Shadow casting

CGG MFF UK Praha

pepca@cgg.mff.cuni.cz

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

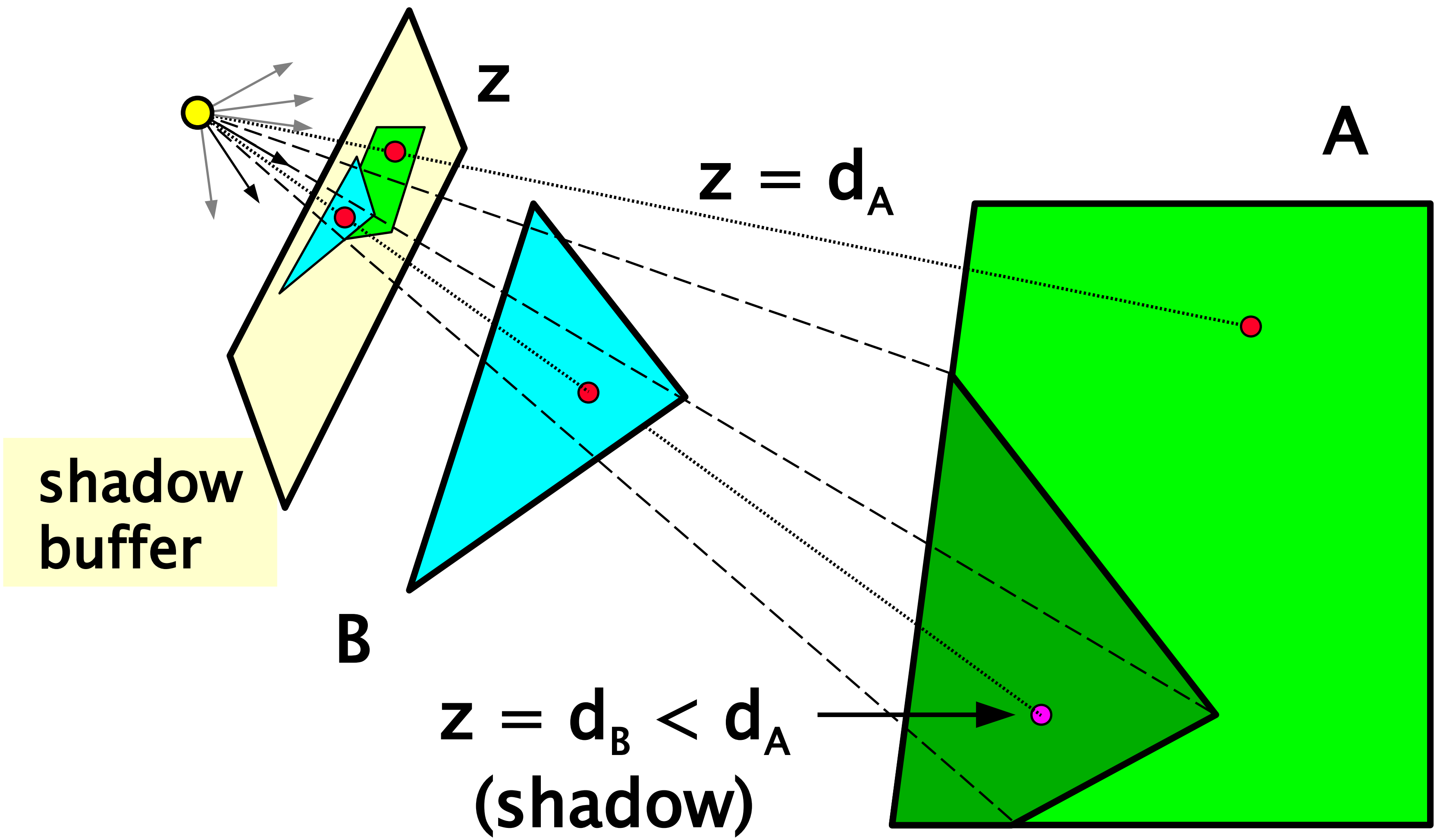
Methods

- **multiple visibility computation**
  - visibility from a **light source's viewpoint**, proper shadow representation, common visibility algorithm
  - **shadow buffer** (shadow depth-buffer)
- **shadow volumes**
  - shadow is a **3D solid**, need for intersection computation
  - shadow solid can be represented by a **BSP**
- **direct shadow computation**
  - scanline methods (scene lit from above)
  - <u>ray-based rendering</u> (ray-tracing, path-tracing)

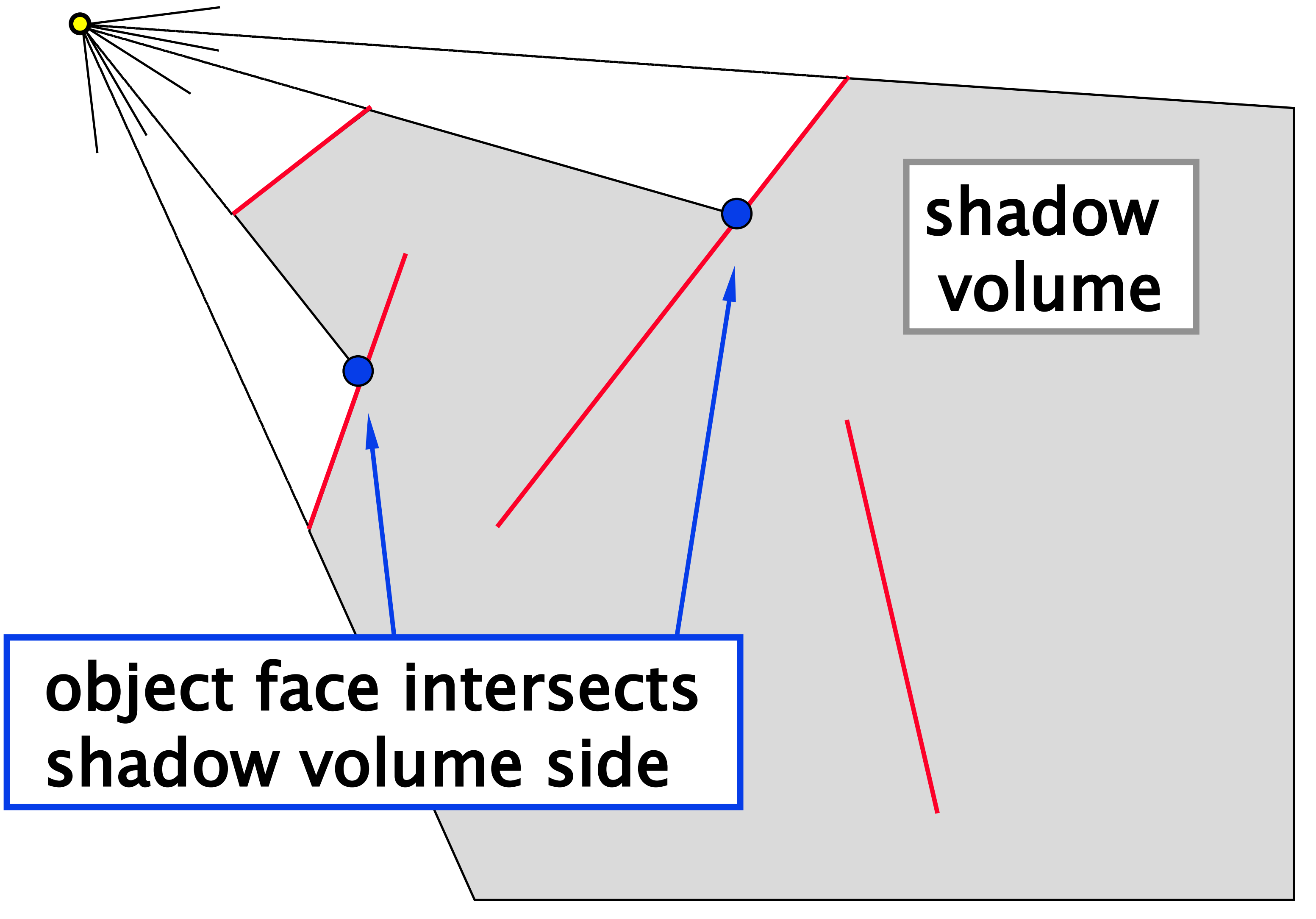
Shadow buffer (shadow map)

- ❶ **depth-buffer from a lightsource viewpoint**
  – only depths will be used (**z[x,y]** matrix)
- ❷ common visibility for **regular scene rendering**
  – pixel-oriented algorithm
  – for every displayed 3D point (pixel) there is **world-space distance** to the point light source **d**
  – in projection plane we already have **z = z[x,y]**
  – if **z < d**, current pixel is in shadow (there was different 3D point closer to the light source)
- ➡ neighbours in **z[x,y]** ⇒ better shadow accuracy

Shadow–buffer (shadow map)



Shadow volumes



Shadow volumes

shadow solid representation options:

- **set of polyhedra**
  - only <u>side faces</u> are needed
  - regular faces are processed in front-to-back order according to light source
  - individual shadow “cones“ must be joined at the end
- **BSP-representation** of shadow volume
  - BSP-representation of regular faces
  - we add virtual faces defined by the light source and <u>lit object edges</u>

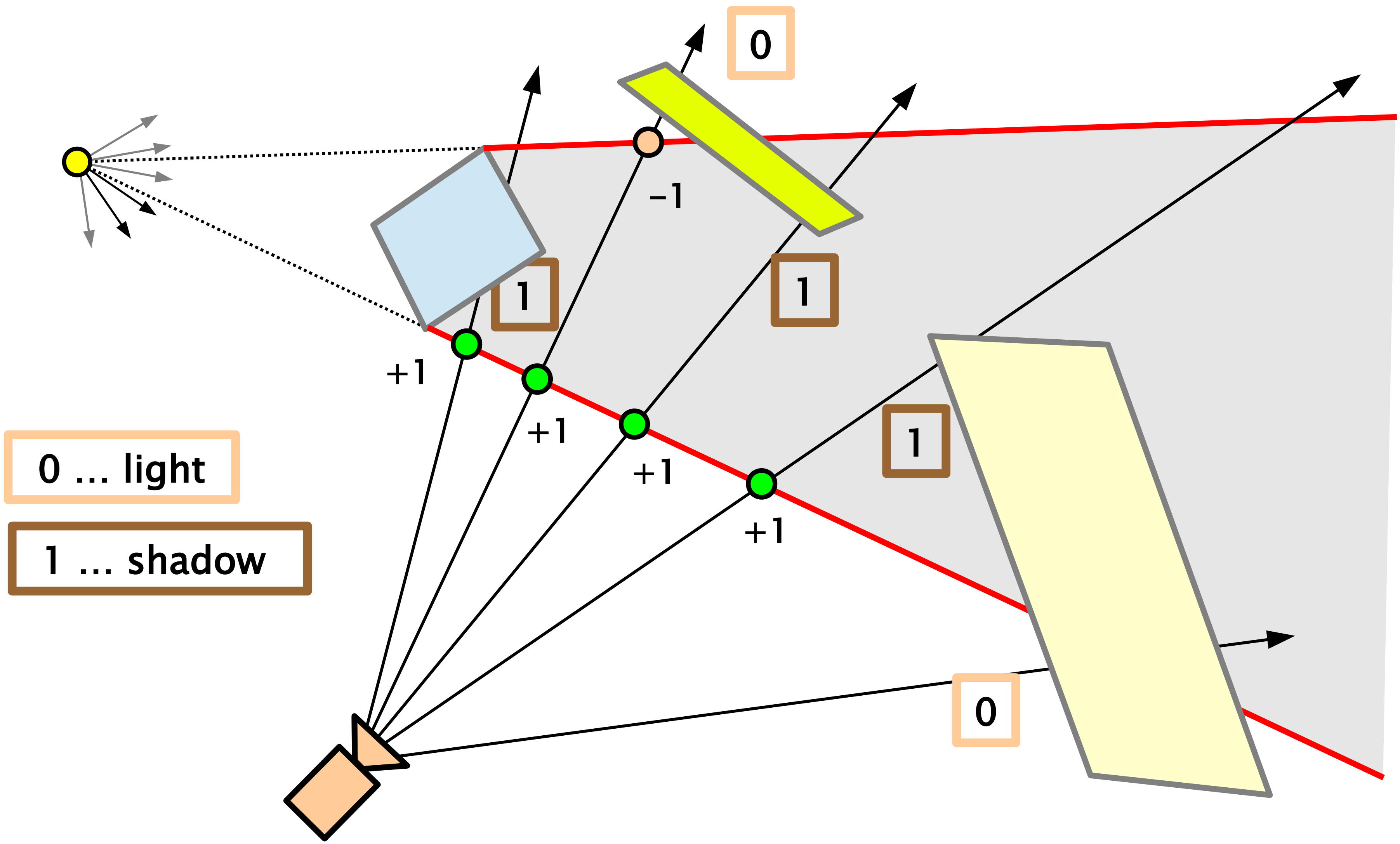
Volumetric shadows

- every lit object casts infinite shadow (set of shadowed points = “**shadow volume**“)
- **side faces** of a shadow volume are invisible (virtual) infinite quadrangles
  - ray from <u>the camera to a rendered point</u> is tested against such faces
  - GPU can rasterize these virtual faces into a “stencil buffer” and use this buffer for realtime shadowing..
- **stencil buffer** defines **lit** and **shadowed** part of a scene
  - the whole process must be iterated for more light sources

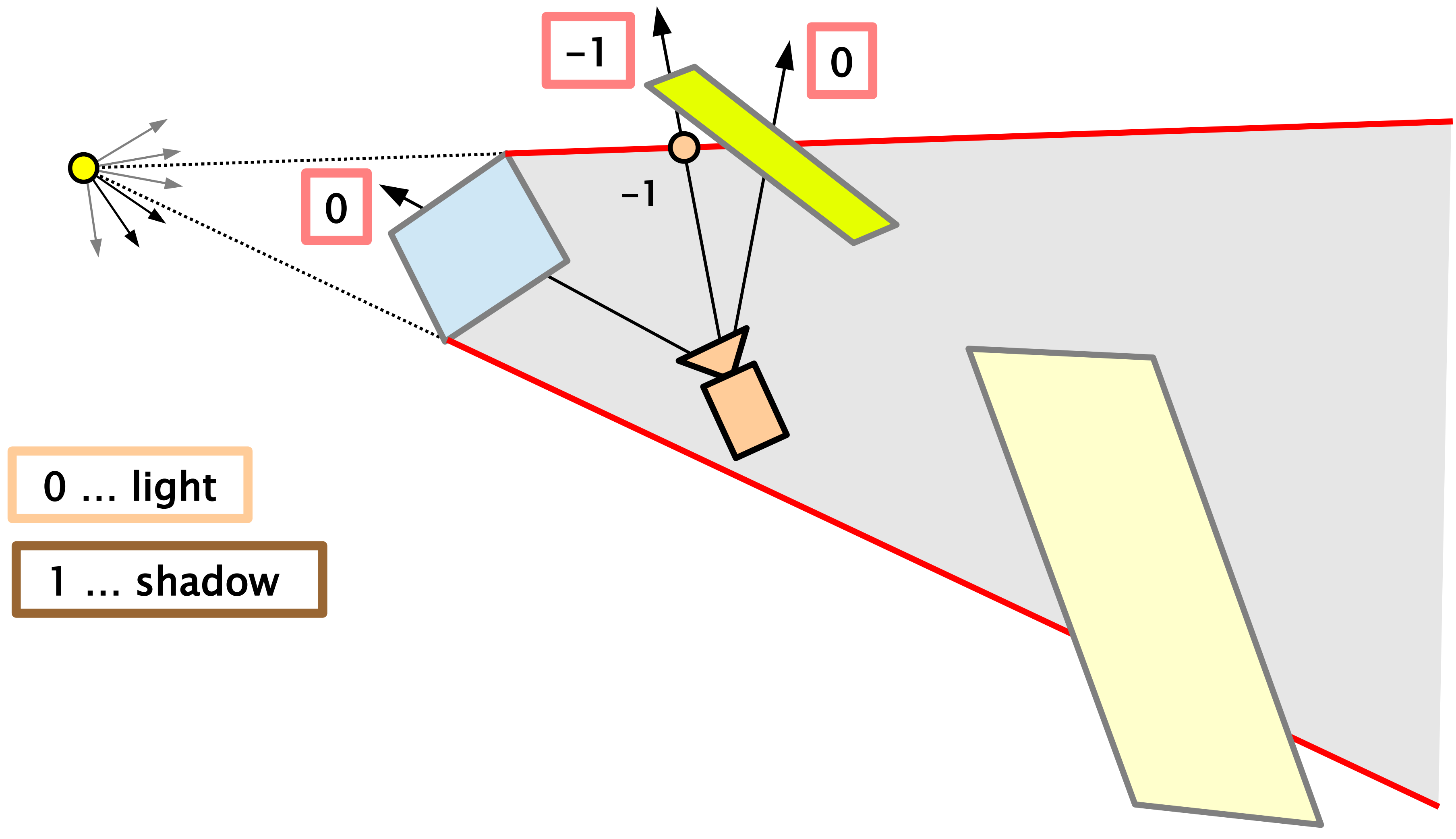
Volumetric shadows I

- common **first phase** – regular <u>visible scene</u> is drawn
  - depth-buffer is updated, lighting is set to “ambient”
- (virtual) side faces of a shadow body – **forward** or **backward**
  - virtual faces do not update **depth-buffer** (but are tested against it!)
- **second phase** – only **virtual faces** are processed:
  - **forward visible** face **increments** stencil value
  - **backward visible** face **decrements** stencil value
- **third phase** – **lit** parts of the scene have **zero stencil value** (contribution of the light source must be added)

Shadow volumes I



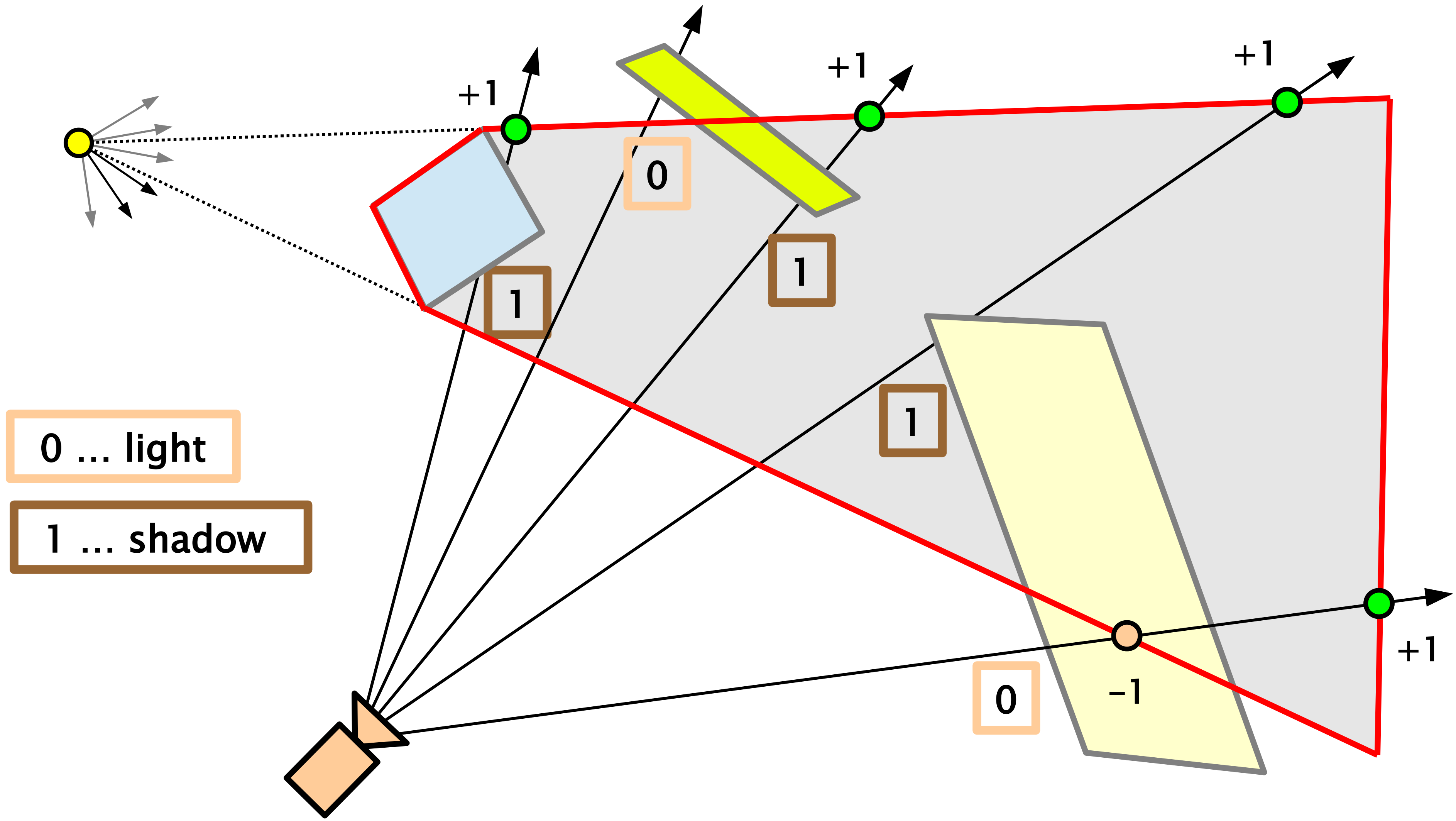
Shadow volumes I – flaw



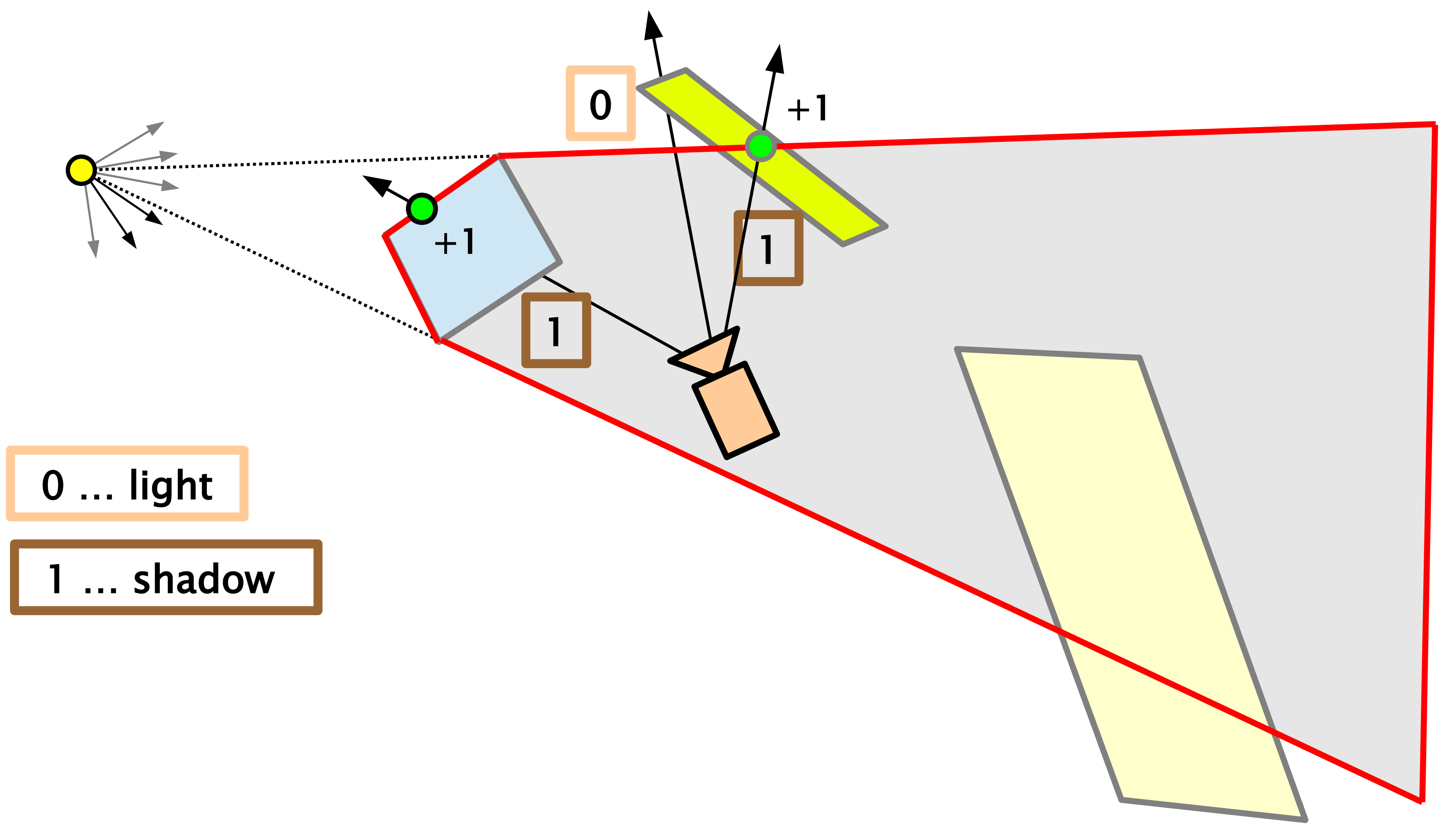
Shadow volumes II

- camera can be **anywhere** (even in a shadow)
  - shadow volume are perfectly closed by **“caps”**
  - one additional “cap” is a lit part of an object, the second one is in infinity
- **second phase –** virtual side faces and “caps” are processed
  - **forward invisible** face **decrements** stencil value
  - **backward invisible** face **increments** stencil value
- **third phase – lit** parts of the scene have **zero stencil value** (contribution of the light source must be added)

Shadow volumes II



Shadow volumes II – correct



Vertices in infinity

- side faces and “caps” have **infinite vertices**
  - more distant from a camera than anything else
- projection of [**x**, **y**, **z**, **1**] to infinity: [**x**, **y**, **z**, **0**]
- **projection matrix** with value **far = ∞**:

$$A = \frac{2n}{r-l} \qquad B = \frac{r+l}{r-l} \qquad C = \frac{2n}{t-b} \qquad D = \frac{t+b}{t-b}$$

$$M(n,\infty,r,l,t,b) = \begin{bmatrix} A & 0 & 0 & 0 \\ 0 & C & 0 & 0 \\ -B & -D & 1 & 1 \\ 0 & 0 & -2n & 0 \end{bmatrix}$$

Projection of infinite points

- projection of **regular 3D point** (including w-division):

$$[\,x,\ y,\ z,\ 1\,]\cdot M \;=\; \left[\,\frac{x}{z}A-B,\ \frac{y}{z}C-D,\ 1-\frac{2n}{z}\,\right]$$

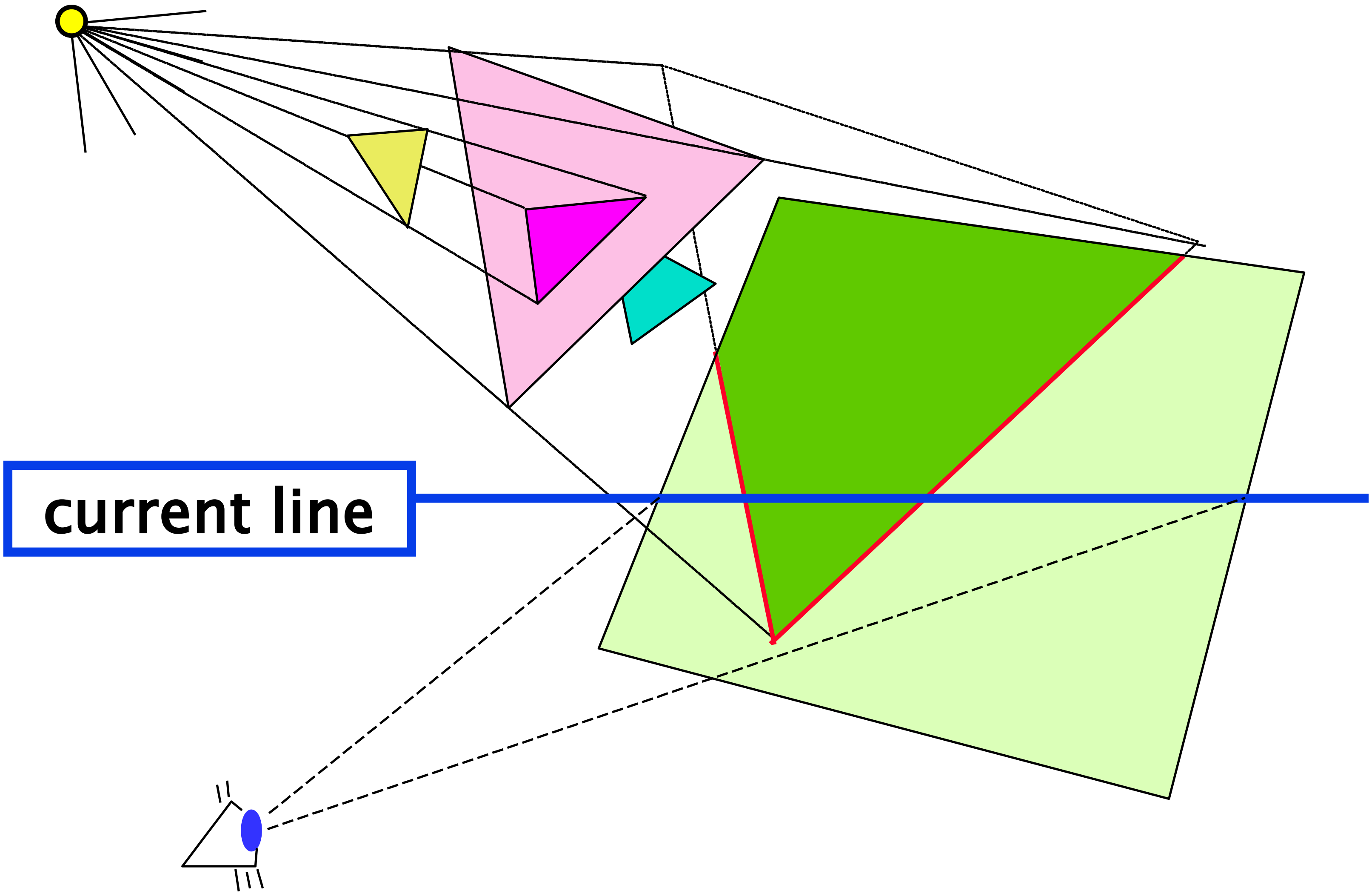
- projection of **infinite (extrinsic) point**:

$$[\,x,\ y,\ z,\ 0\,]\cdot M \;=\; \left[\,\frac{x}{z}A-B,\ \frac{y}{z}C-D,\ 1\,\right]$$

Scanline algorithm

- 3D scene lit from **above**
  - the same direction as scanline order
- potentially shadowers (edges) are projected to **currently rendered face**
  - these edges were already processed (or are currently processed)
  - only lit edges (parts) are used
- further improvement (Bouknight a Kelley, 1970)
  - preprocessing (projection froma a light source) gives us an estimate, which are able to shadow at all

Scanline algorithm



References

- **J. Foley, A. van Dam, S. Feiner, J. Hughes**: ***Computer Graphics, Principles and Practice***, 745-753
- **Jiří Žára a kol.**: ***Počítačová grafika***, principy a algoritmy, 361-363