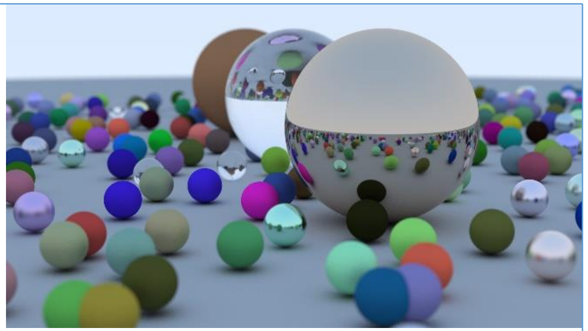
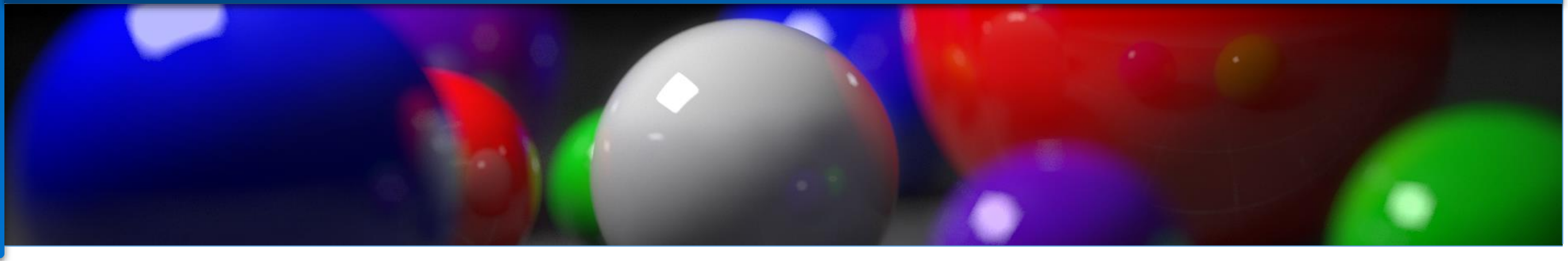
Comp4422



Computer Graphics

Lecture 11:

Visibility



Objectives

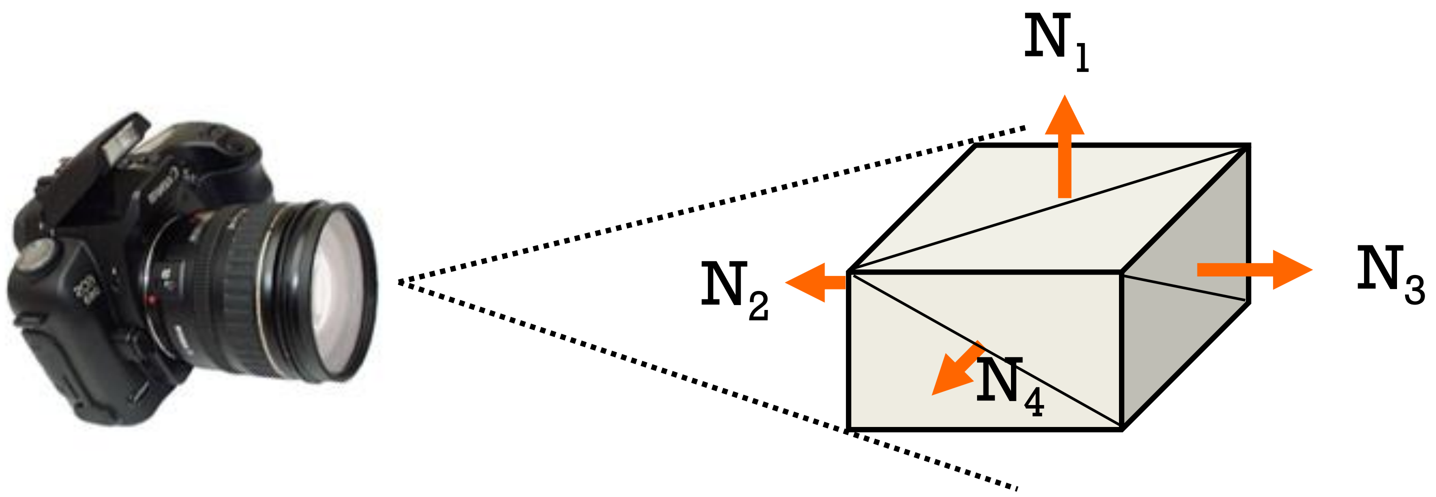
1. Learn about visible surfaces
2. Learn about the object space
3. Learn about the image space
4. Learn Z-Buffer Algorithm

1

Visibility Problem Definition

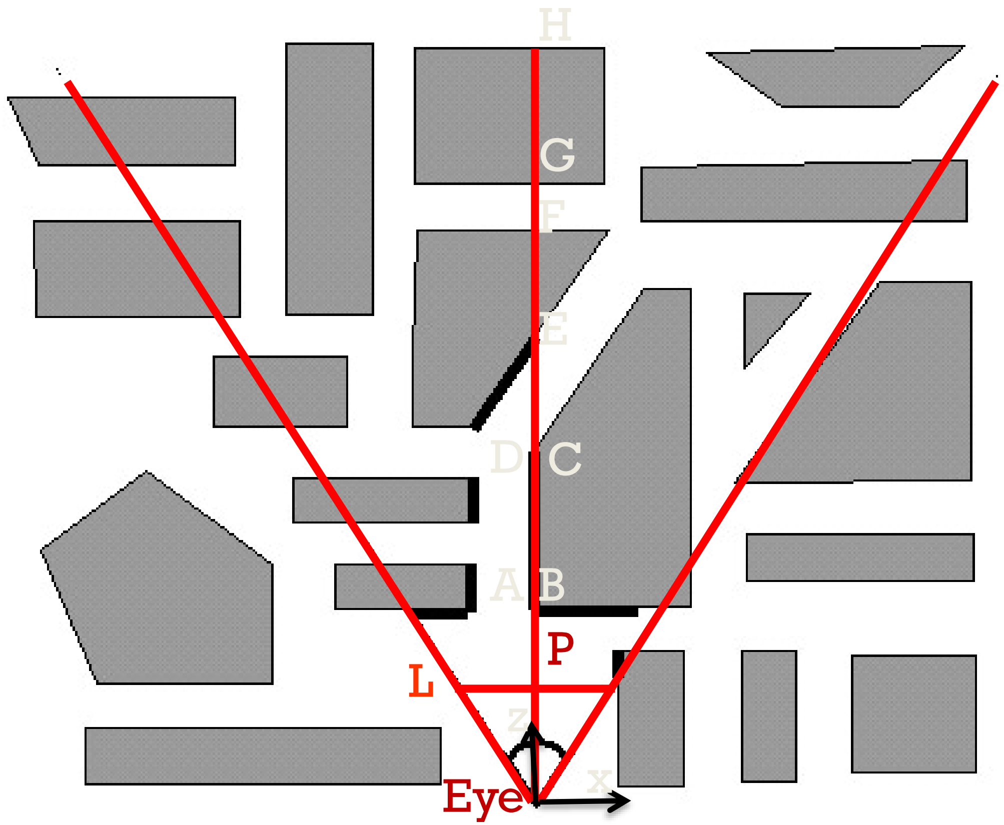
Part 1

Visibility Problem



Visible Surfaces and Culling

- Only what you see:



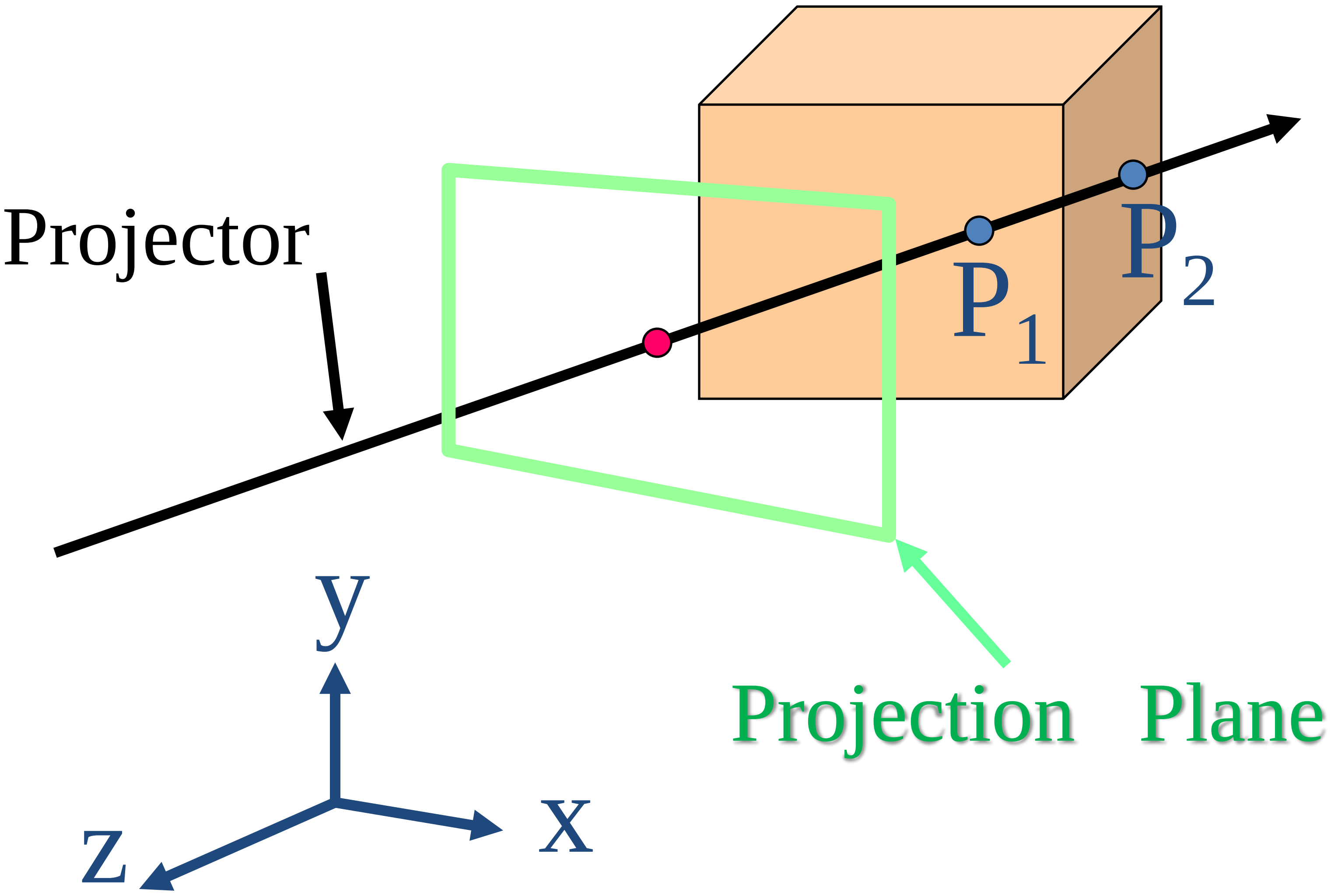
Motivation

- Too much to see = too many CPU cycles

Motivation (2)

- Too many surfaces = visibility cycles

Visibility Problem



Visibility Problem

- Given two points:

$$P_1 = [\ x_1,\ \ y_1,\ \ z_1\ ]$$
$$P_2 = [\ x_2,\ \ y_2,\ \ z_2\ ]$$

- Questions:

1. Are they on the same projector?

2. Which one is closer to the eye?

Visibility Questions

1. Are they on the same projector?

NO ⟶ Both points are visible

YES ⟶ Determine the depth

↓

2. Which one is closer to the eye?

Parallel Projection

- Points are on the same projector

if $x_1 = x_2$

and $y_1 = y_2$

Perspective Projection

- Points are on the same projector

if $x_1 / z_1 = x_2 / z_2$

and $y_1 / z_1 = y_2 / z_2$

Visibility Algorithms

- **Hidden Surfaces**
- **Object Space**
- **Image Space**

Hidden Surface Algorithms

- Classification:

1. Object Space
2. Image Space

2

Object Space Visibility

Part 2

Object Space

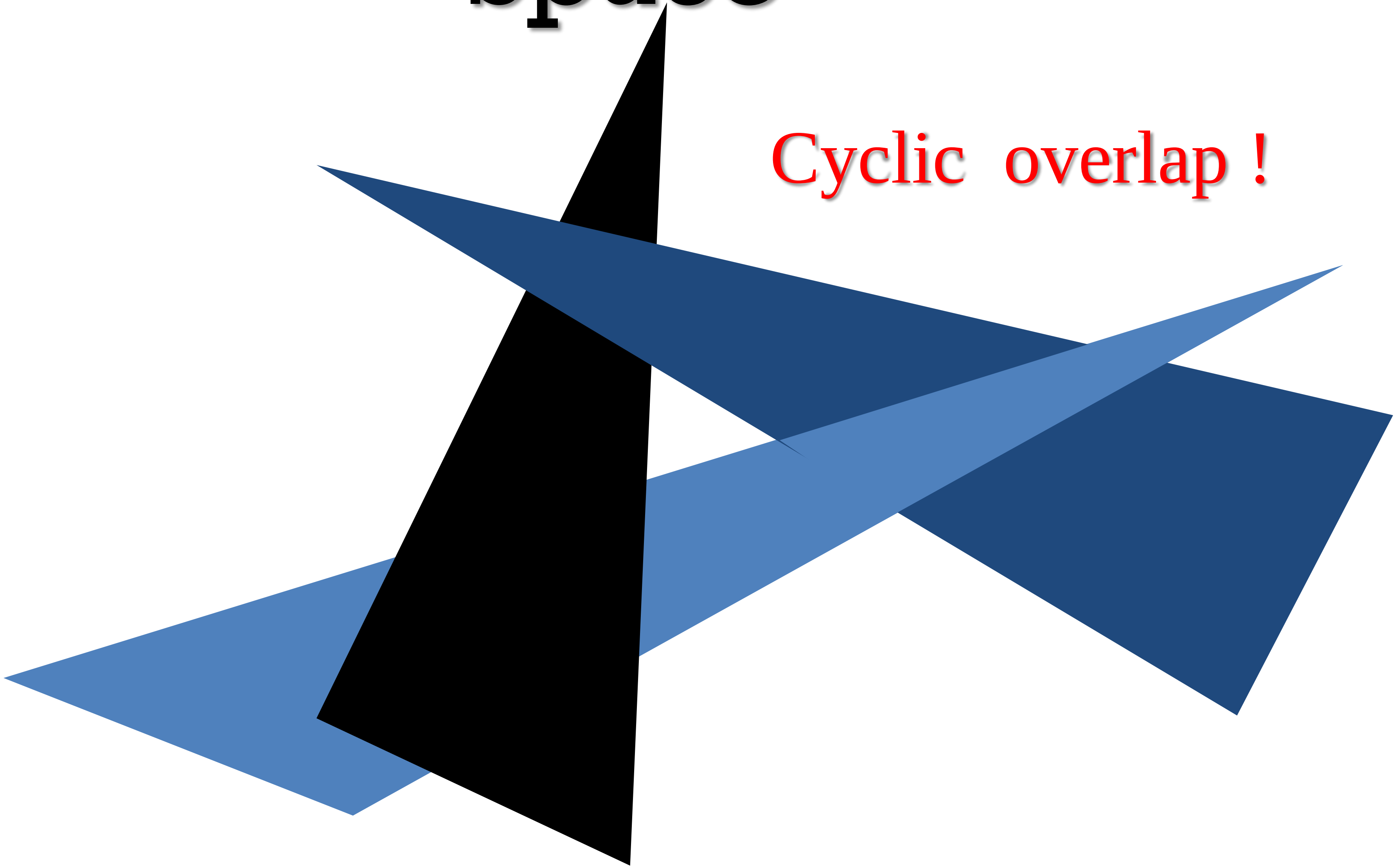
Object Space

1. Compare the depth of all objects
2. Eliminate entire invisible objects
3. Eliminate invisible parts

Algorithm:

```
for    each  object   do
          determine  the unobstructed  parts;
          draw            the unobstructed  parts;
done
```

Problems in Object Space



Cyclic overlap !

3

Image Space Visibility

Part 3

Image Space

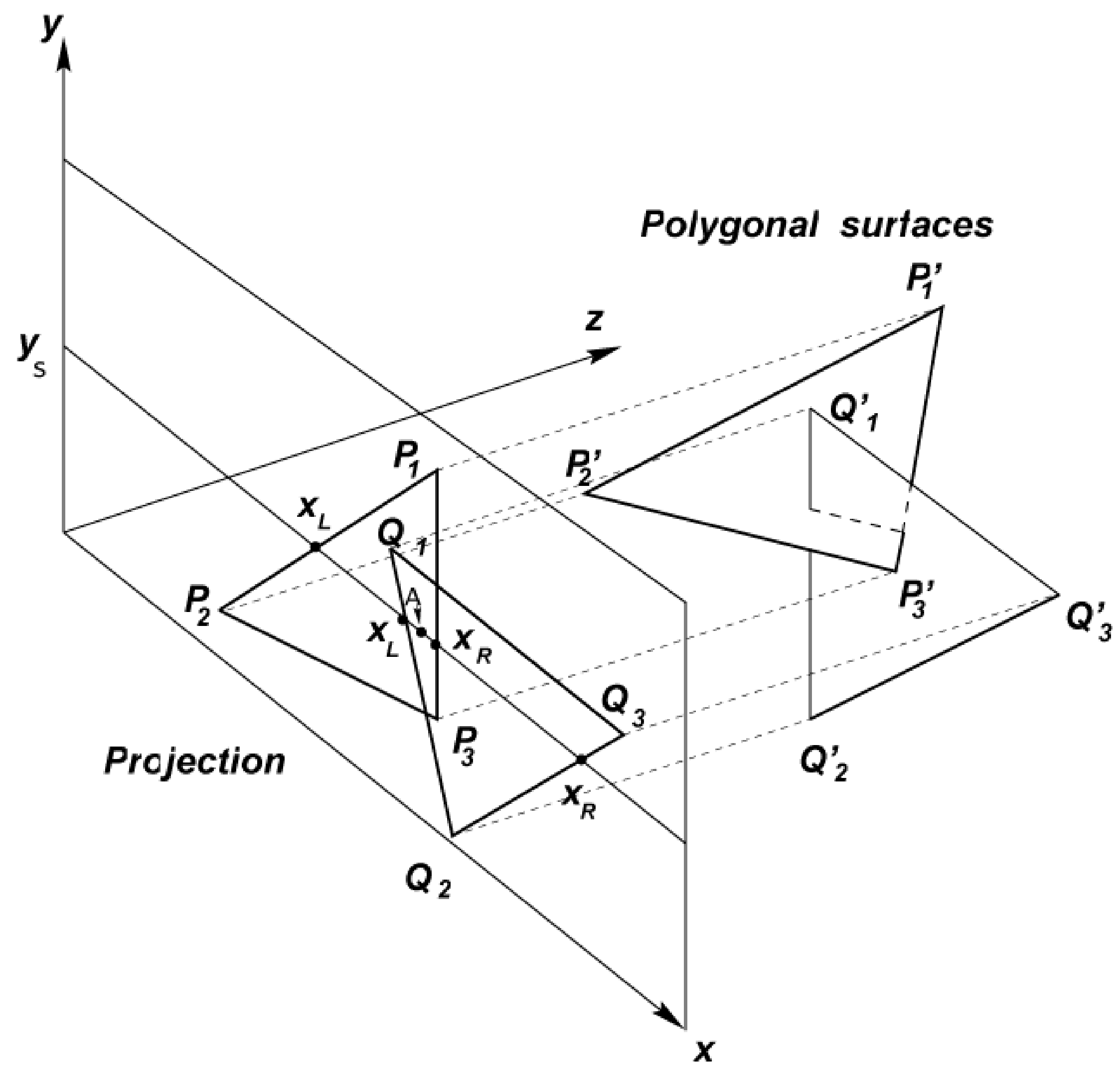
Image Space

1. At each pixel
2. Determine which object is visible

Algorithm:

```
for   each pixel   do
        determine  the object closest to the viewer;
        fill  the pixel with the color of the object;
done
```

Projection Scan



4 The Z-Buffer Algorithm

Part 4

The Z-Buffer Algorithm

Z-Buffer Algorithm Short Form

```
for    each  polygon   do
        for  each  pixel (x,y)  in  the polygon  do
                newZ :=  polygonZ(x,y);
                if newZ < zBuffer(x,y)
                    zBuffer(x,y) := newZ;
                    FrameBuffer(x,y) := polyColor(x,y);
                endif
        done
done
```

How do we compute Z(x,y)?

- A polygon is a planar surface
- Plane equation:

$$Ax + By + Cz + D = 0$$

$$z(x, y) = \frac{-D - Ax - By}{C}$$

$$z(x + \Delta x, y) = \frac{-D - A(x + \Delta x) - By}{C}$$

Depth Coherence

$$z(x+\Delta x, y) = \frac{-D - Ax - By}{C} - \frac{A\Delta x}{C}$$

$$z(x+\Delta x, y) = z(x, y) + K\Delta x$$

Part 5

Image Space Interpolation

Interpolations in Image Space

- After polygon projection and clipping:

Color information

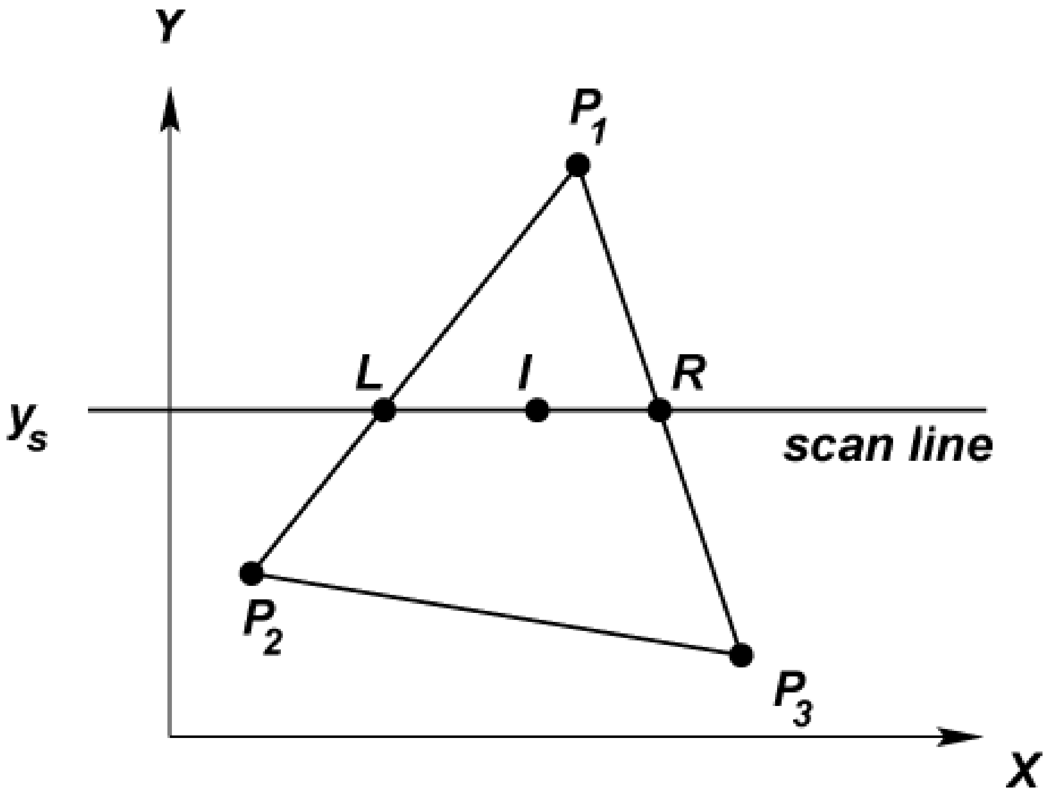
Frame buffer contains: c1 c2 c3

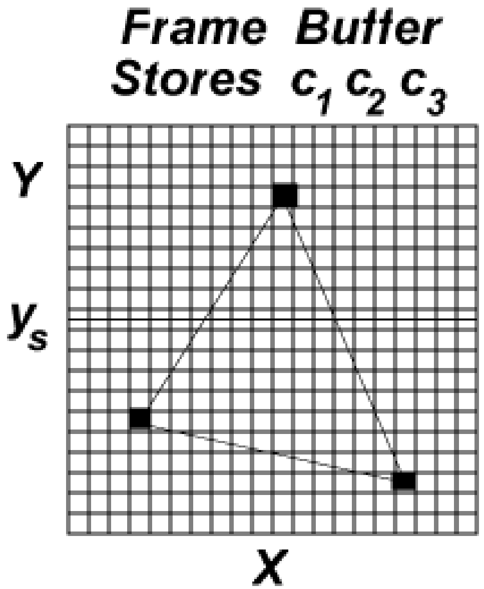
Depth information

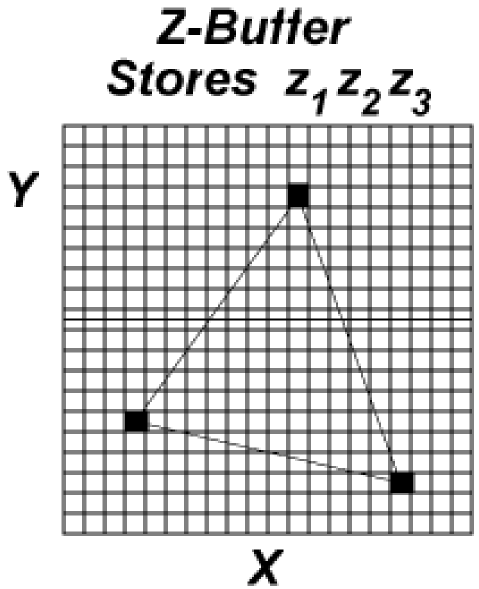
Z-buffer contains:

Interpolation

In Image Space







Interpolations

At scan-line yS:

$$\frac{xL - x1}{x2 - x1} = \frac{yS - y1}{y2 - y1}$$

$$\frac{xR - x1}{x3 - x1} = \frac{yS - y1}{y3 - y1}$$

Similarly

At scan-line yS:

$$\frac{zL - z1}{z2 - z1} = \frac{yS - y1}{y2 - y1}$$

$$\frac{zR - z1}{z3 - z1} = \frac{yS - y1}{y3 - y1}$$

Finally…

At scan-line yS:

$$\frac{zI - zL}{zR - zL} = \frac{xI - xL}{xR - xL}$$

Hence…

$$xL = x1 + \left(x2 - x1\right)\frac{yS - y1}{y2 - y1}$$

$$xR = x1 + \left(x3 - x1\right)\frac{yS - y1}{y3 - y1}$$

$$zL = z1 + \left(z2 - z1\right)\frac{yS - y1}{y2 - y1}$$

$$zR = z1 + \left(z3 - z1\right)\frac{yS - y1}{y3 - y1}$$

Therefore...

$$zI = zL + \left(zR - zL\right)\frac{xI - xL}{xR - xL}$$

Part 6

The Full Z-Buffer Algorithm

Z-Buffer Algorithm:

- Initialization:

```
for    each  pixel(x,y)   do
        zBuffer(x,y) := maxDepth;
done
```

Z-Buffer Algorithm:

```
for    each  polygon    do
       for   y := yMin   to   yMax   do
              construct  the  edgeList[y];
              for  each  segment in edgeList[y]  do
                     interpolate  xL,  xR;
                     interpolate  zL,  zR;
                     interpolate  cL,  cR;

                     Inner  Loop (next slide)
               done;
       done;
done;
```

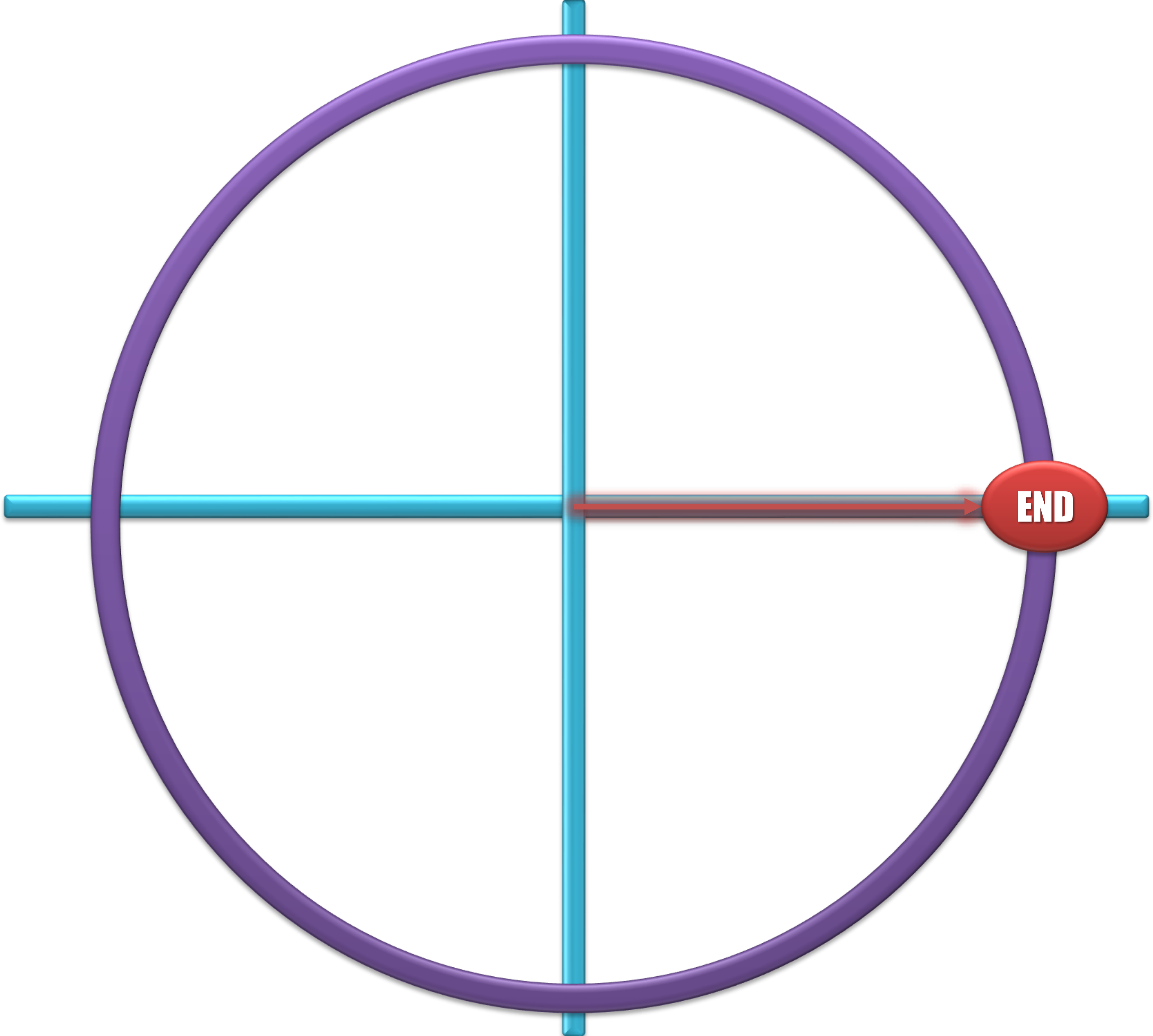
Z-Buffer Algorithm:

Inner Loop:

```
for    x := xL  to  xR   do
        z := interpolate(zL, zR);
        c := interpolate(cL, cR);
        if   z  <  zBuffer(x,y)   then
             zBuffer(x,y) := z;
             fBuffer(x,y) := c;
        endif
done
```

We learnt…

1. Visibility Modeling
2. Identify Visible Surfaces
3. Paint Surfaces On the Screen
4. The Z-Buffer Alg.

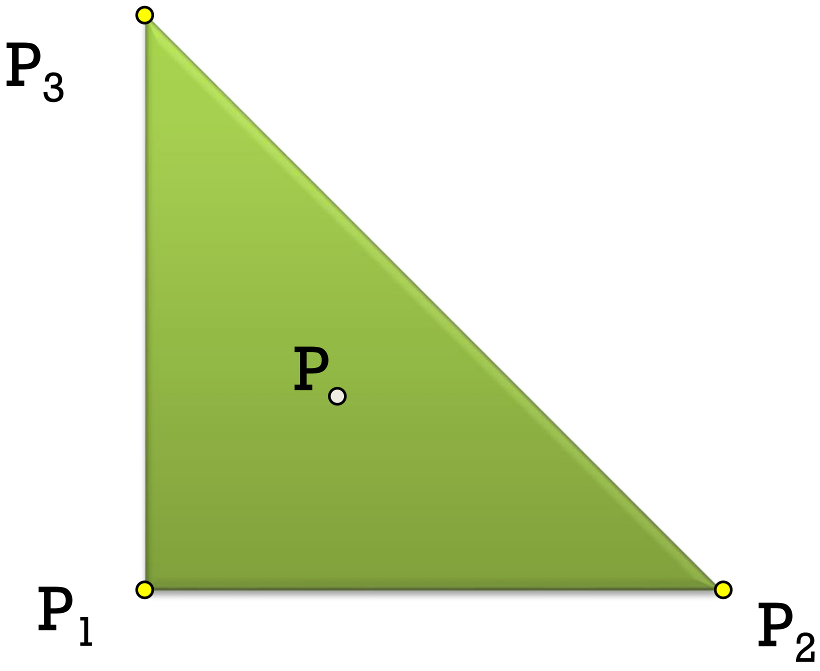

END

Recap

1. Convex Hulls
2. Barycentric Coordinates
3. Weighted Poins

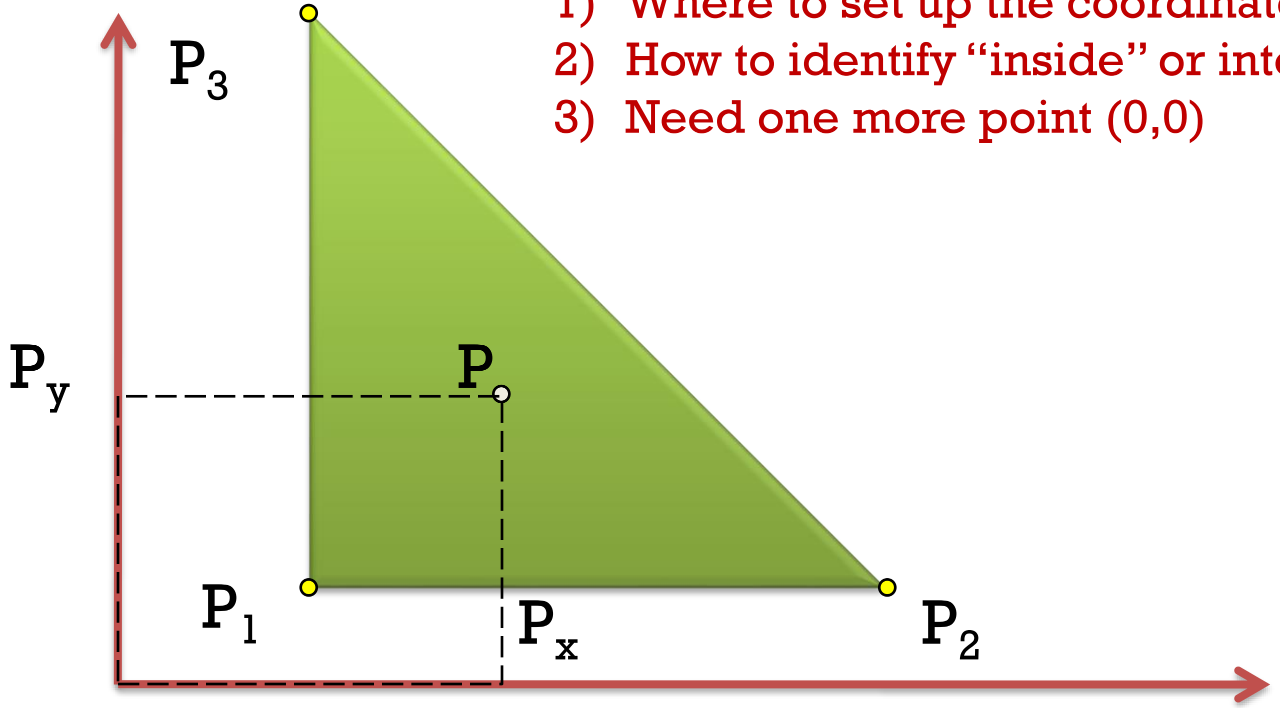
Problem

How to express the interior point of a polygon in terms of its vertices?



Solution 1

Use a coordinate system



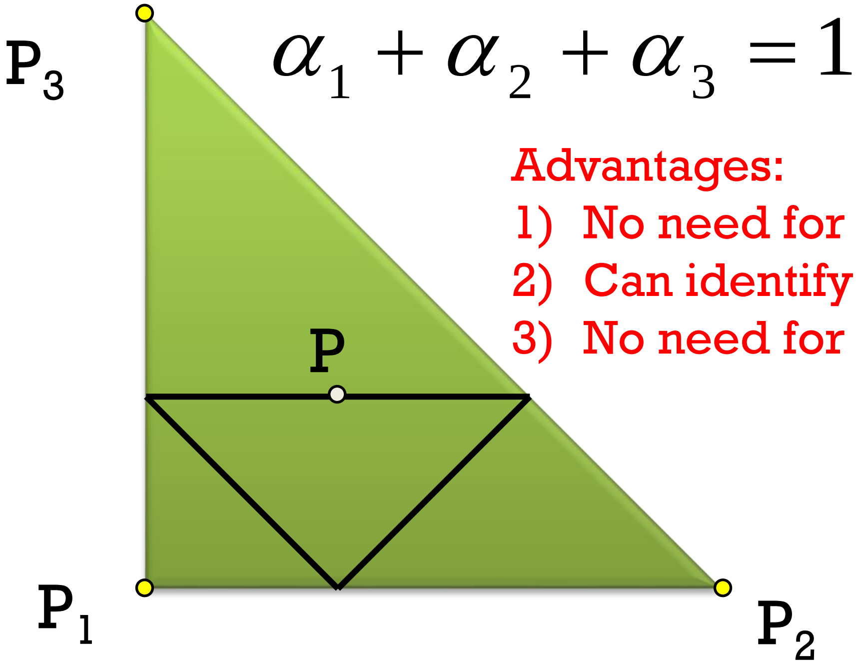
Problems:
1) Where to set up the coordinate system?
2) How to identify "inside" or internal points?
3) Need one more point (0,0)

Solution 2

Barycentric coordinate system

$$P = \alpha_1 P_1 + \alpha_2 P_2 + \alpha_3 P_3$$

$$\alpha_1 + \alpha_2 + \alpha_3 = 1$$

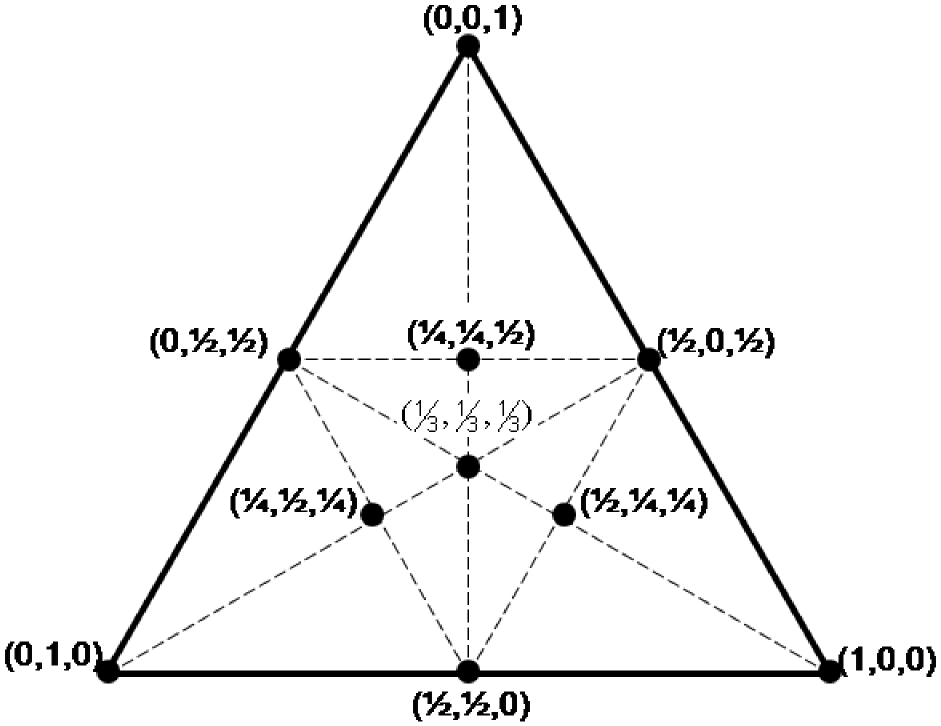


Advantages:

1) No need for a coordinate system
2) Can identify "inside" or internal points
3) No need for one more point (0,0)

Barycentric Coord.

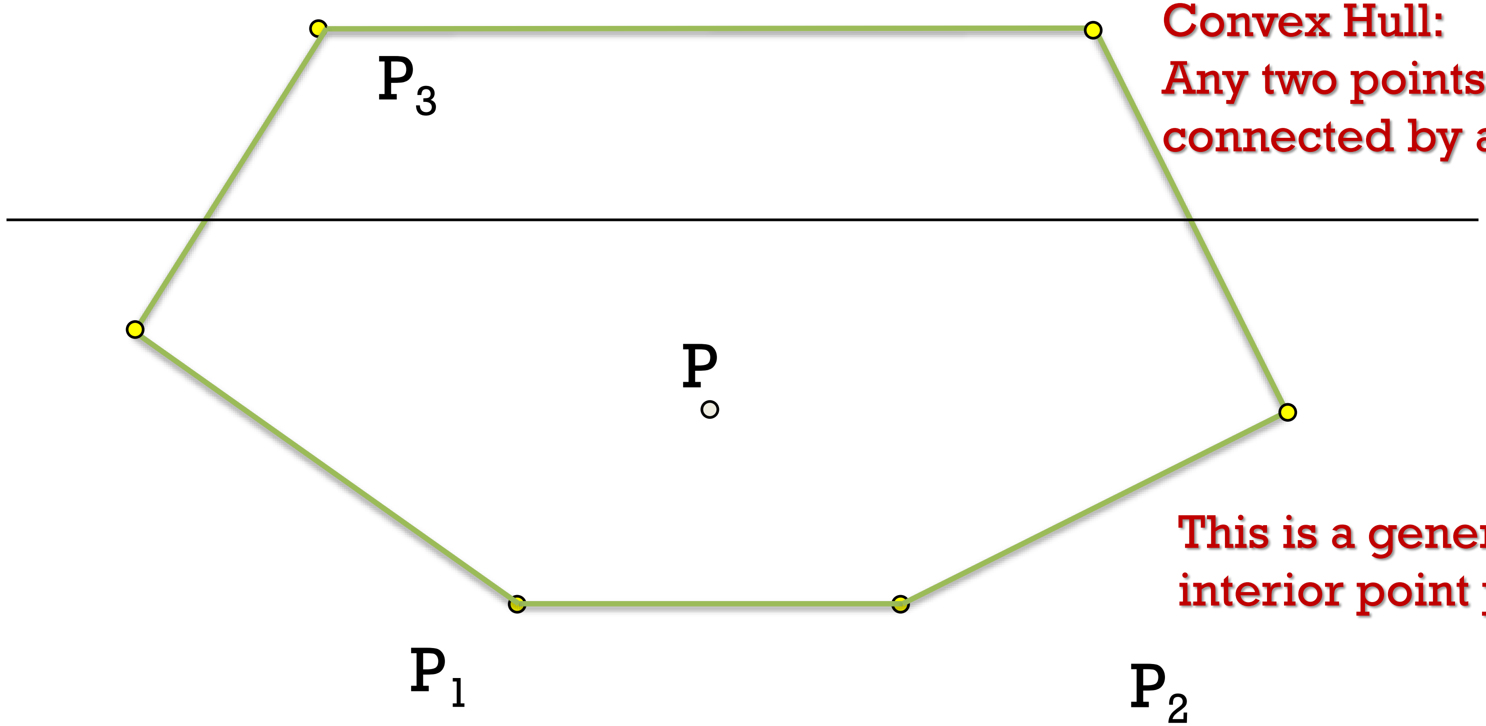
Another way:
Barycentric Coordinates



Convex Hull

$$P = \sum_{i=1}^{n} \alpha_i P_i \quad \text{such that} \quad \sum_{i=1}^{n} \alpha_i = 1$$

Convex Hull:
Any two points inside can be connected by a straight line.



This is a generalization of the interior point problem

The END