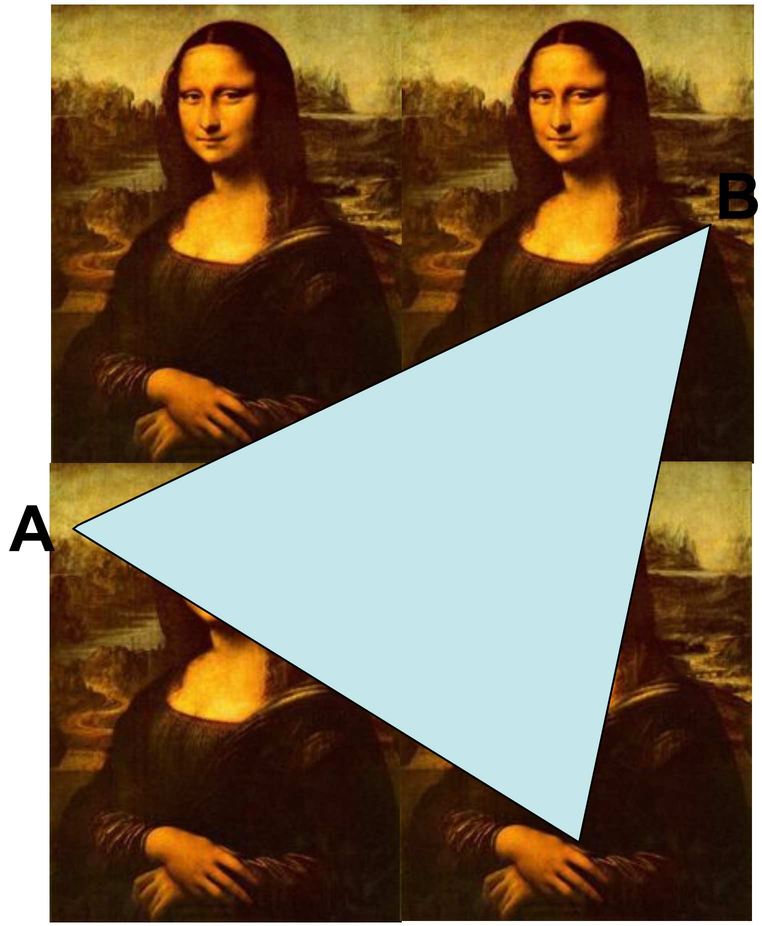


Image to Triangle - 2

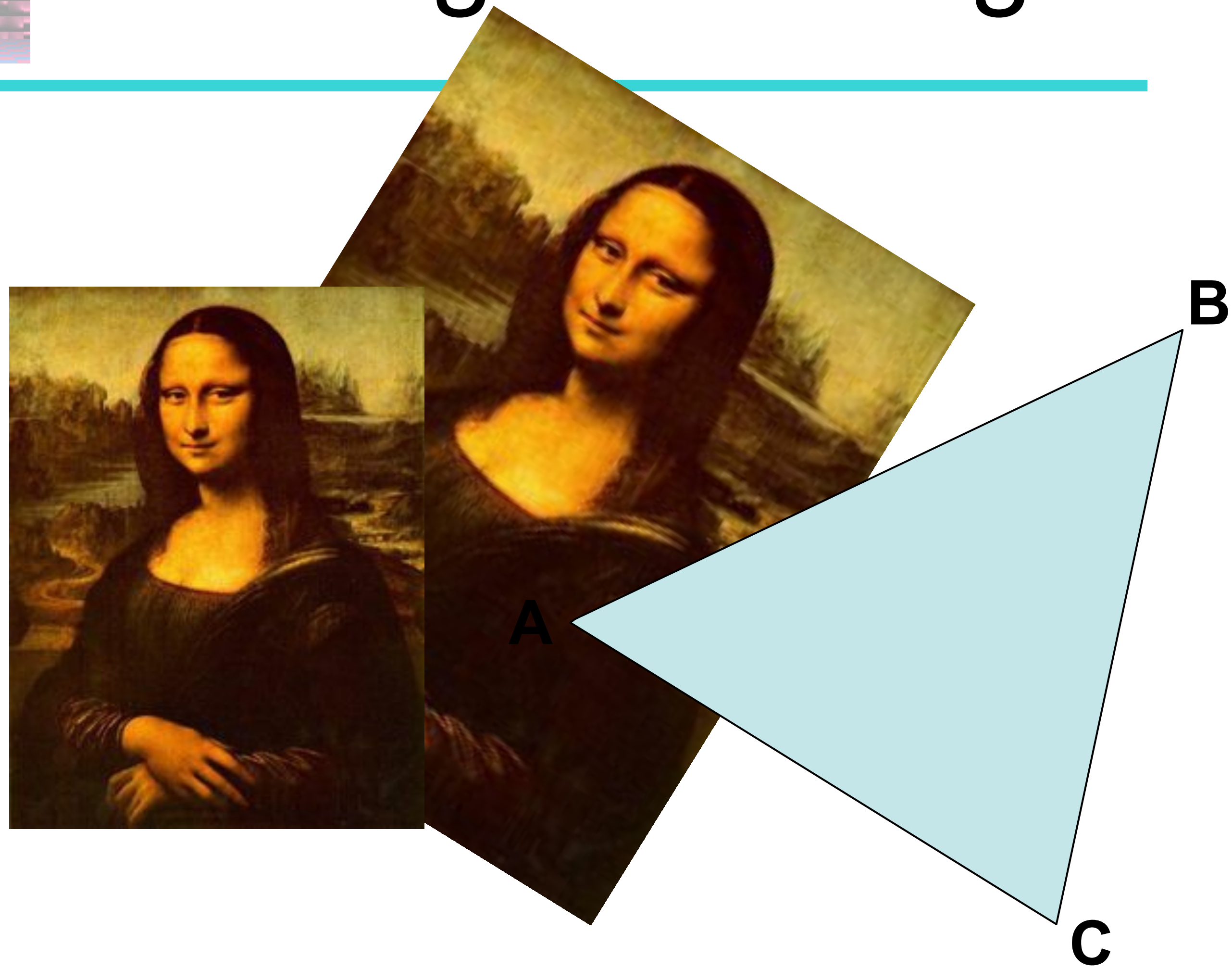
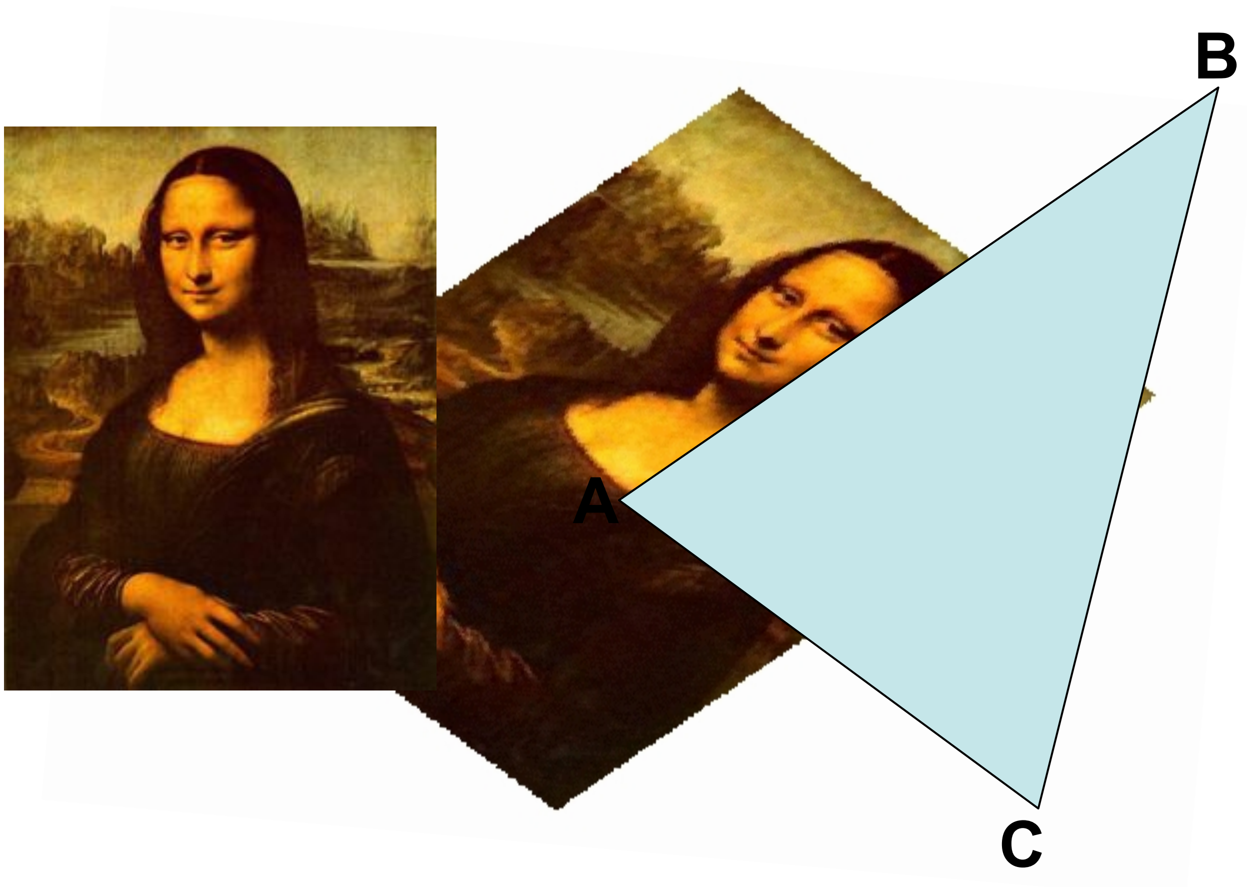
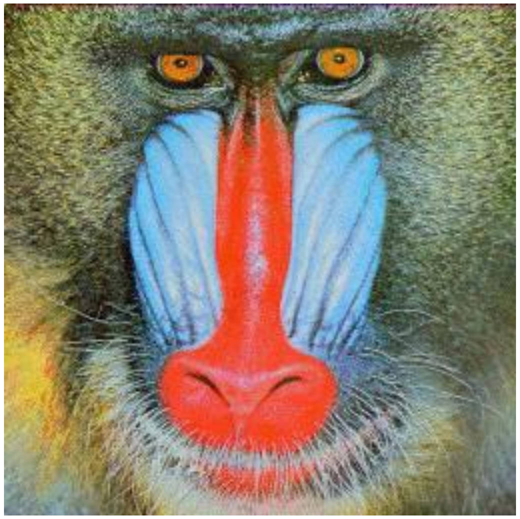


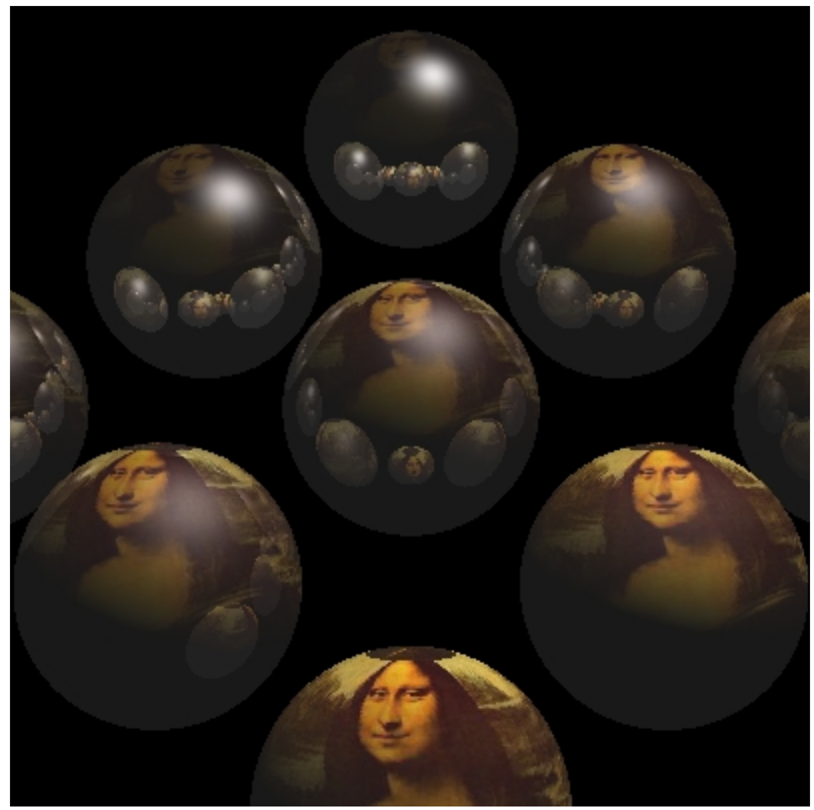
Image to Triangle - 3



Mandrill Sphere



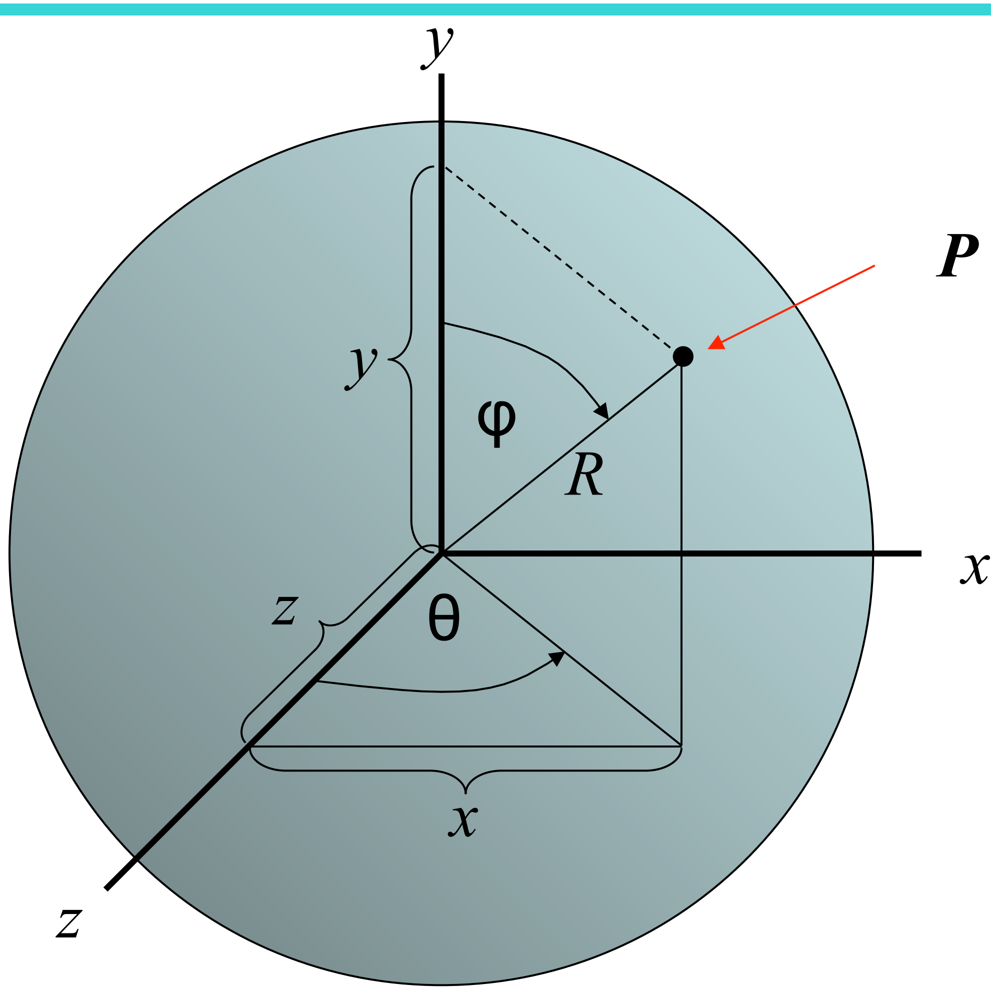
Mona Spheres



Tova Sphere

More Textured Spheres

Spherical Geometry



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

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

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

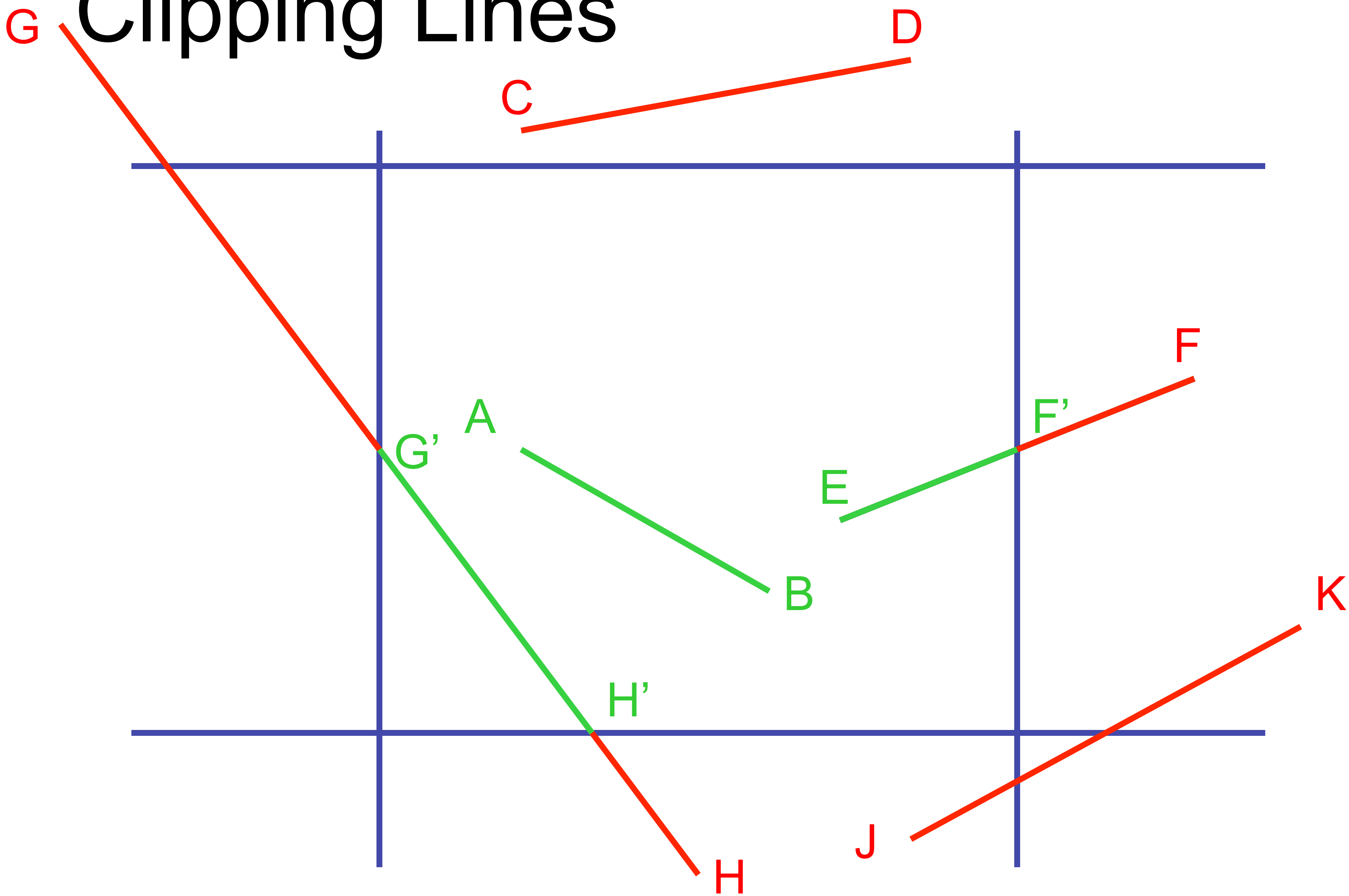
```
// for texture map – in lieu of using sphere color
double phi, theta;           // for spherical coordinates
double x, y, z;    // sphere vector coordinates
int h, v;              // ppm buffer coordinates
Vector3D V;

     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);
```

Clipping Lines



Intersections

We know how to find the intersections of a line segment

$$P + t(Q-P)$$

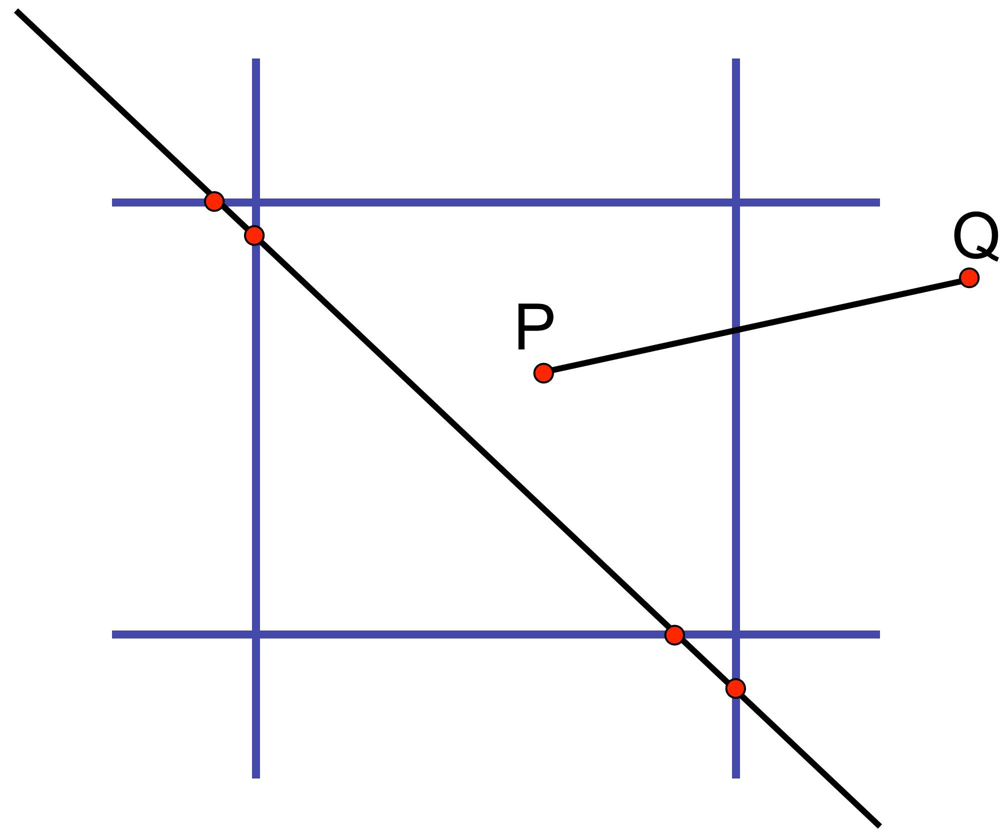
with the 4 boundaries

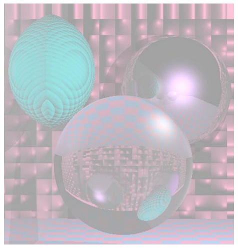
$$x = xmin$$

$$x = xmax$$

$$y = ymin$$

$$y = ymax$$





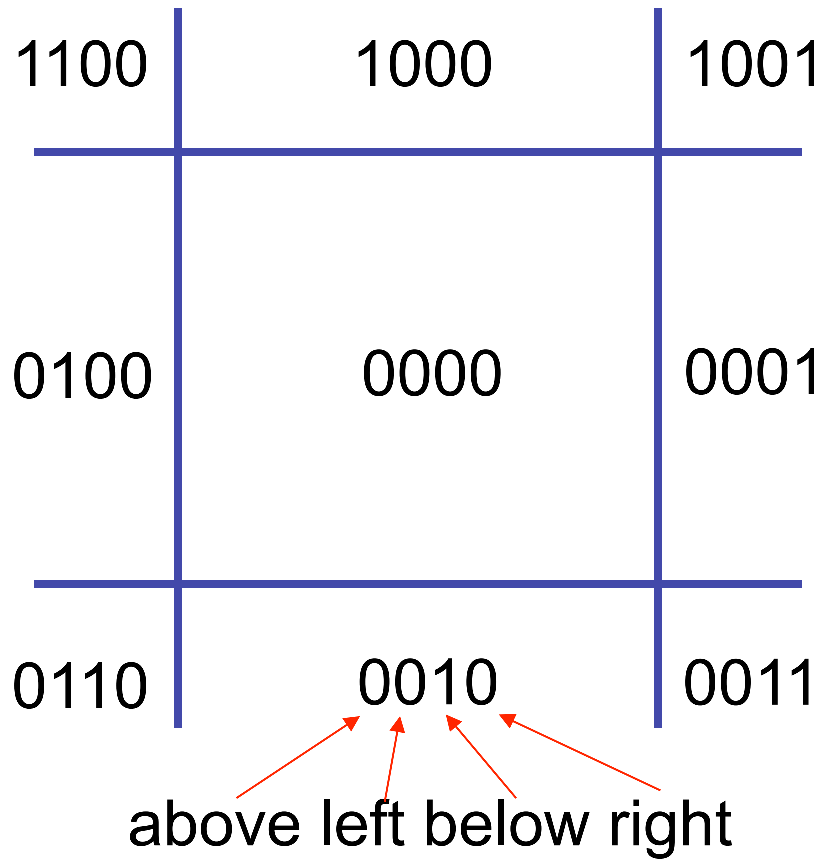
Cohen-Sutherland Clipping

1. Assign a 4 bit *outcode* to each endpoint.
2. Identify lines that are trivially accepted or trivially rejected.

if (outcode(P) = outcode(Q) = 0)
 accept

else if (outcode(P) & outcode (Q)) ≠ 0) reject

else test further



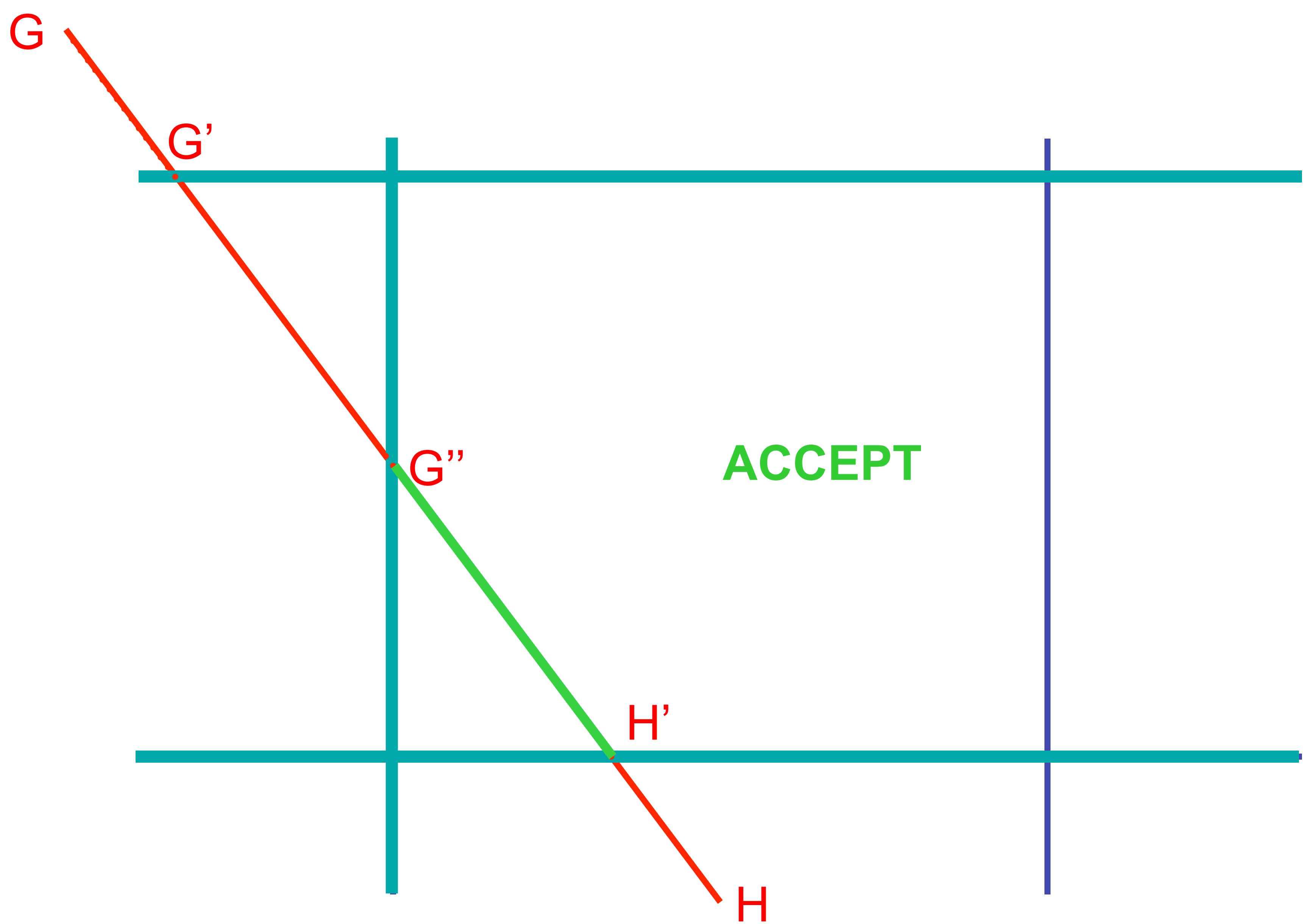
Cohen-Sutherland continued

Clip against one boundary at a time, **top**, left, bottom, right.

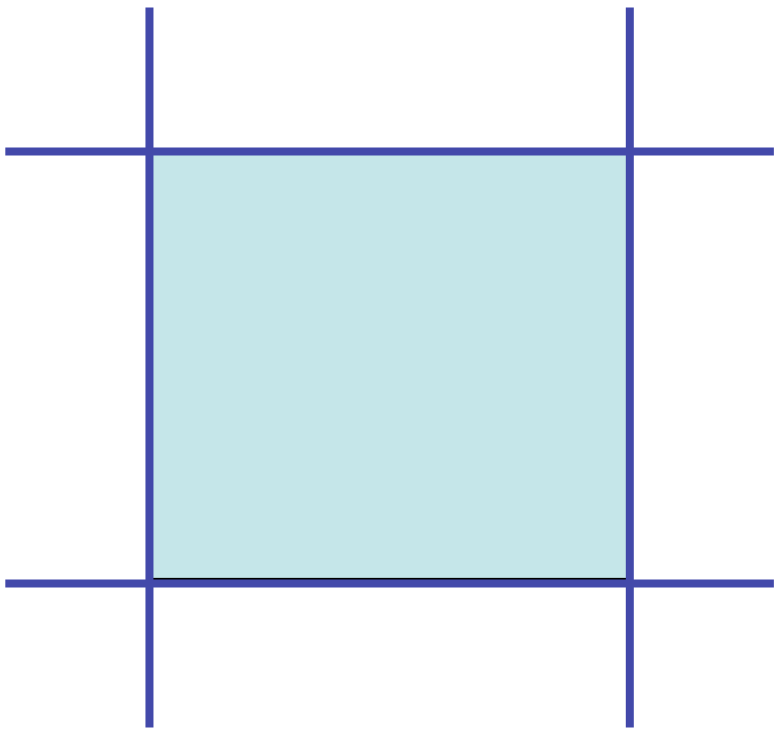
Check for trivial accept or reject.

If a line segment PQ falls into the “test further” category then

```
if (outcode(P) & 1000 ≠ 0)
        replace P with PQ intersect y = top
else if (outcode(Q) & 1000 ≠ 0)
        replace Q with PQ intersect y = top
go on to next boundary
```


G
G'
G''
ACCEPT
H'
H

Liang-Barsky Clipping

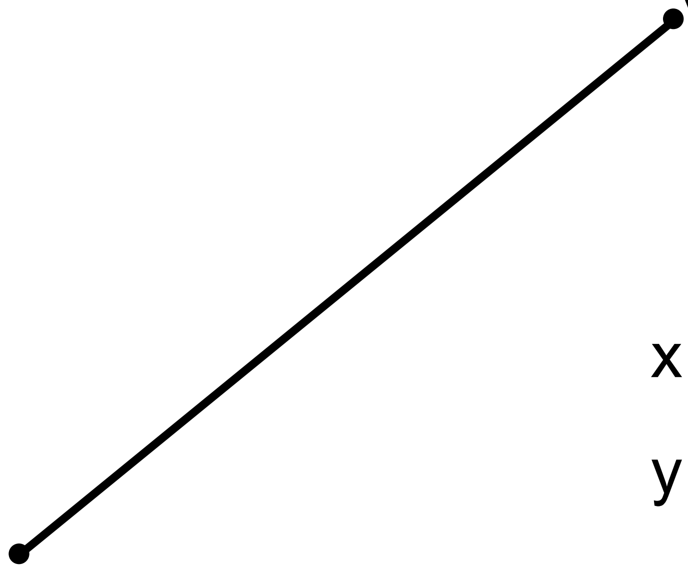


Clip window interior is defined by:

$$x_{left} \le x \le x_{right}$$

$$y_{bottom} \le y \le y_{top}$$

Liang-Barsky continued



$x = x_0 + t\Delta x \quad \Delta x = x_1 - x_0$

$y = y_0 + t\Delta y \quad \Delta y = y_1 - y_0$

$t = 0$ at $V_0 \qquad t = 1$ at V_1

Liang-Barsky continued

Put the parametric equations into the inequalities:

$xleft \le x_0 + t\Delta x \le xright$

$ybottom \le y_0 + t\Delta y \le ytop$

$-t\Delta x \le x_0 - xleft$ $\quad$ $t\Delta x \le xright - x_0$

$-t\Delta y \le y_0 - ybottom$ $\quad$ $t\Delta y \le ytop - y_0$

These decribe the interior of the clip window in terms of t.

Liang-Barsky continued

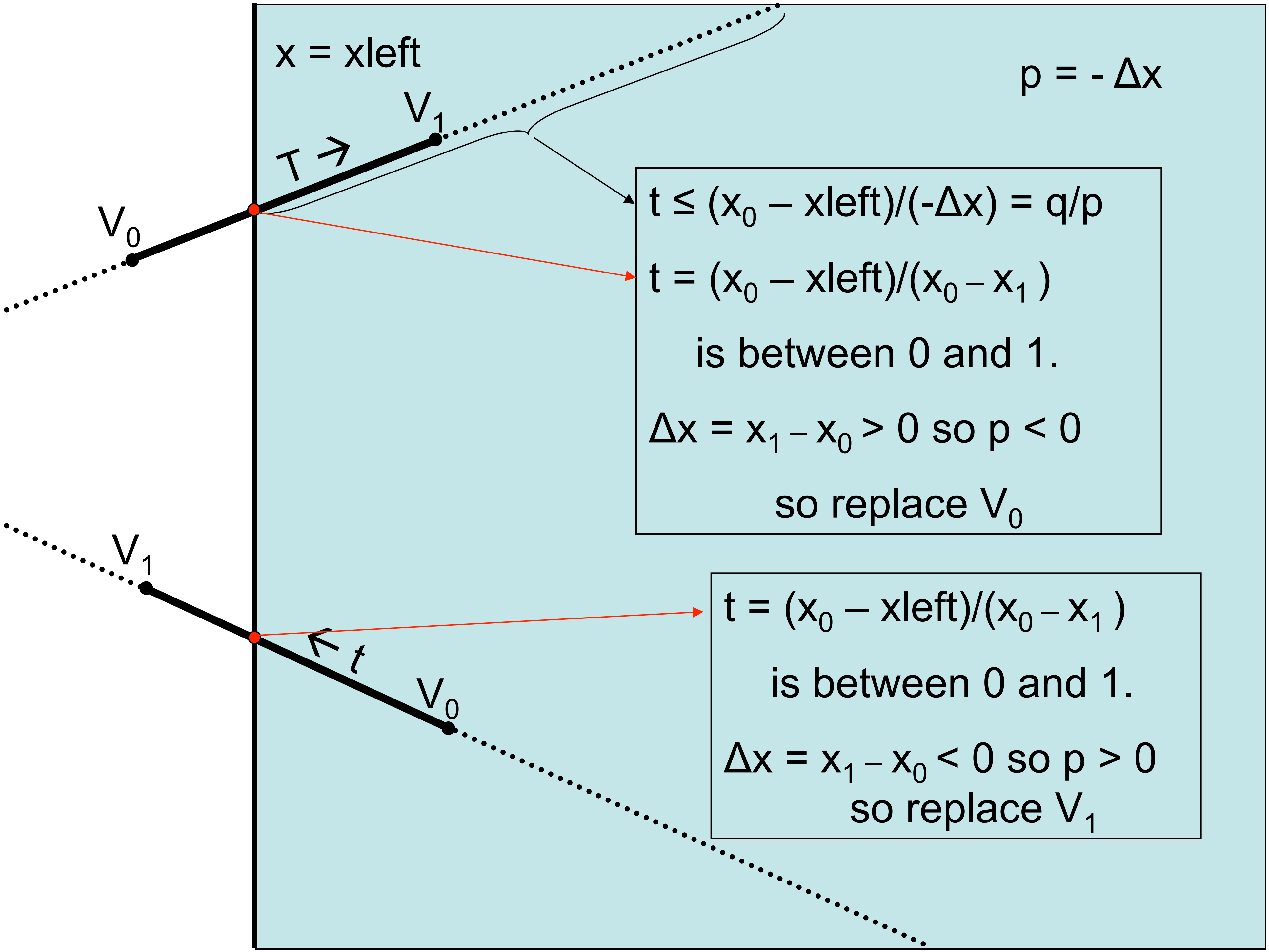
$-t\Delta x \le x_0 - xleft$ $\quad$ $t\Delta x \le xright - x_0$

$-t\Delta y \le y_0 - ybottom$ $\quad$ $t\Delta y \le ytop - y_0$

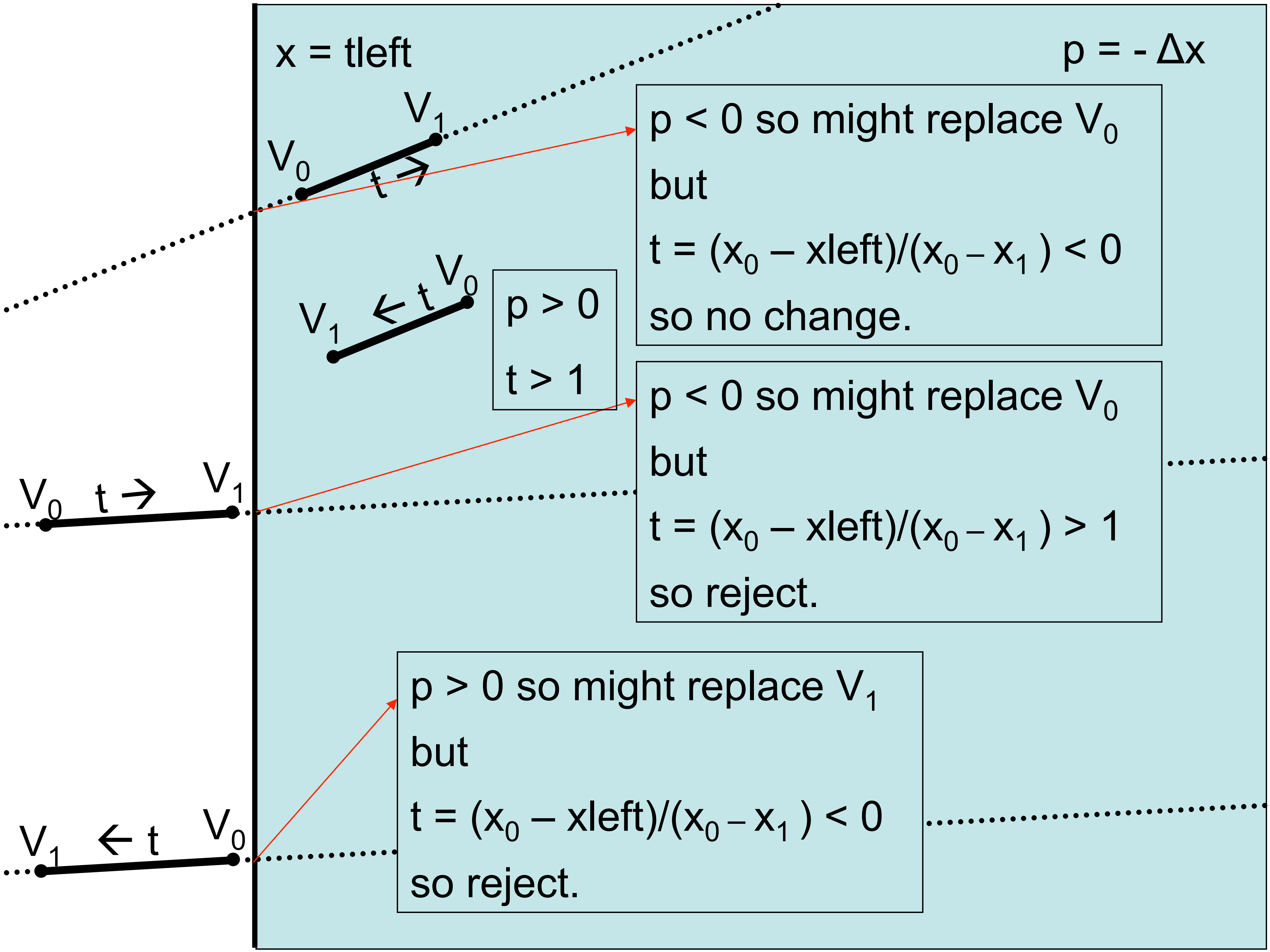
- These are all of the form

  $tp \le q$

- For each boundary, we decide whether to accept, reject, or which point to change depending on the sign of p and the value of t at the intersection of the line with the boundary.



x = xleft
p = - Δx
V1
T →
V0
t ≤ (x0 – xleft)/(-Δx) = q/p
t = (x0 – xleft)/(x0 – x1)
is between 0 and 1.
Δx = x1 – x0 > 0 so p < 0
so replace V0
V1
← t
V0
t = (x0 – xleft)/(x0 – x1)
is between 0 and 1.
Δx = x1 – x0 < 0 so p > 0
so replace V1



x = tleft
p = - Δx
V_0
V_1
t →
p < 0 so might replace V_0
but
t = $(x_0$ – xleft$)/(x_0 - x_1) < 0$
so no change.
V_1 ← t V_0
p > 0
t > 1
V_0 t → V_1
p < 0 so might replace V_0
but
t = $(x_0$ – xleft$)/(x_0 - x_1) > 1$
so reject.
V_1 ← t V_0
p > 0 so might replace V_1
but
t = $(x_0$ – xleft$)/(x_0 - x_1) < 0$
so reject.

Liang-Barsky Rules

- $0 < t < 1$, $p < 0$ replace V_0
- $0 < t < 1$, $p > 0$ replace V_1

- $t < 0$, $p < 0$ no change
- $t < 0$, $p > 0$ reject

- $t > 1$, $p > 0$ no change
- $t > 1$, $p < 0$ reject