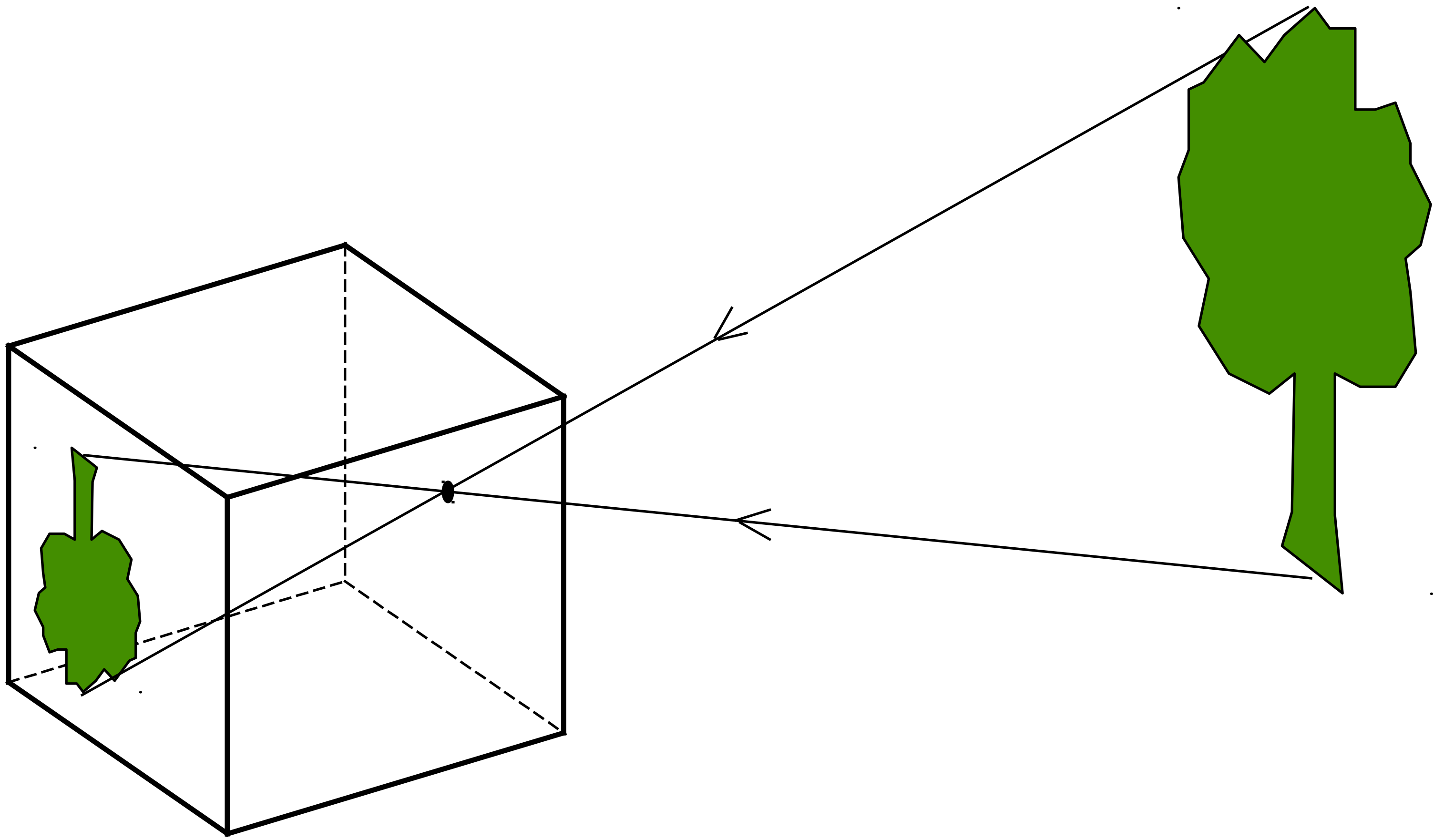
Recursive Ray Tracing

CGG MFF UK Praha

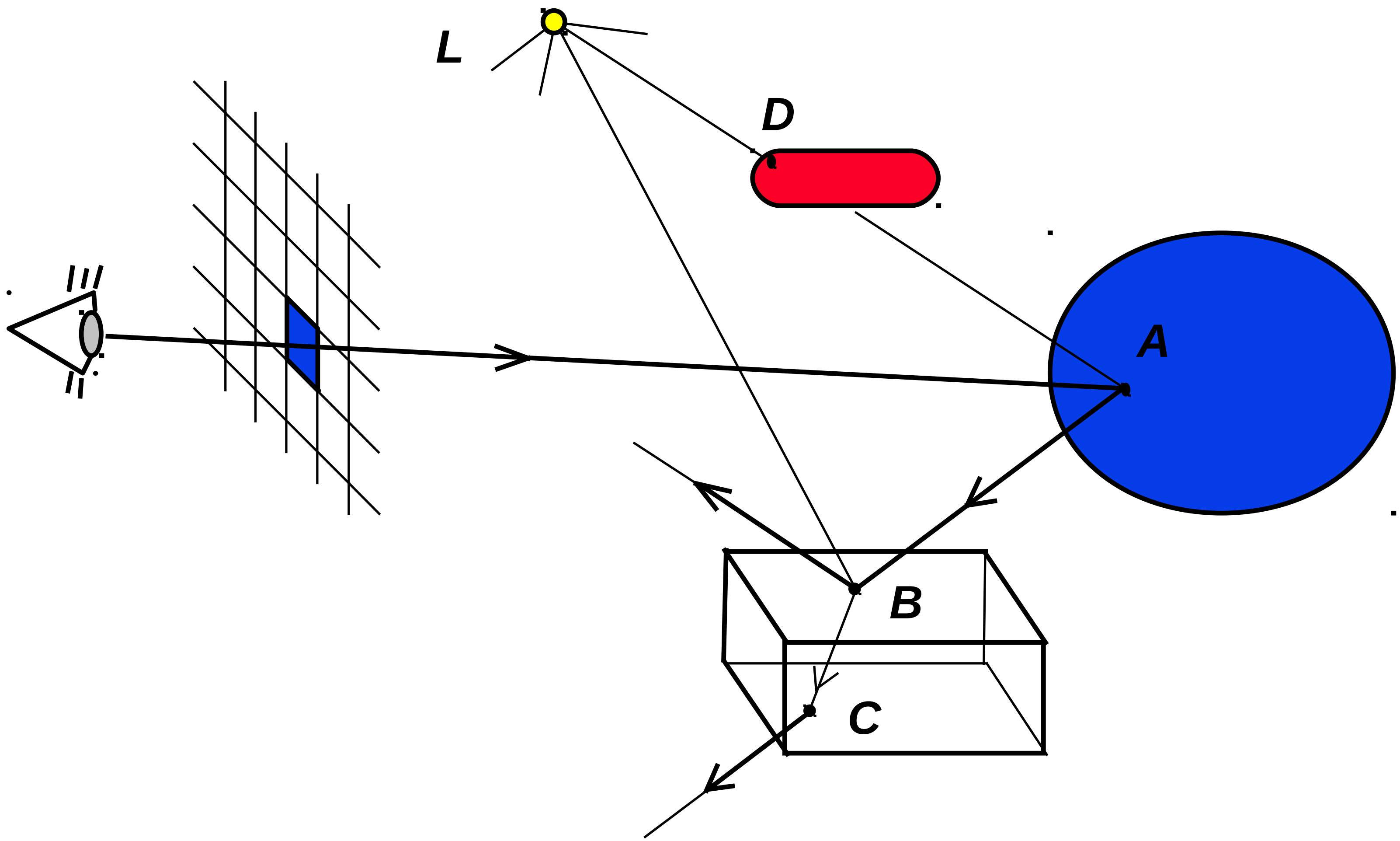
pepca@cgg.mff.cuni.cz

http://cgg.mff.cuni.cz/~pepca/

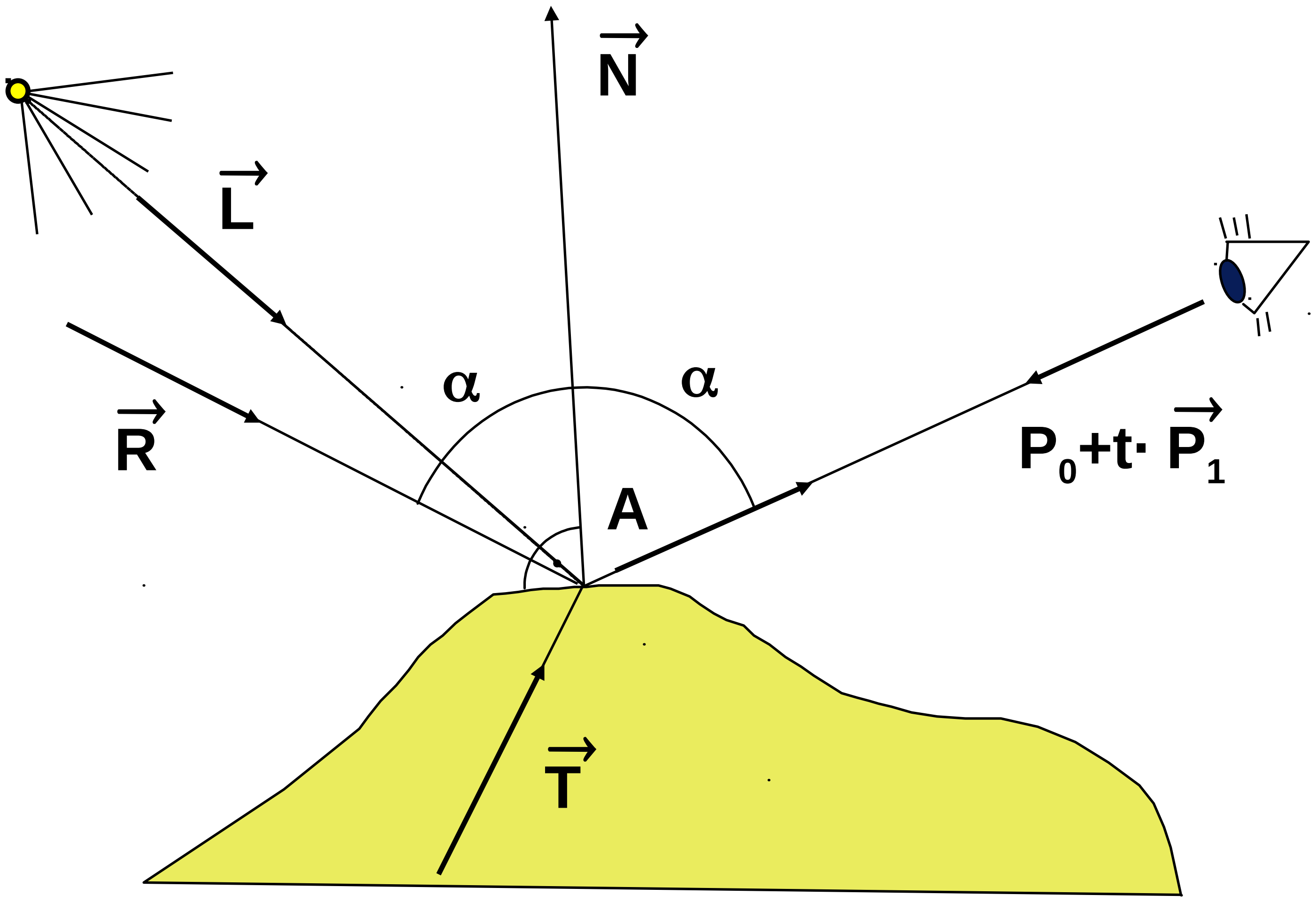
Pinhole Camera Model



Backward / Reverse Ray Tracing



Lighting Computations



Recursive Implementation

```
function Trace ( P0, P1 : Point3D; depth : integer ) : RGB;
          { P0..počátek paprsku, P1..směr, depth..počet odražení }
var A, R, T : Point3D;              { pomocné body a vektory }
    B : RGB;                        { výsledná barva }
begin
  A := Prusecik(Scena,P0,P1);  { průsečík paprsku se scénou }
  if A==0 then Sleduj := Pozadi  { paprsek na nic nenarazil }
          else
    begin                           { paprsek narazil na těleso }
      B := 0;
      for i := 1 to N do          { příspěvky od světelných zdrojů }
        if Prusecik(Scena,A,L[i]-A)==0 then B := B + kL * Svetlo(A,L[i]);
      hloubka := hloubka + 1;
      if hloubka <= maxhloubka then { konec rekurze }
        begin
          if "A je odrazivé" then
            begin
              "spočítej R"          { odražený paprsek }
              B := B + kR * Sleduj(A,R,hloubka);
            end;
```

Recursive Implementation

```
            if "A je průhledné" then
              begin
                "spočítej T"              { lomený paprsek }
                B := B + kT * Sleduj(A,T,hloubka);
              end;
          end;
        Sleduj := B;                      { nastřádaná návratová hodnota }
      end;
    end;
```

Recursion Management

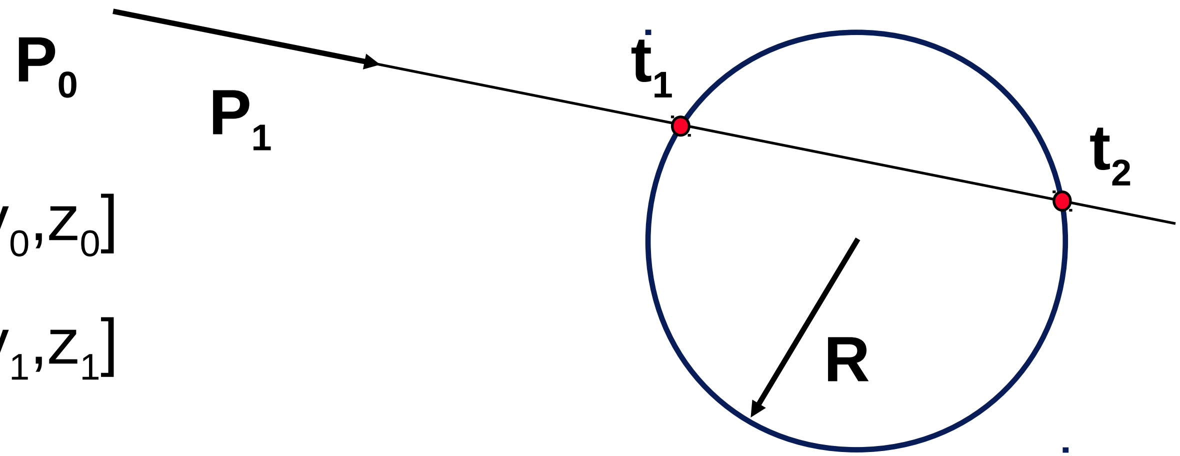
- ❶ **static** – limited by a constant (not suitable for scenes with many mirrors)
- ❷ **dynamic** – according to performance of the ray
  - ➨ „**performace**” is the percentage the ray can still contribute to the colour of a given pixel (primary rays: 100%)
  - ➨ Limit on the „performance” constant (e.g. 2-10%)
- ❸ **Combined** – limit on the recursion depth, and the „performance” of the ray

Intersection Computations

- ❶ Intersection coordinates (or „infinity”)
- ❷ ID of the solid (surface) that was hit
- ❸ Normal vector at the intersection

- ➨ **Time-consuming operation** (80-90% of render time)
  - – Acceleration techniques extremely important
- ◆ **Analytical solution** (sphere, cylinder, cube, ..)
- ◆ **Numerical solution** (subdivision surfaces, higher order surfaces, rotational surfaces, ..)

Ray - Sphere Intersections



$\mathbf{P_0} = [x_0, y_0, z_0]$

$\mathbf{P_1} = [x_1, y_1, z_1]$

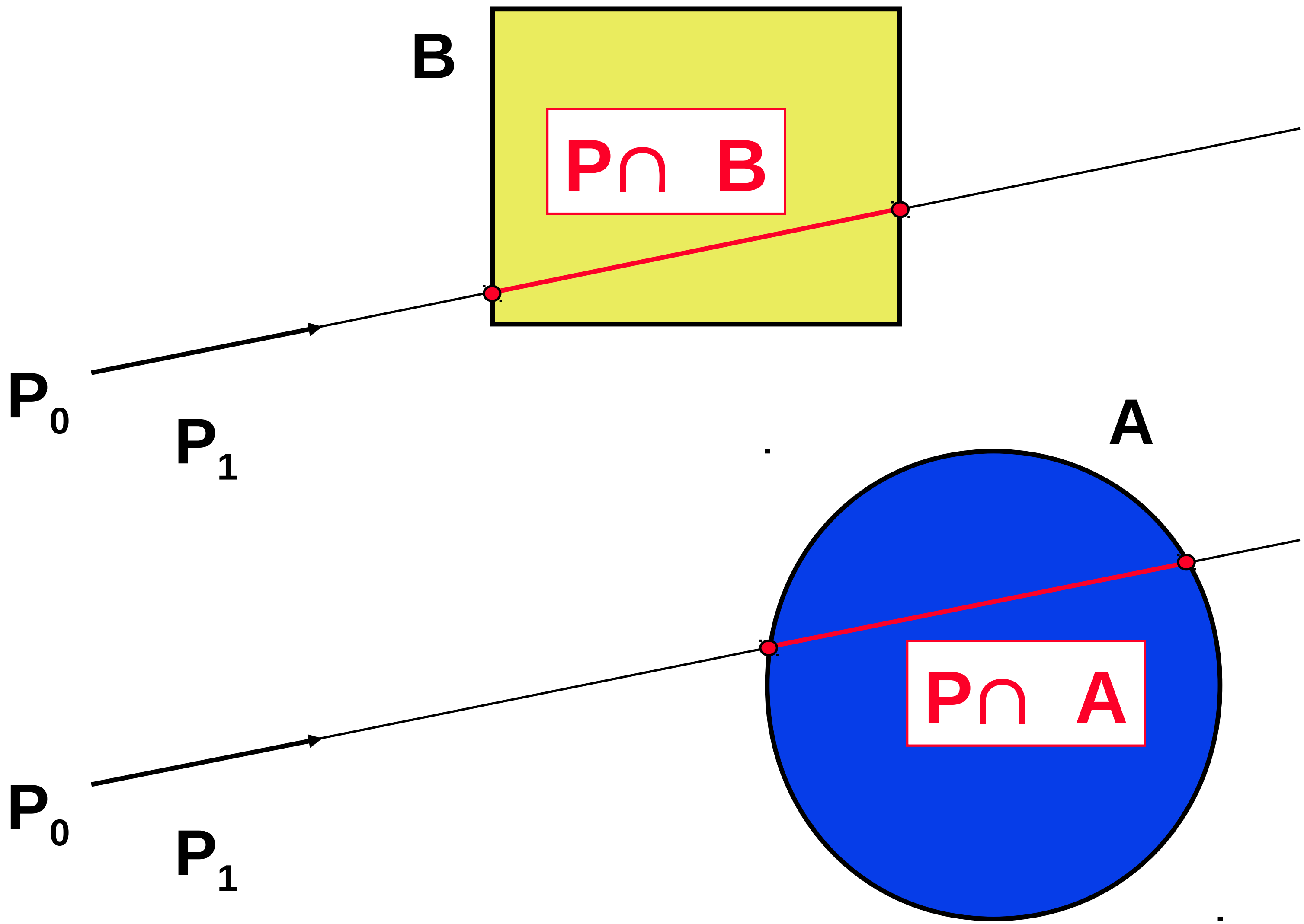
- Ray: $\mathbf{P_0} + \boldsymbol{t}\,\mathbf{P_1}, \quad \boldsymbol{t} > \mathbf{0}$ (1)
- Sphere (at origin): $\mathbf{x^2 + y^2 + z^2 - R^2 = 0}$ (2)
- When substituting (1) into (2) we obtain a **quadratic equation** for ***t***:

$$\mathbf{t^2\,(x_1^2+y_1^2+z_1^2) + 2t\,(x_0x_1+y_0y_1+z_0z_1) + x_1^2+y_1^2+z_1^2-R^2 = 0}$$

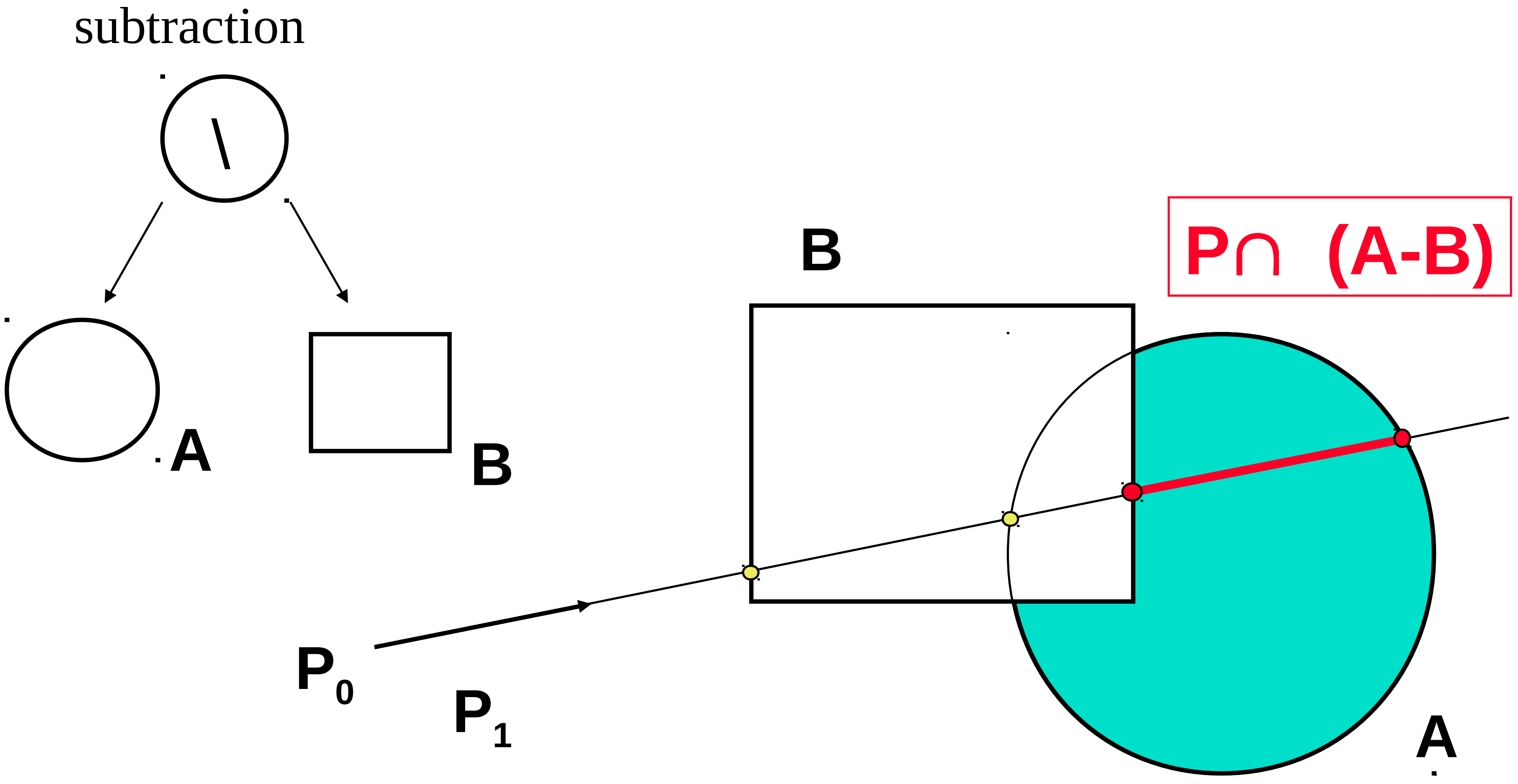
Ray Intersections with CSG

- For **elementary solids**, intersections can be calculated
  - Start and end of ray traversal through a solid body
- **Set theoretic operations** on all intersections along the ray:
  - Distributive: $\mathbf{P \cap (A-B) = (P \cap A) - (P \cap B)}$
  - The usual ray-object intersection is an interval
- **Geometric transformations**:
  - The inverse transformation is applied to the ray

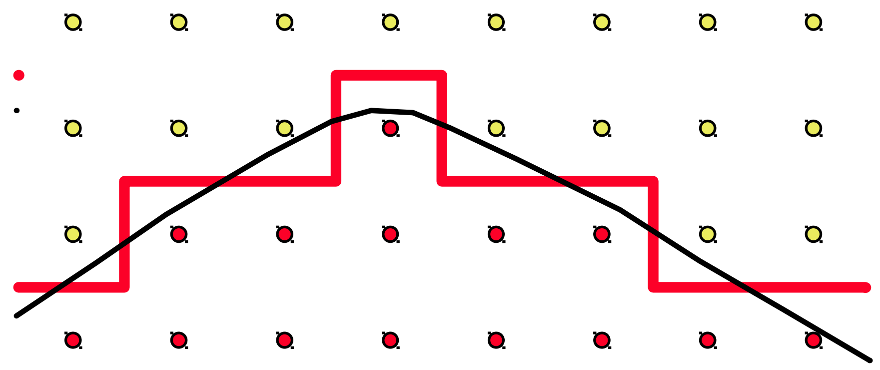
Intersection P∩ A,P∩ B



Intersection: P∩ (A-B)

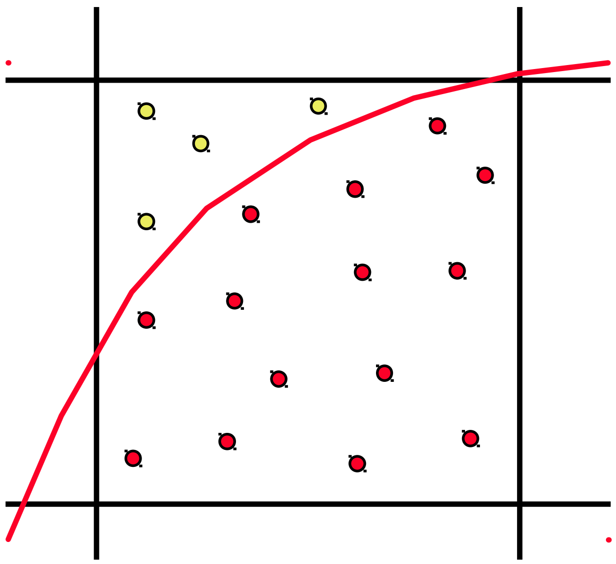


Anti-aliasing



- Only one ray per pixel leads to „**aliasing**"
  - Jagged edges
  - Interference
- **Increased resolution** only solves the problem partially (and at great expense)

Multiple Sampling



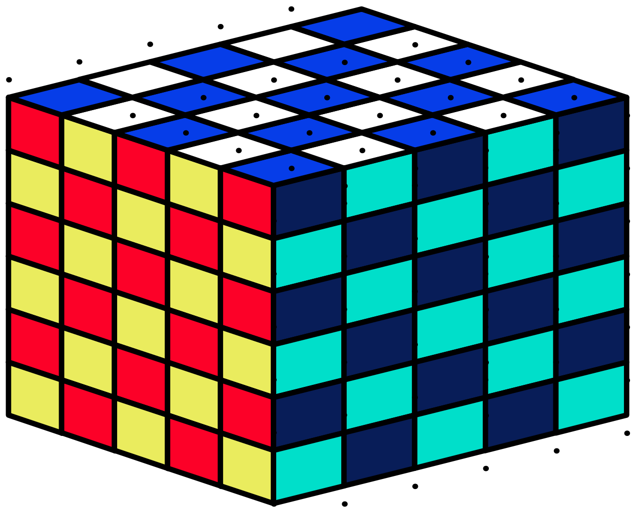
- Multiple rays are shot into each pixel
- The resulting colour is obtained via arithmetic mean
- Transitions are softer (no jaggies)
- The rays should cover the pixel area evenly, but not regularly!

Textures

- Changing the **colour** on object surfaces
- They can also affect **reflectivity** (**k_D** and **k_S**), **normal vectors**, ...

- Realistic capture of **material properties** is possible (micro-structure)
  - Wood, orange skin, polished metal ..
- Replaces complex **geometry** (water waves, ..)

2D Textures



- Covers the object surface (like wallpaper)

- **Texture maping**: $[x,y,z] \rightarrow [u,v]$
- **Custom texture**: $[u,v] \rightarrow$ **colour** (normal, ..)

3D Textures

- Model changes that occur **inside a solid**
- Mimics the **internal structure of materials** (wood, marble, ...)

- It is not necessary to perform a **mapping**
- **3D texture**: [x,y,z] → **colour** (reflectance, etc.)

- Frequently, **3D noise functions** are used (simulation of natural phenomena)

End

Further information:

- **A. Glassner**: ***An Introduction to Ray Tracing***, Academic Press, London 1989, 1-31
- **Jiří Žára a kol.**: ***Počítačová grafika***, principy a algoritmy, 374-378