Ray Tracing I: Ray-Shape Intersection

cs348b
Matt Pharr

Overview

- Today
  - Basic ideas
  - Ray-shape intersections
- Thursday
  - Handling scenes with large numbers of objects

Classic Ray Tracing

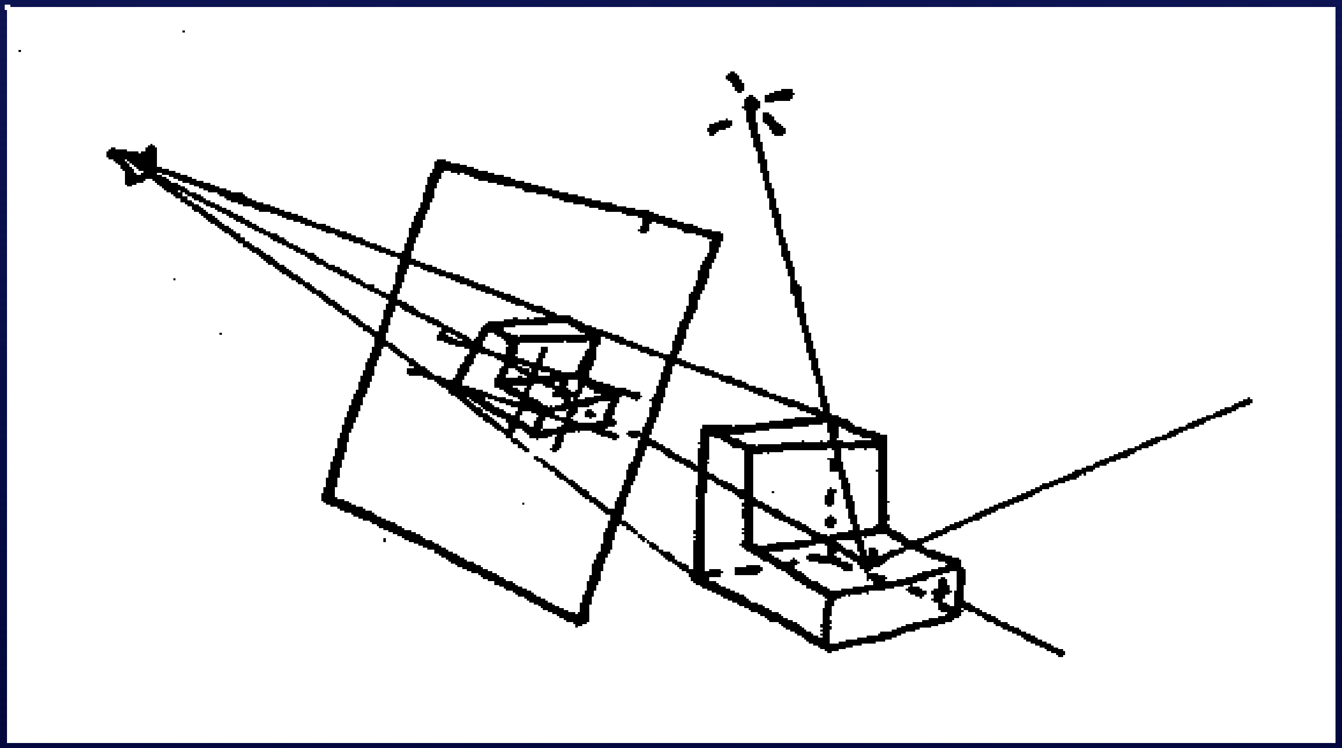
- Greek philosophers: do light rays go from the eye to the light, or from the light to the eye?
- Gauss: rays through lenses
  - Behavior of representative samples gives insight to the behavior of the system

Ray Casting vs. Z-buffering

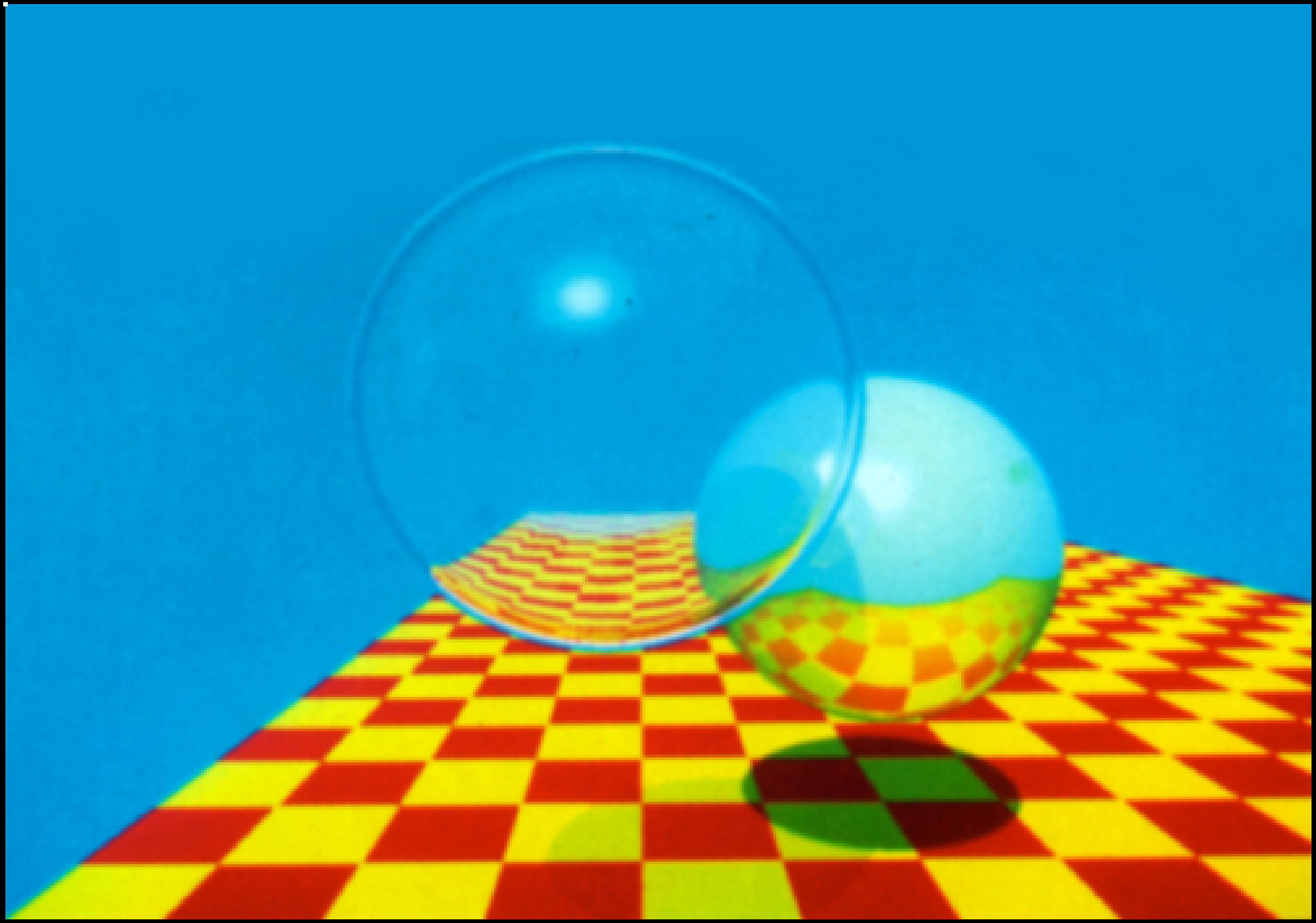
- Z-buffer:
  - For each shape
    - Find pixels where it is visible, test z, shade
- Ray casting:
  - For each pixel
    - Find first shape that is visible from it

Ray Tracing in Computer Graphics

- Appel 1968: ray casting
  - Generate image with one ray per pixel
  - Find shadows by sending ray to the light

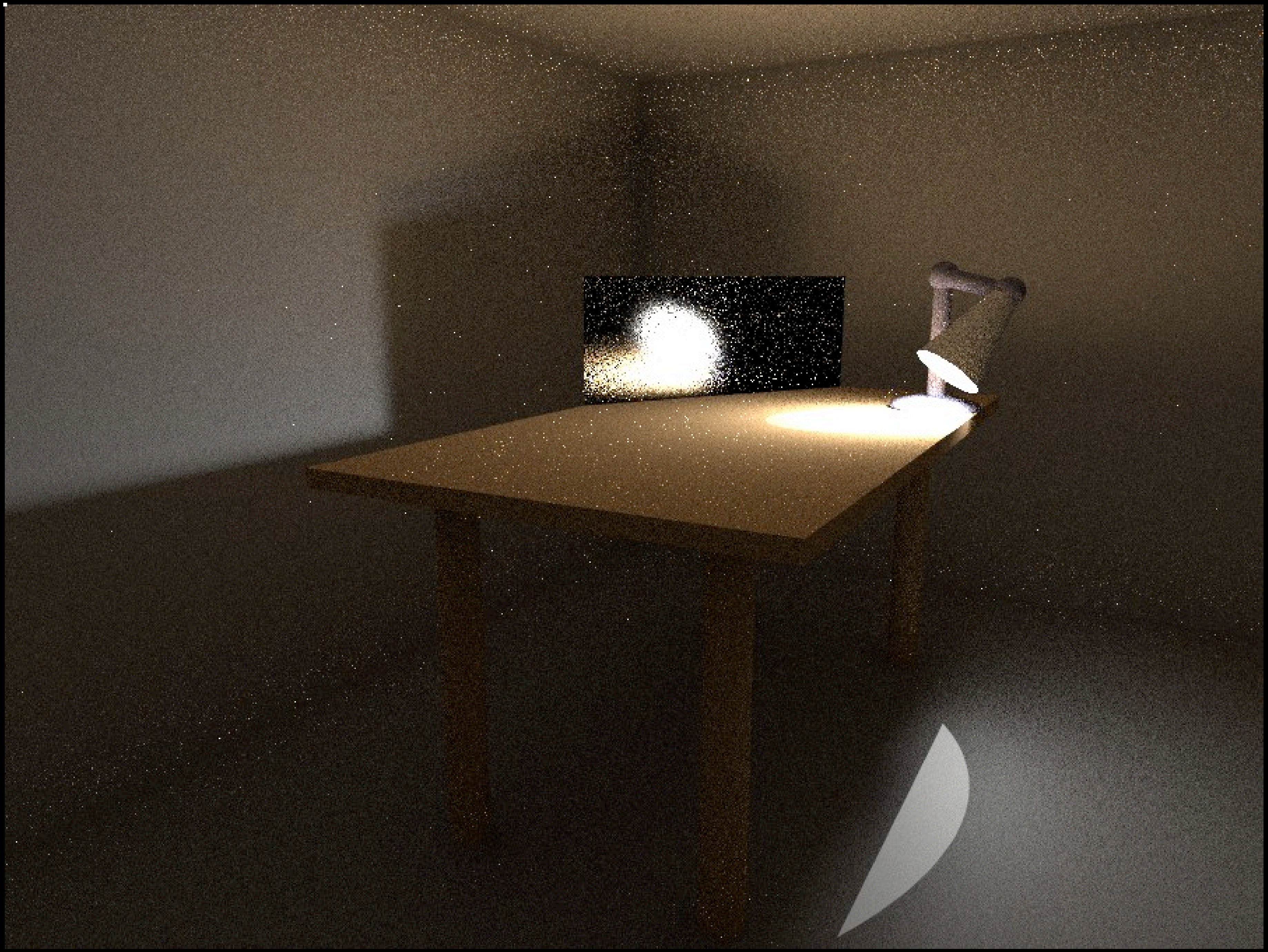


Ray Tracing in Computer Graphics



Whitted 1979: recursive ray tracing for specular reflection and refraction

Looking ahead: path tracing



Ray Tracing Demo

Ray Tracing System Architecture

- Shapes
- Accelerator
- Lights
- Materials
- Camera and sampler
- Integrator

Ray-Plane Intersection

- Ray: $r(t) = O + t \times D$ $t \in [t_{\min}, t_{\max})$
- Plane: $(P - P) \cdot N = 0$
- Substitute and solve for t...
  - Greater than tmin and less than tmax?
  - Found a valid intersection

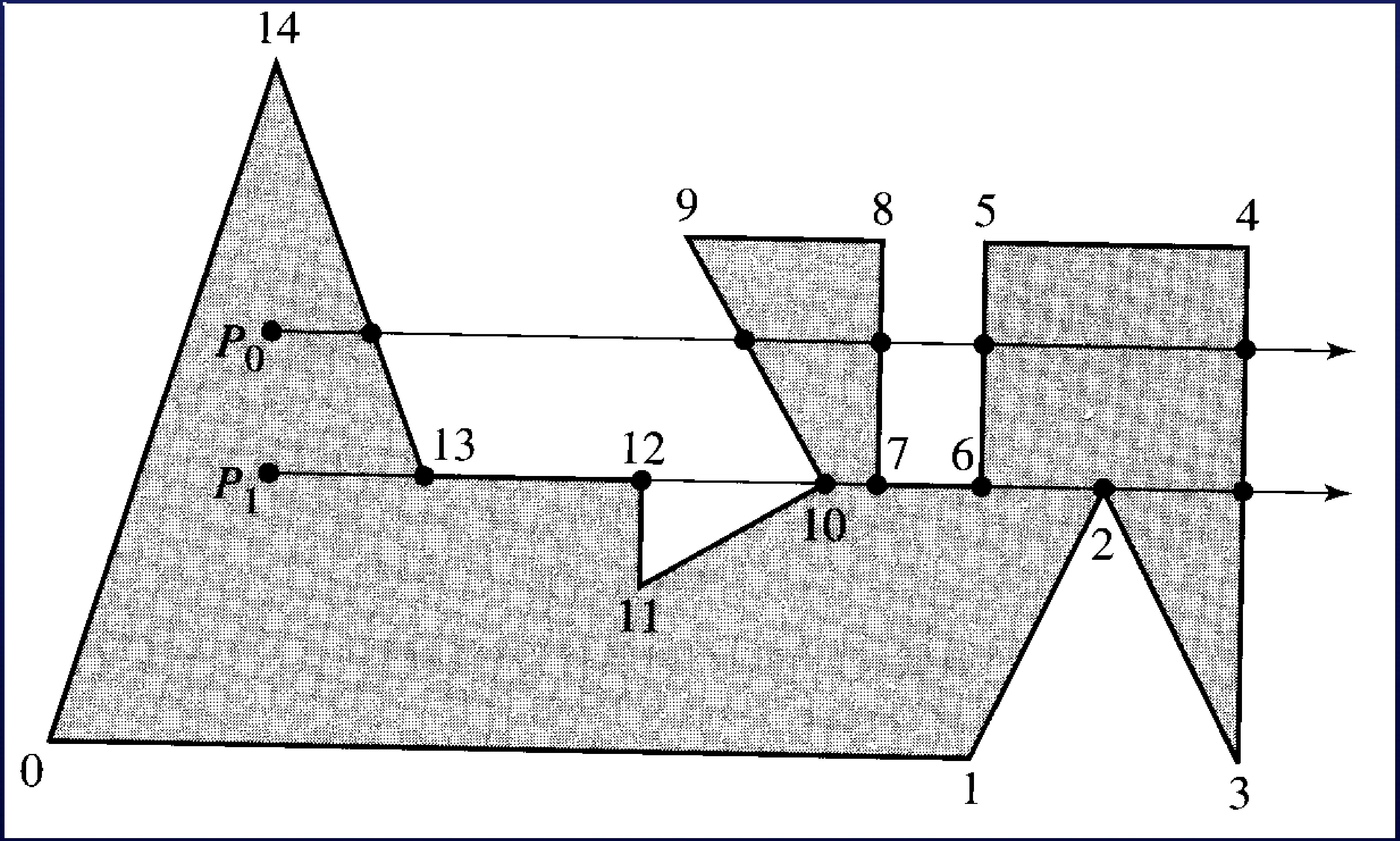
Ray-Polyhedra Intersection

- Multiple ray-plane tests for polyhedra
  - e.g. box
  - Can optimize for axis aligned case
- Compute set-intersection of t ranges
  - Degenerate? No hit
  - Continue until all slabs processed

Ray-Polygon Intersection

- Find intersection with plane of polygon
- Project to 2D and do point-in-polygon test
  - Shoot ray in (1,0) direction, count crossings
  - Odd? Inside!

Point in Polygon



Point in Polygon

```
bool ptInPolygon(point p, polygon g) {
  bool inside = false;
  for each edge from v[i] to v[i+1] {
    if ((v[i].y <= p.y && p.y < v[i+1].y) ||
        (v[i+1].y <= p.y && p.y <= v[i].y)) {
      x = v[i].x + (p.y-v[i].y) *
        (v[i+1].x-v[i].x) / (v[i+1].y-v[i].y);
      if (x > p.x)
        inside = !inside;
    }
  }
  return inside;
}
```

Ray-Sphere Intersection

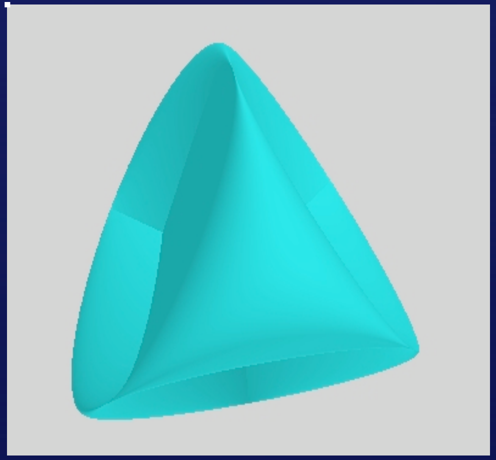
- Implicit surface: $f(x, y, z) = 0$
- Implicit sphere: $x^2 + y^2 + z^2 - 1 = 0$
- Parametric surface: $f(u, v) = (x, y, z)$
- Parametric sphere:

$$\phi = u \cdot \phi_{\max}$$

$$\theta = \theta_{\min} + v \cdot (\theta_{\max} - \theta_{\min})$$

$$x = r \cos\phi \cos\theta$$
$$y = r \sin\phi \cos\theta$$
$$z = r \sin\theta$$

Ray-Algebraic Surface Intersection



- Degree n polynomial
  - Linear: plane
  - Quadric: sphere, cylinder, paraboloid
  - Quartic: torus
- Gives univariate polynomial in t along the ray
  - Closed form solutions
  - Standard numerical algorithms approaches
- Gradient of polynomial gives surface normal at intersection

Various Details

- [tmin, tmax) range
  - Carried along with ray, updated to track closest intersection
- Object transformations
  - Transform the ray origin and direction by the inverse transform
- Normalize ray direction vector?
  - Can make intersection tests faster, but renormalizing after transform is slow

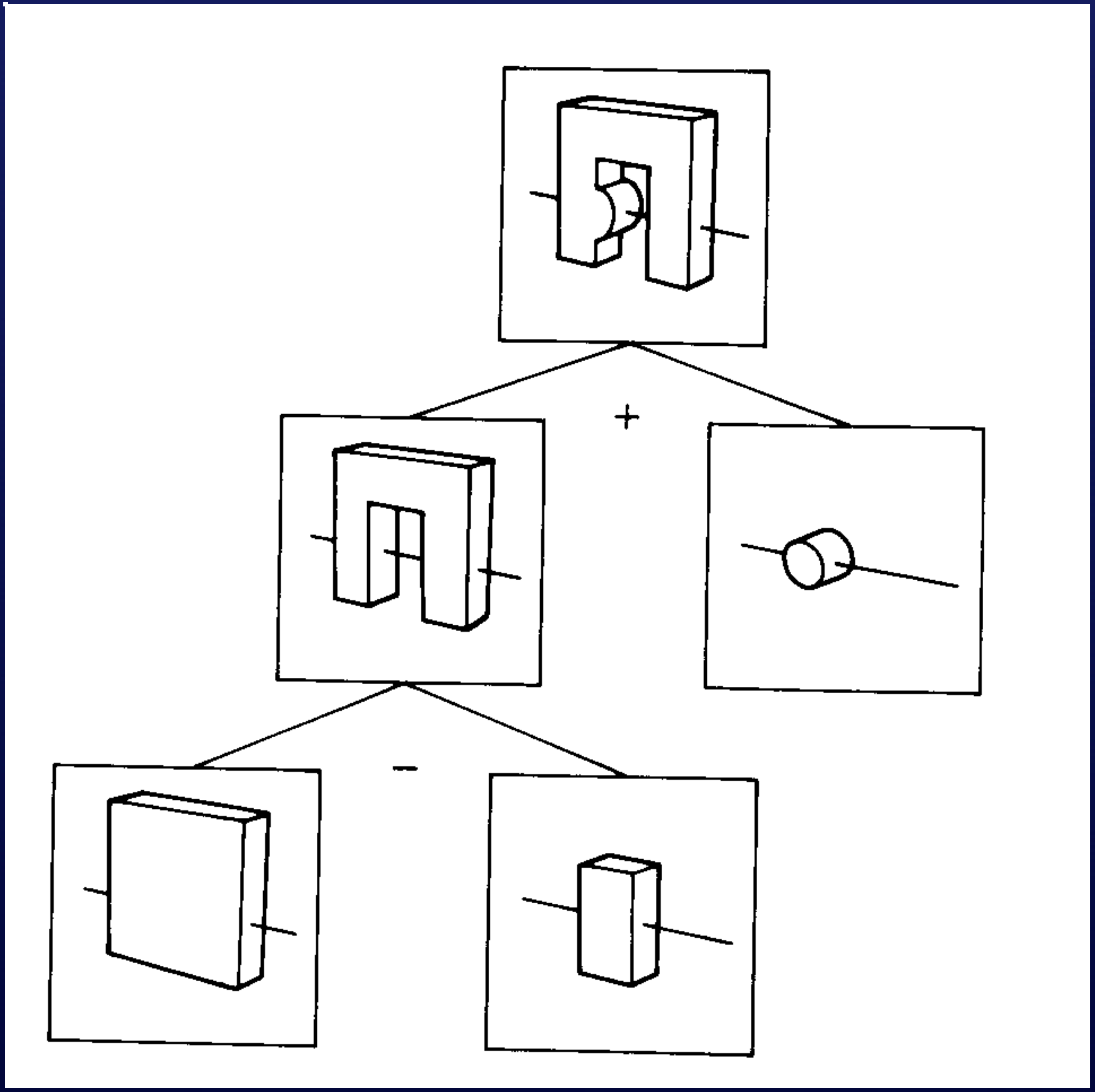
Shape Intersection Interface

- Intersect(): general rays
- IntersectP(): shadow rays: no geom. info
- WorldBound(): world space bounding box
- ObjectBound(): object space bbox
- CanIntersect(): can we call Intersect()?
- Refine(): new shapes

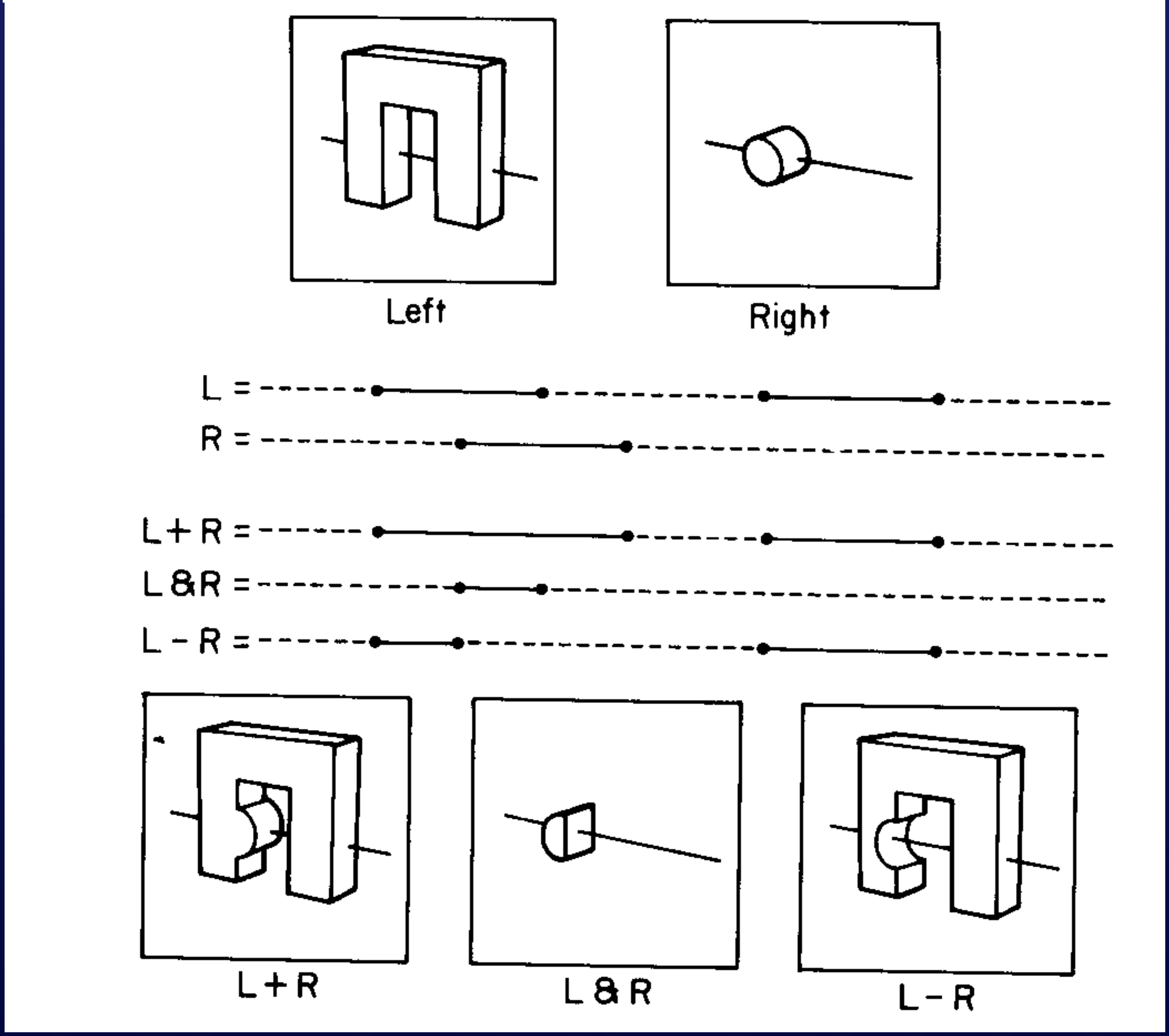
Local Differential Geometry

- Shape-independent method for representing intersection information
  - Point P
  - Normal N
  - Parametric (u,v)
  - Partial derivatives
  - (Tangents, change in normal, ...)

Constructive Solid Geometry



Constructive Solid Geometry



History

- Polygons: Appel '68
- Quadrics, CSG: Goldstein & Nagel '71
- Tori: Roth '82
- Bicubic patches: Whitted '80, Kajiya '82
- Algebraic surfaces: Hanrahan '82
- Swept surfaces: Kajiya '83, van Wijk '84
- Fractals: Kajiya '83
- Deformations: Barr '86
- NURBS: Sturzlinger '98
- Subdivision surfaces: Kobbelt et al '98