matfyz

Computer Graphics Charles University

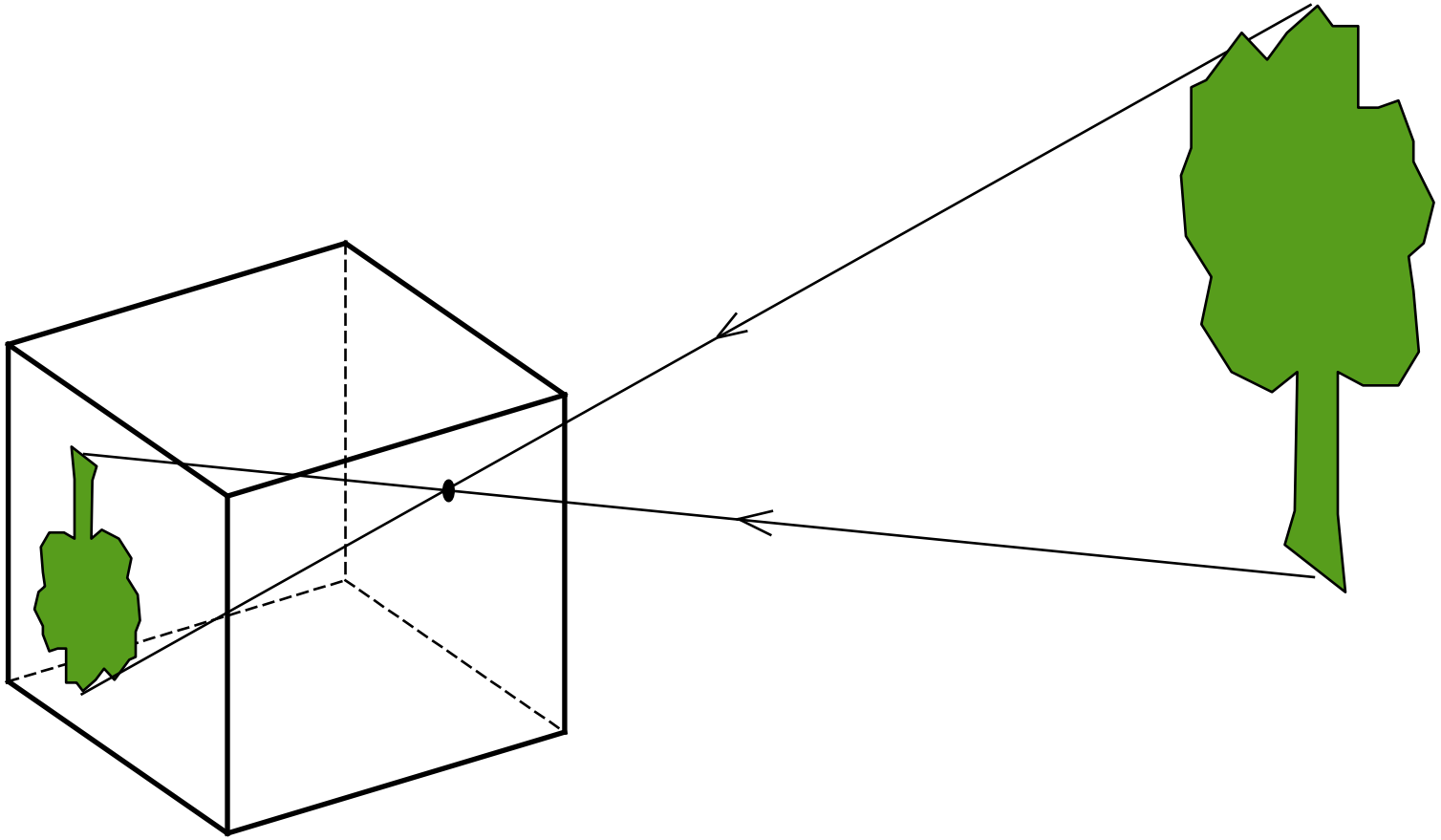
Rekurzivní sledování paprsku (Ray-tracing)

CGG MFF UK Praha

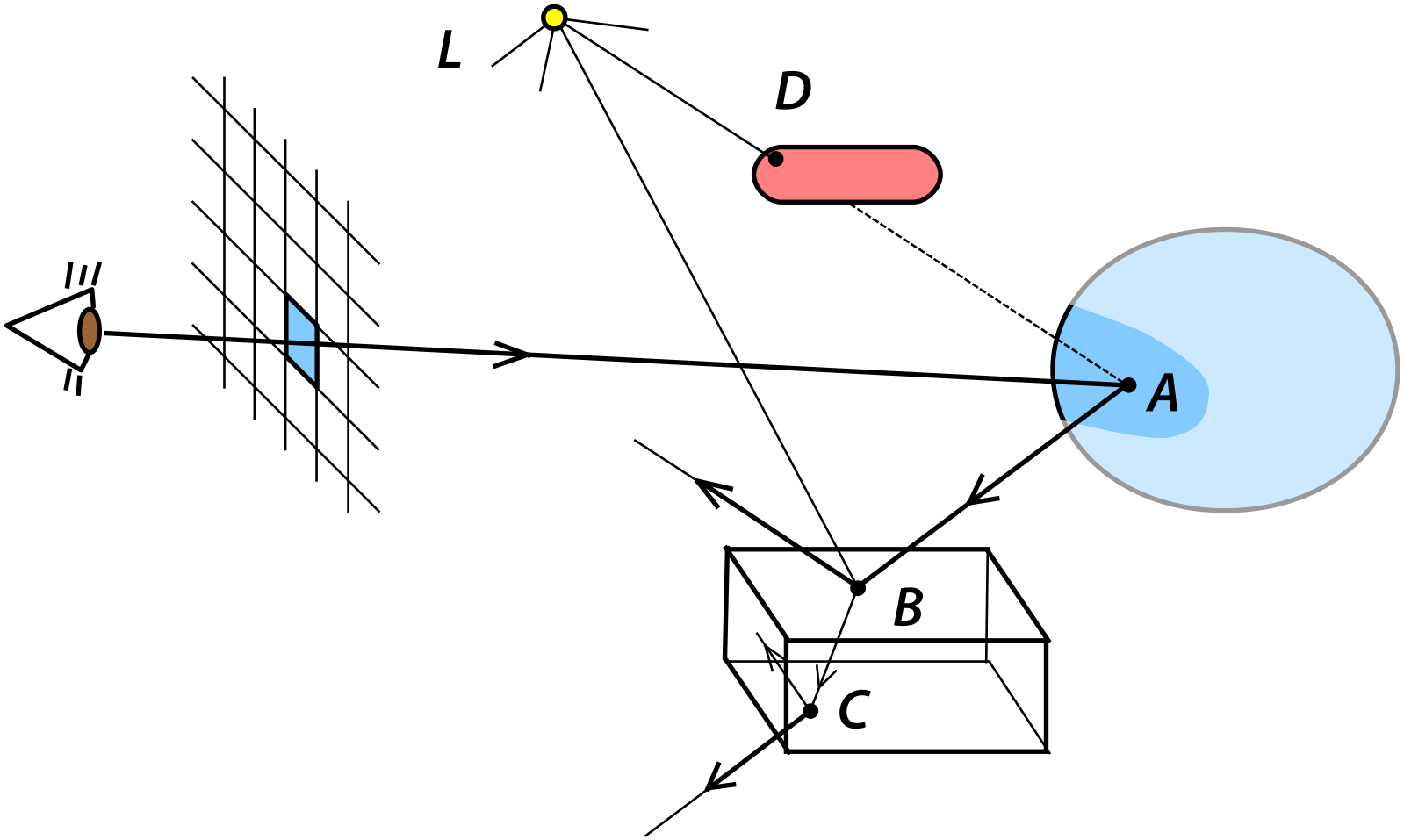
pepca@cgg.mff.cuni.cz

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

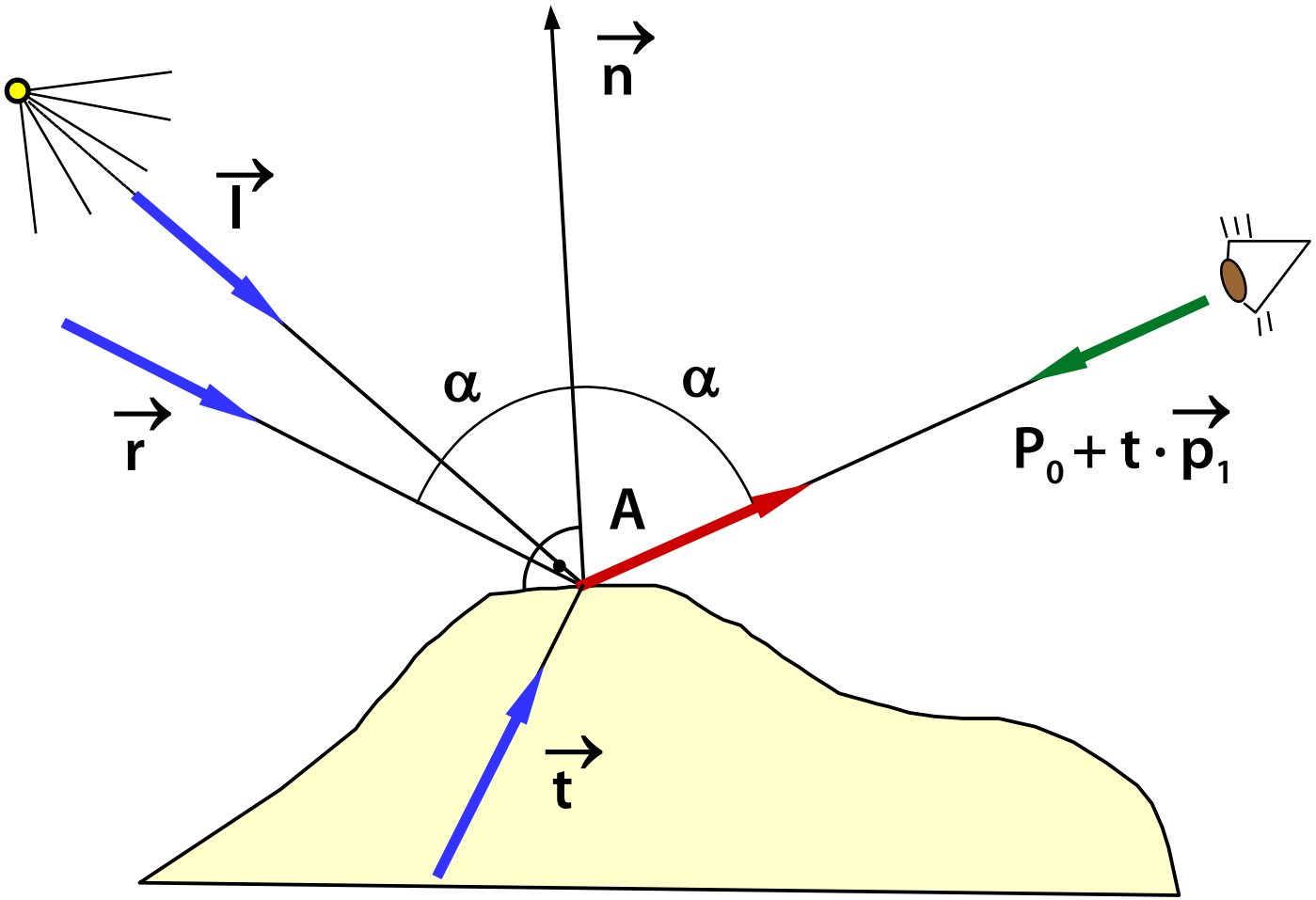
Model dírkové kamery



Zpětné sledování paprsku



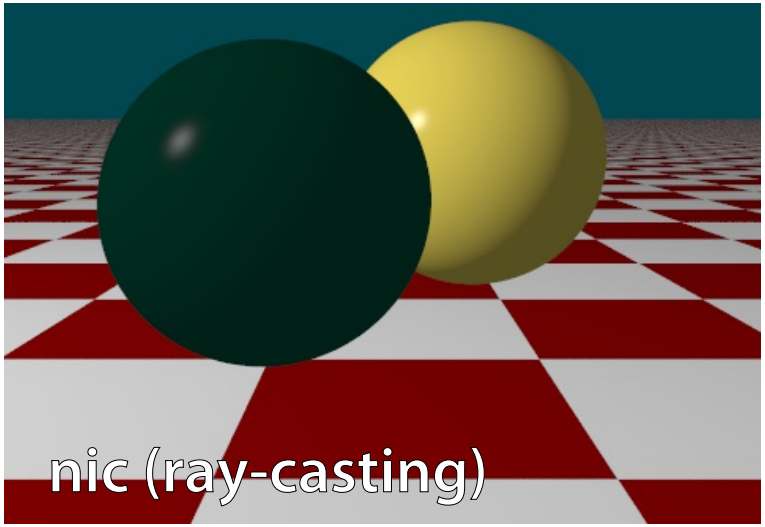
Skládání světla

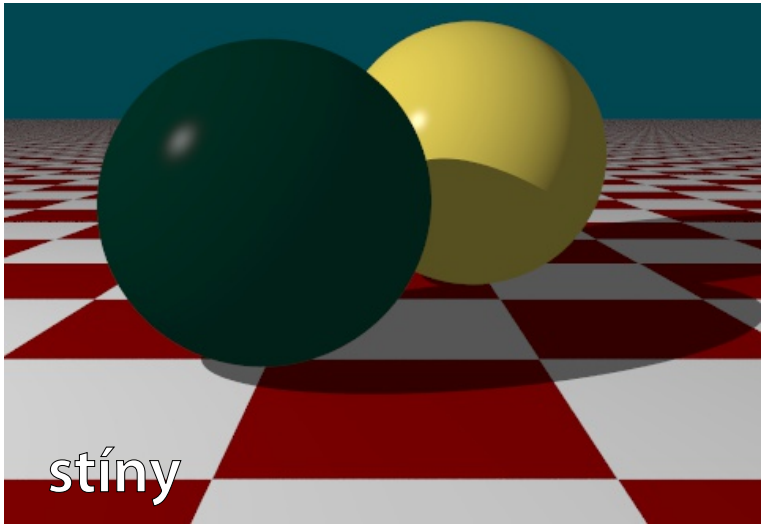


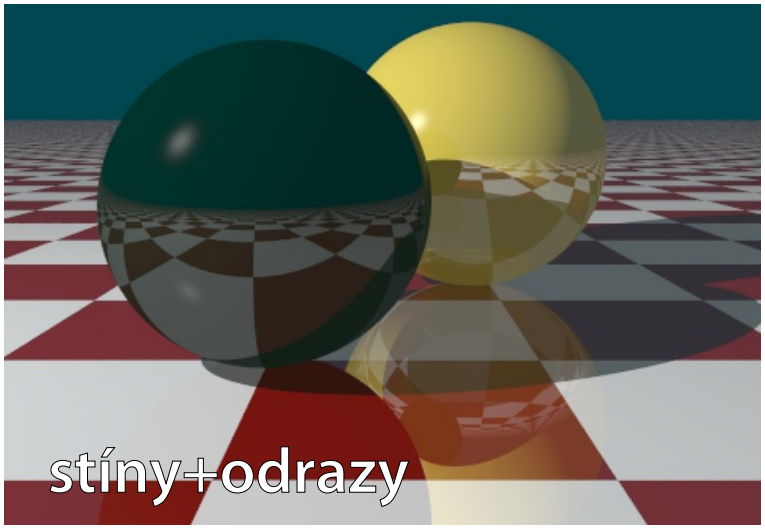
Rekurzivní implementace

```
int maxDepth = 10;              // Maximum recursion level.
RayScene scene;                 // Global scene object (geometry, materials, light sources…)

RGB shade (Vector3d P0, Vector3d p1, int depth)
  // P0 – ray origin, p1 – ray direction, depth – interactions so far
{
  Vector3d A = intersection(scene, P0, p1);
  if (!isValid(A)) return scene.background; // No intersection at all.

  RGB color{0};                 // Result color.
  for (const auto& light : scene.lightSources)
    if (!isValid(intersection(scene, A, light.point - A)))
      color += scene.kL(A) * light.contribution(A, -p1, scene.material(A), scene.normal(A));

  if (++depth >= maxDepth) return color;

  if (scene.isGlossy(A))        // Recursion - reflection.
  {
    Point3d r = reflection(p1, scene.normal(A));
    color += scene.kR(A) * shade(A, r, depth);
  }

  if (scene.isTransparent(A)) // Recursion – refraction.
  {
    Point3d t = refraction(p1, scene.normal(A), scene.index(A));
    color += scene.kT(A) * shade(A, t, depth);
  }

  return color;
}
```

Jednotlivé složky







Řízení hloubky rekurze

Statické – omezení konstantou (nehodí se pro scény obsahující zrcadla i méně odrazivé lesklé povrchy)

Dynamické – podle „významu“ (importance, performance) paprsku

- „**význam**” je procentuální podíl právě sledovaného paprsku na výsledné barvě pixelu (pro primární paprsky: 100%)
- omezení „významu” konstantou (např. 1-2%)

Kombinované – omezení hloubky i „významu” paprsku

Výpočet průsečíku

Geometrický výpočet, jehož výsledkem jsou

- souřadnice průsečíku (stačí 1D, speciální hodnota: „nekonečno”)
- normálový vektor povrchu tělesa
- 2D texturové souřadnice
- číslo tělesa (plochy), odkaz na těleso (barva, materiál…)

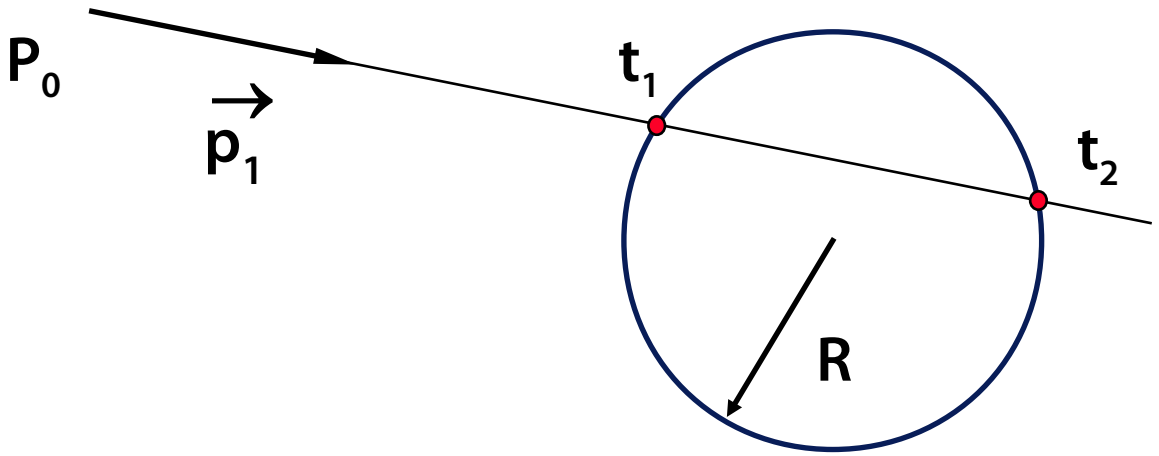
Časově nejnáročnější operace (90-95% času)

- urychlovací metody

Analytický výpočet (koule, válec, kvádr…)

Numerický výpočet (aproximační plochy, rotační tělesa, implicitní povrchy…)

Průsečík paprsku s koulí



$P_0 = [x_0, y_0, z_0]$

$\vec{p}_1 = [x_1, y_1, z_1]$

Paprsek: $\mathbf{P(t) = P_0 + t\,\vec{p}_1,\quad t > 0}$ (1)

Koule (střed v počátku): $\mathbf{x^2 + y^2 + z^2 - R^2 = 0}$ (2)

Po dosazení (1) do (2) vyjde **kvadratická rovnice (*t*)**

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

Průsečík s CSG scénou

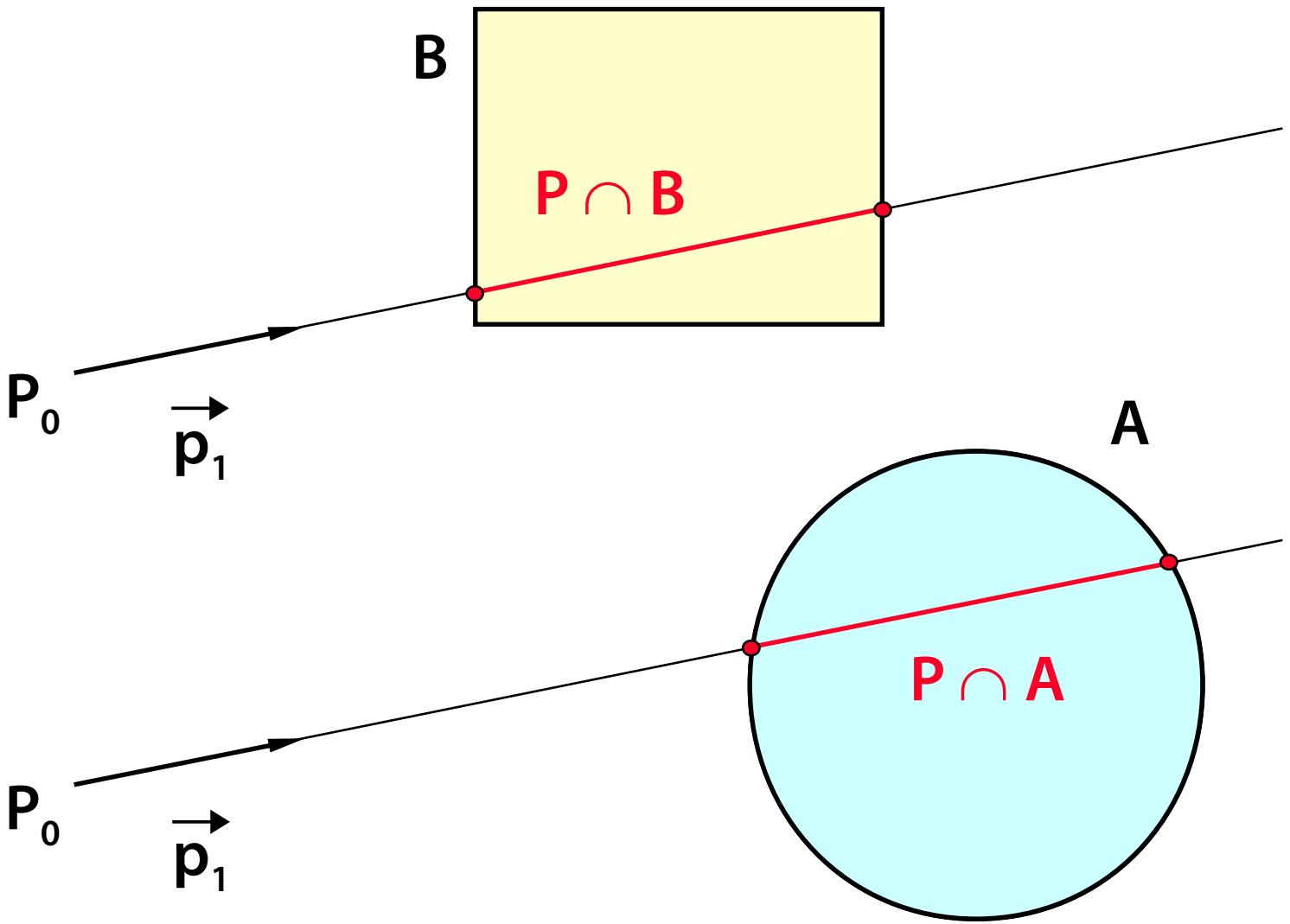
Pro **elementární tělesa** lze snadno průsečíky spočítat

- začátek a konec průniku paprsku s tělesem pro konvexní tělesa

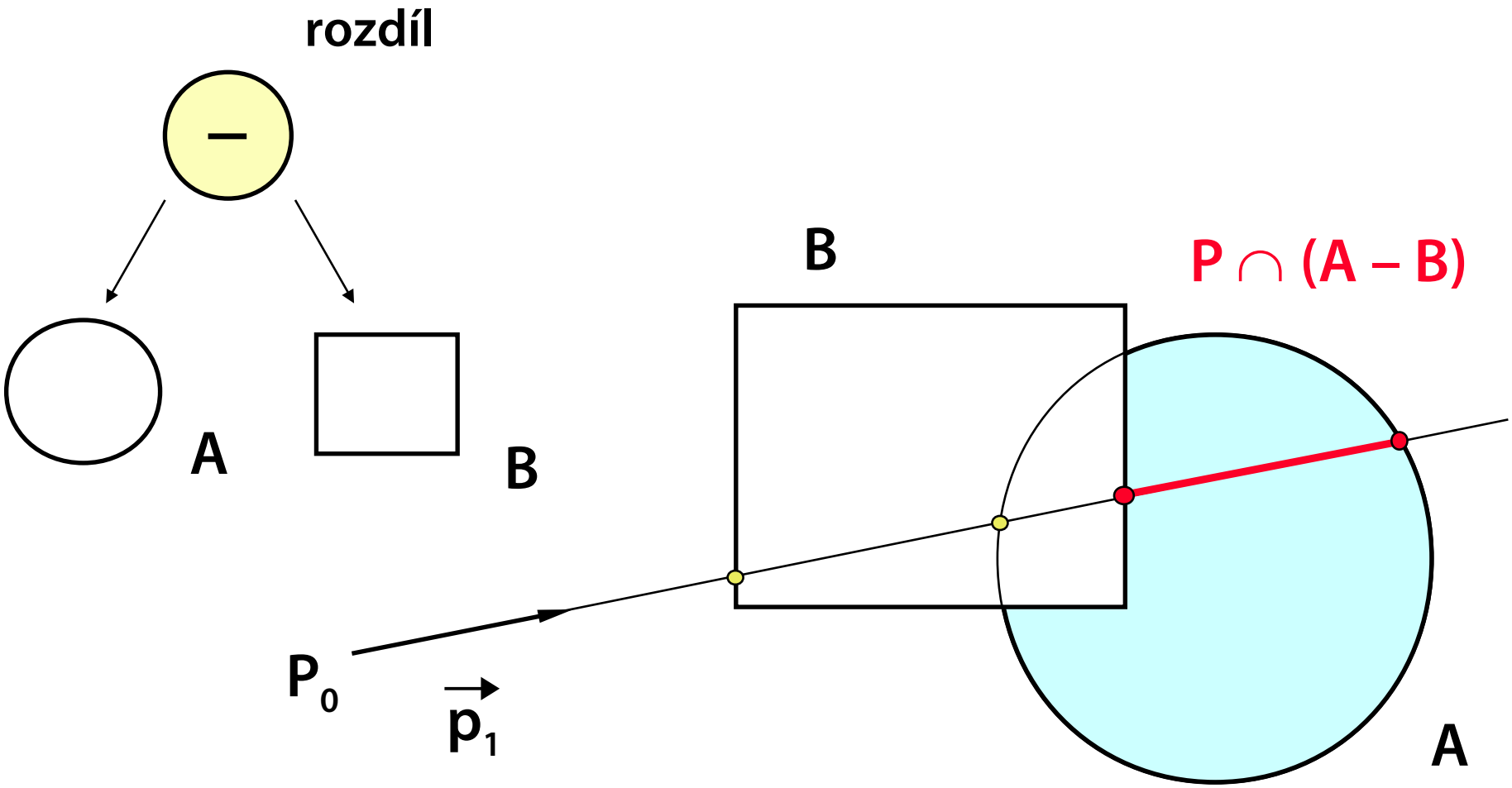
Množinové operace se provádí na polopřímce paprsku

- díky distributivitě: $\mathbf{P} \cap (\mathbf{A} - \mathbf{B}) = (\mathbf{P} \cap \mathbf{A}) - (\mathbf{P} \cap \mathbf{B})$
- obecný průnik paprsku se scénou je množina intervalů

Průsečíky $P \cap A$, $P \cap B$



Průsečík $P \cap (A - B)$



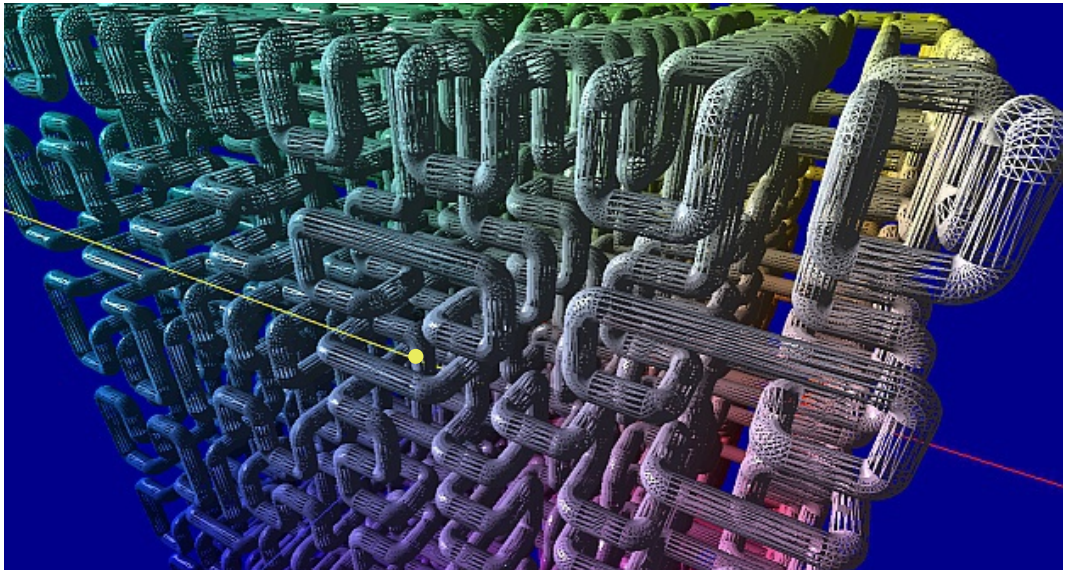
Průsečík s trojúhelníkovou sítí

Scéna reprezentována **sítí trojúhelníků**

- jednoduchý koncept (jednoduché API)
- jakoukoli geometrii lze pomocí trojúhelníků aproximovat

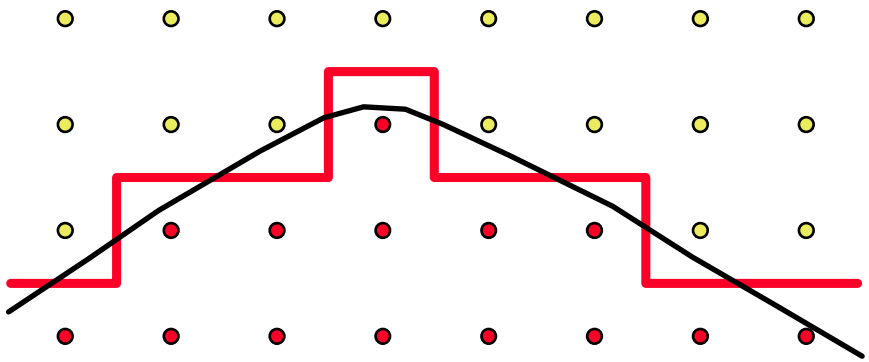
Jednoduchý test **paprsek – trojúhelník**

- velké množství trojúhelníků (10^6 až 10^{10}), ***O(N)*** je moc pomalé
- urychlovací techniky se snaží o lepší složitost, např. ***O(log N)***



N ≈ 10^6

Vyhlazování (anti-aliasing)

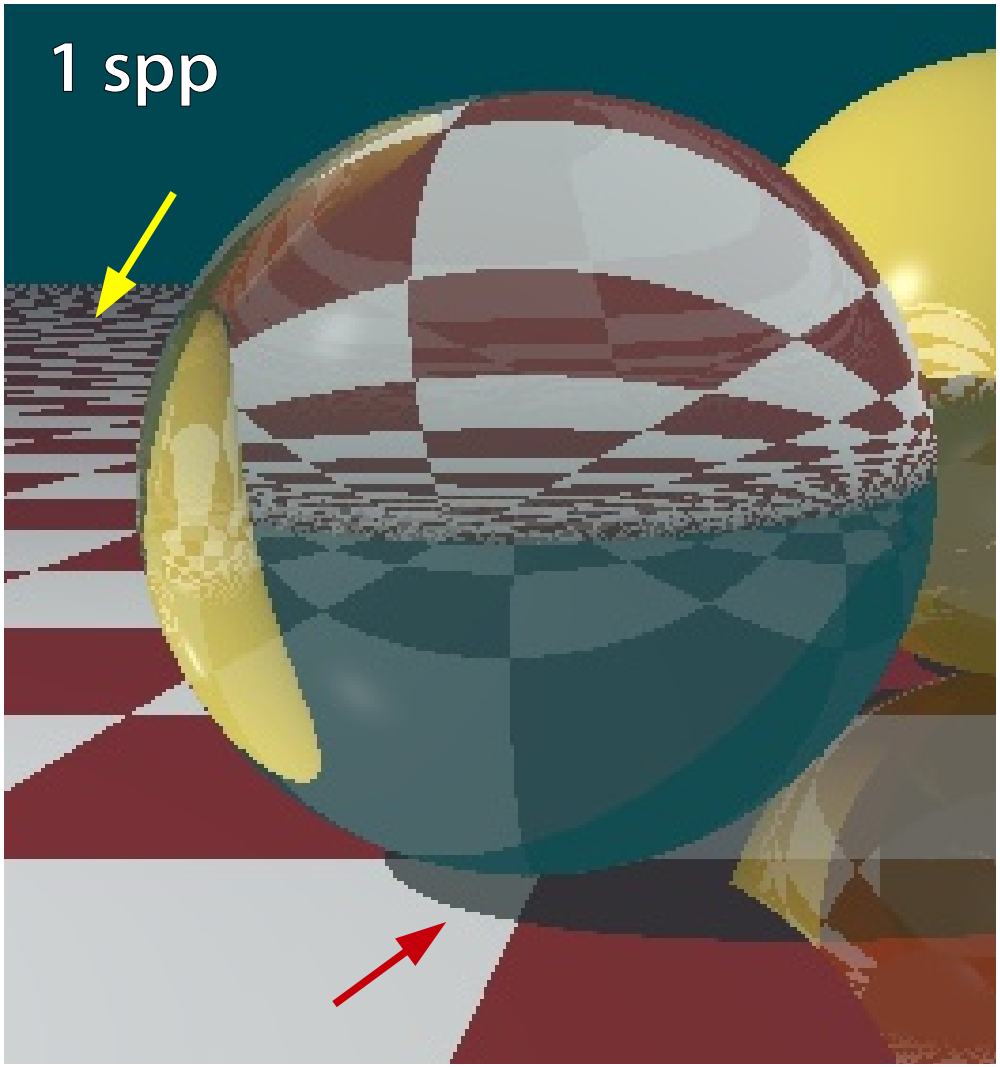


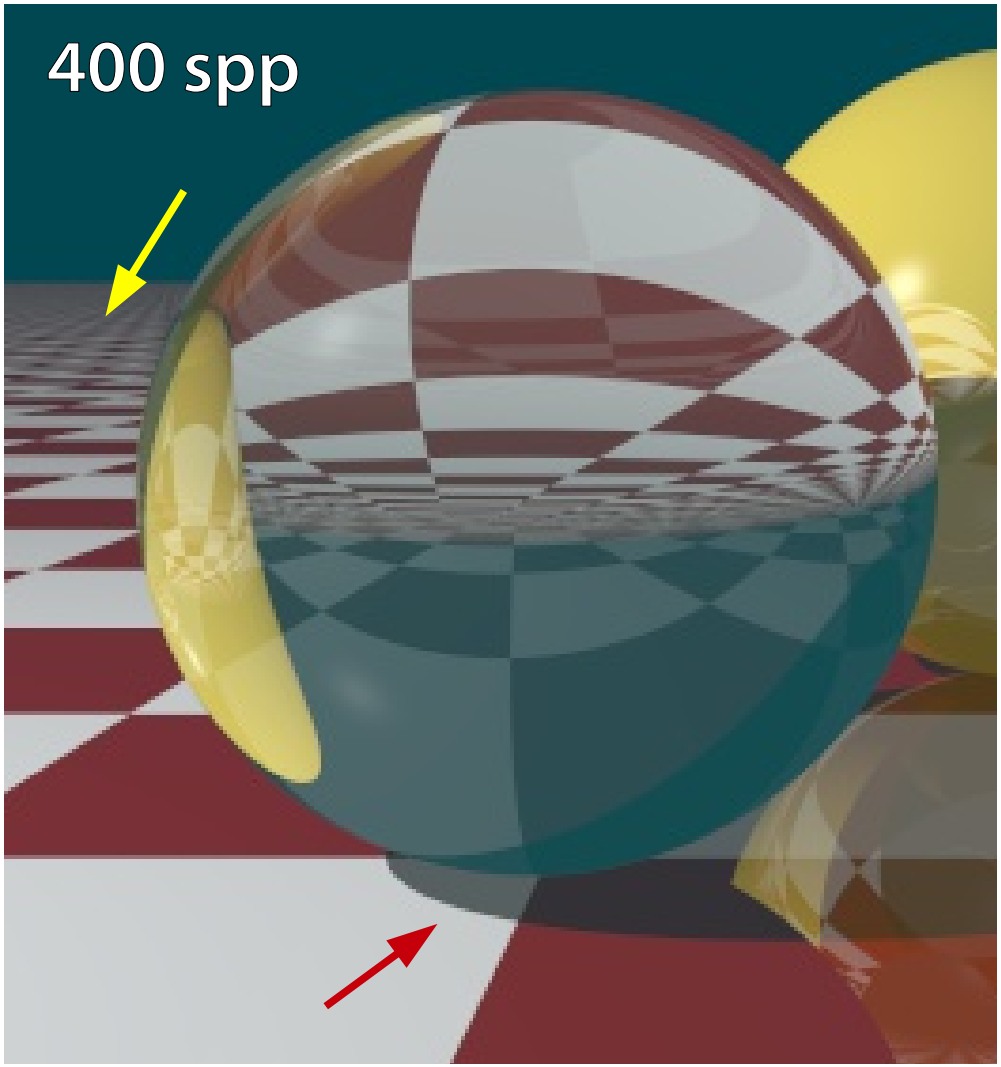
Pouze jeden paprsek na jeden pixel – vzniká tzv. „**alias**"

- zubaté okraje
- interference

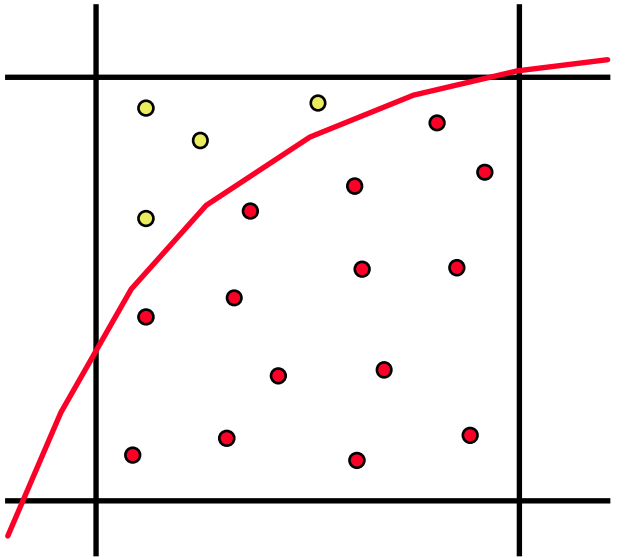
Zvětšením rozlišení se problém nevyřeší

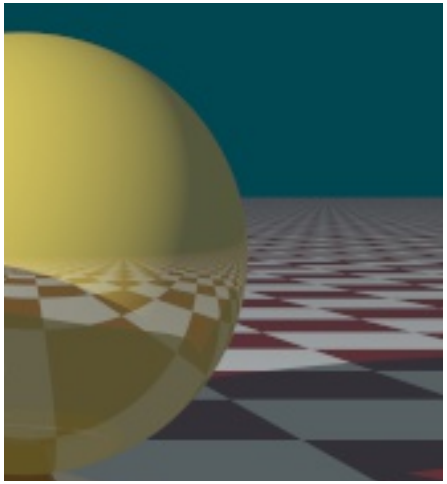
Ukázka vyhlazování (super-sampling)





Převzorkování (super-sampling)





Posílá se **více paprsků** jedním pixelem

Výsledná barva se spočte jako **aritmetický průměr**

Přechody budou **jemnější** (bez zubů)

Paprsky by měly pokrývat plochu pixelu **rovnoměrně**, ale ne úplně pravidelně!

Textury

Změna **barvy** na povrchu předmětů

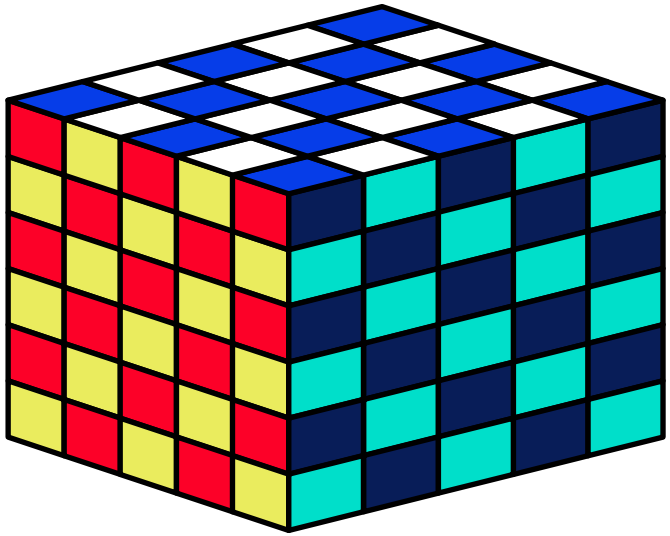
Mohou ovlivňovat též **odrazivost** ($\mathbf{k_D}$ a $\mathbf{k_S}$), **normálový vektor** („normal map“)…

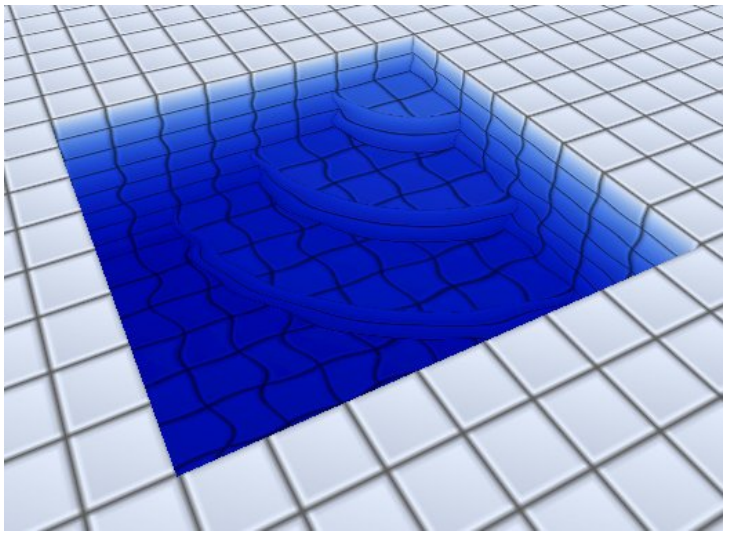
Realistické napodobení **fyzikálních vlastností materiálu** (barevný vzorek, mikro- i makro-struktura povrchu)

- dřevo, kůra pomeranče, omítka, leštěný kov…

Nahrazení složité **geometrie** (vlny na vodě…)

2D textura





Pokrývá **povrch** tělesa (jako tapeta, samolepka)

Mapování textury: $[x, y, z] \rightarrow [u, v]$

Vlastní textura: $[u, v] \rightarrow$ **barva** (normála, materiál…)

3D textury

Reprezentují změny veličin **uvnitř tělesa**

Napodobují **vnitřní strukturu materiálu**

- dřevo, mramor…

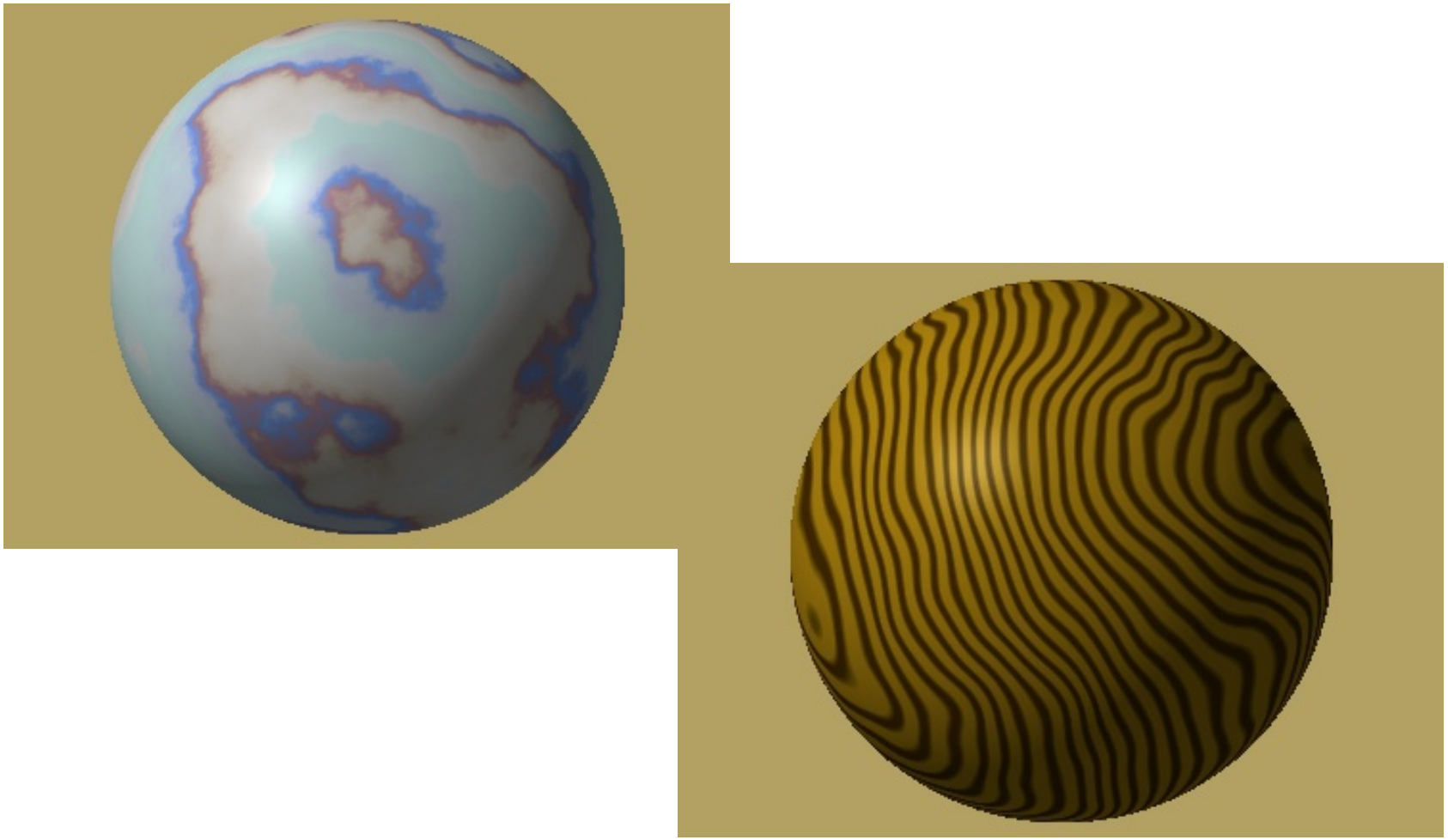
Není třeba **mapování**

3D textura: $[x, y, z] \rightarrow$ **barva** (materiál…)

Často se využívají 3D **šumové funkce**

- napodobení náhodné turbulence/vrásnění

Příklady 3D textur



Literatura

A. Glassner: ***An Introduction to Ray Tracing***, Academic Press, London 1989, 1-31

Jiří Žára a kol.: ***Počítačová grafika, principy a algoritmy***, 374-378