Computer Graphics 18 - Review

Tom Thorne

Slides courtesy of Taku Komura
www.inf.ed.ac.uk/teaching/courses/cg

Graphics pipeline

Geometry

- Transformation
- Perspective projection
- Hidden surface removal

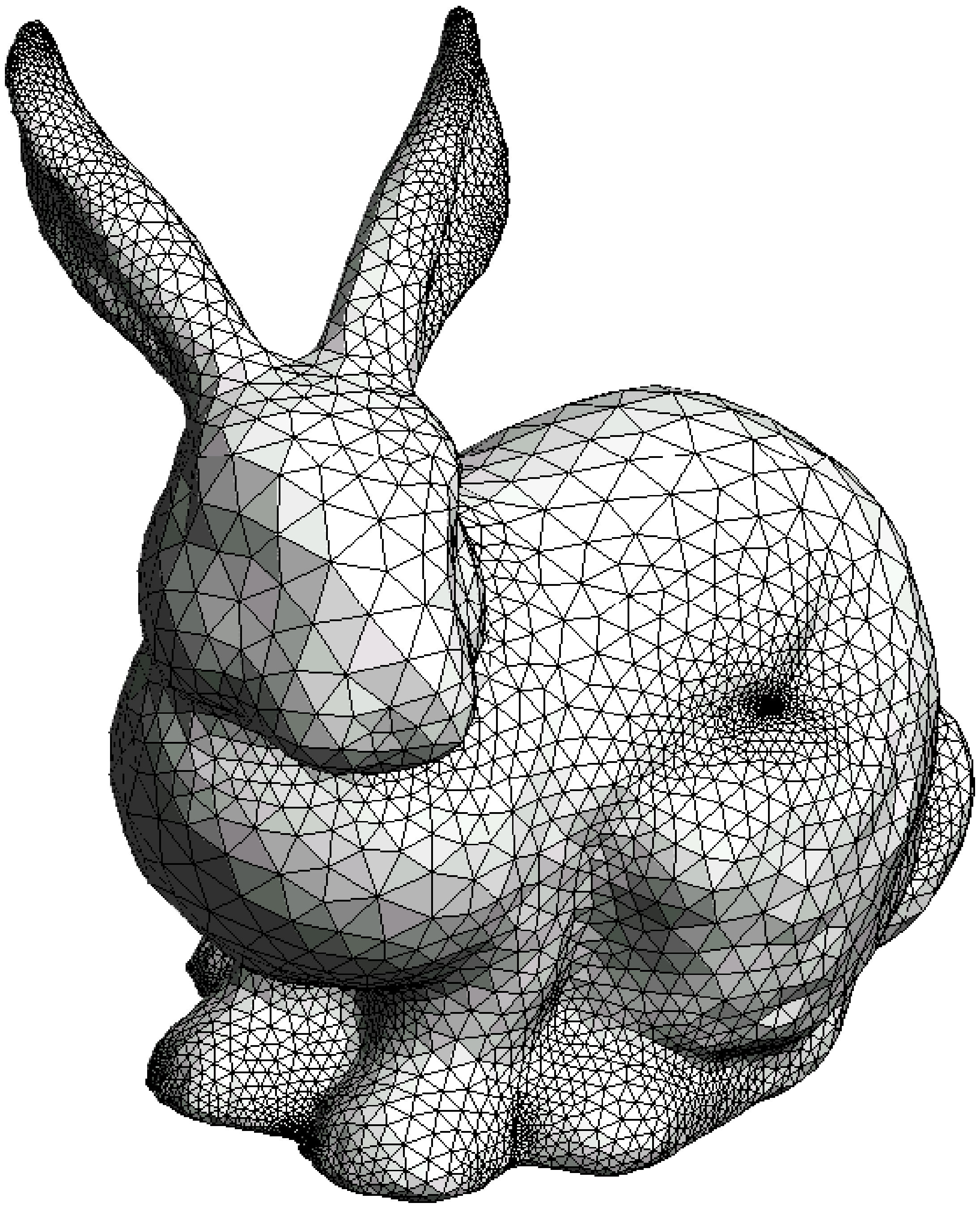
Shading and lighting

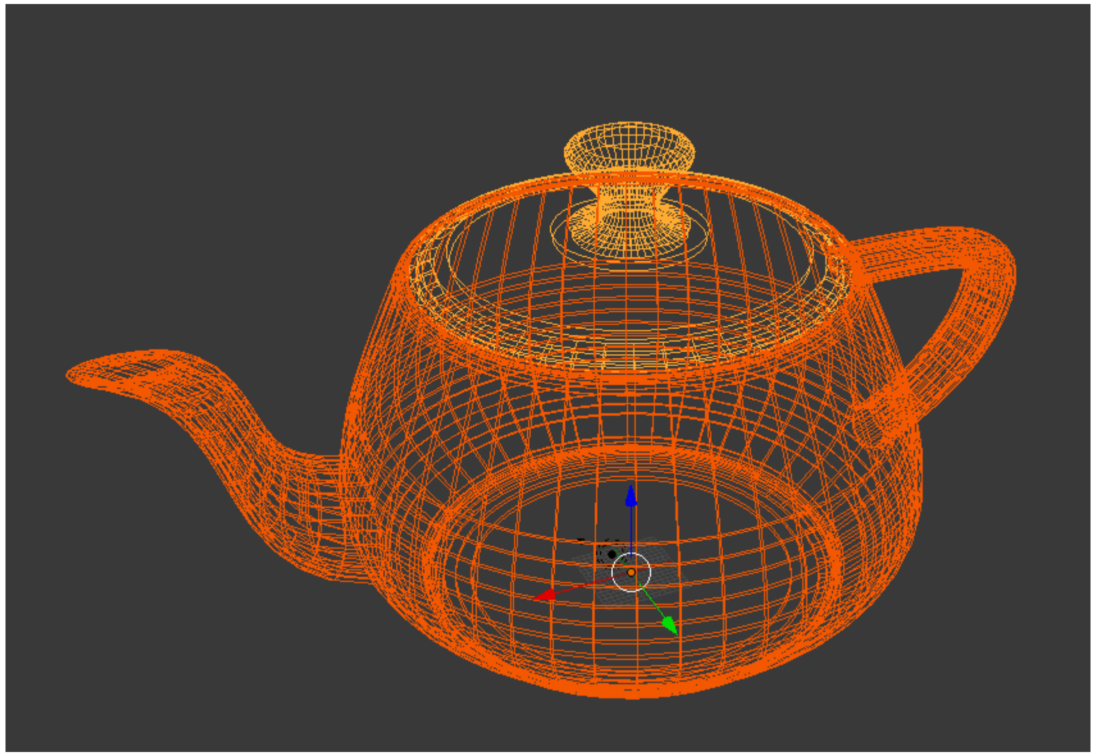
- Reflections
- Shadows

Rasterisation

- Anti aliasing
- Texture mapping
- Bump mapping
- Ambient occlusion

Mesh structures



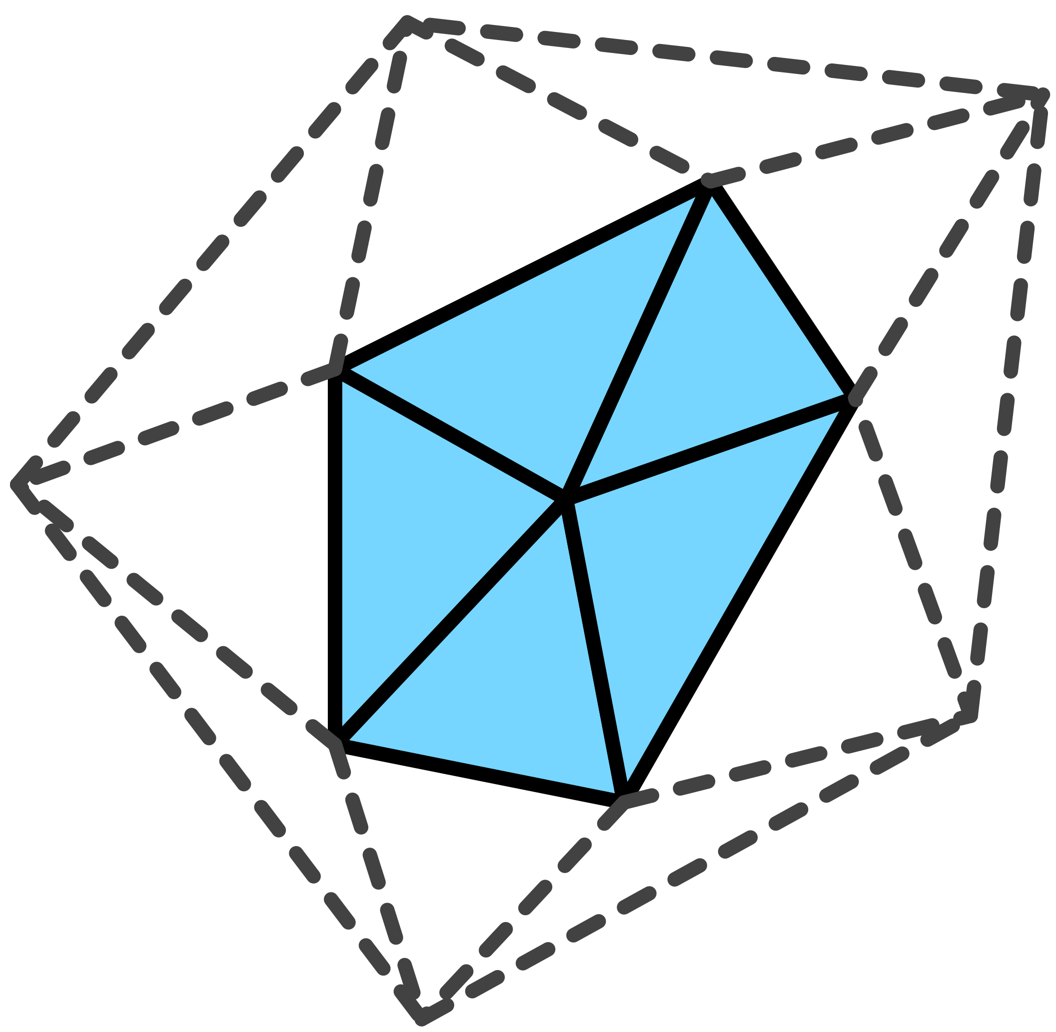


Objects represented as a set of polygons

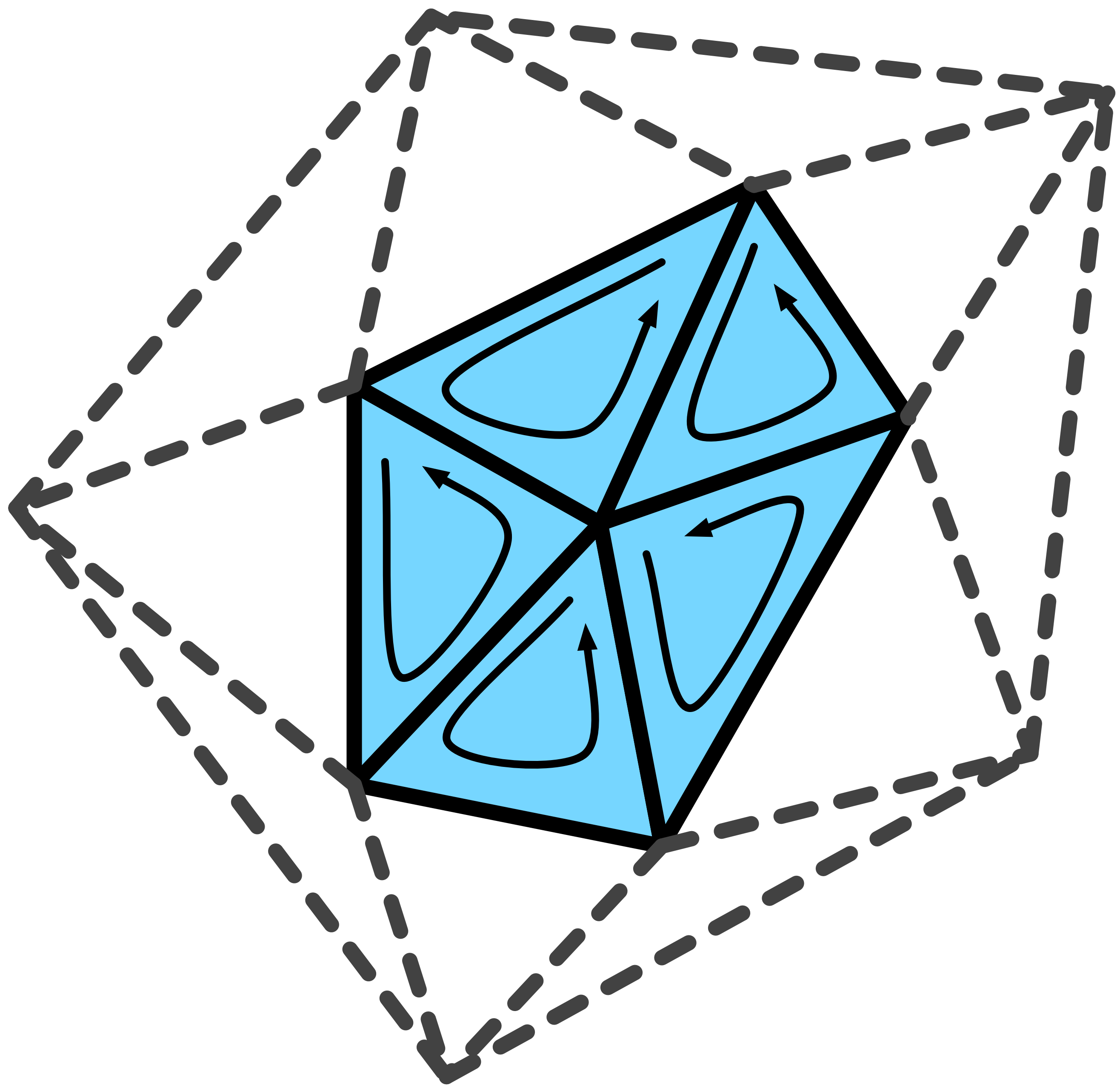
Mesh topology

Manifolds:

- All edges belong to two triangles
- All vertices have a single continuous set of triangles around them



Orientation



Vertexes in triangle list stored in counter clockwise order

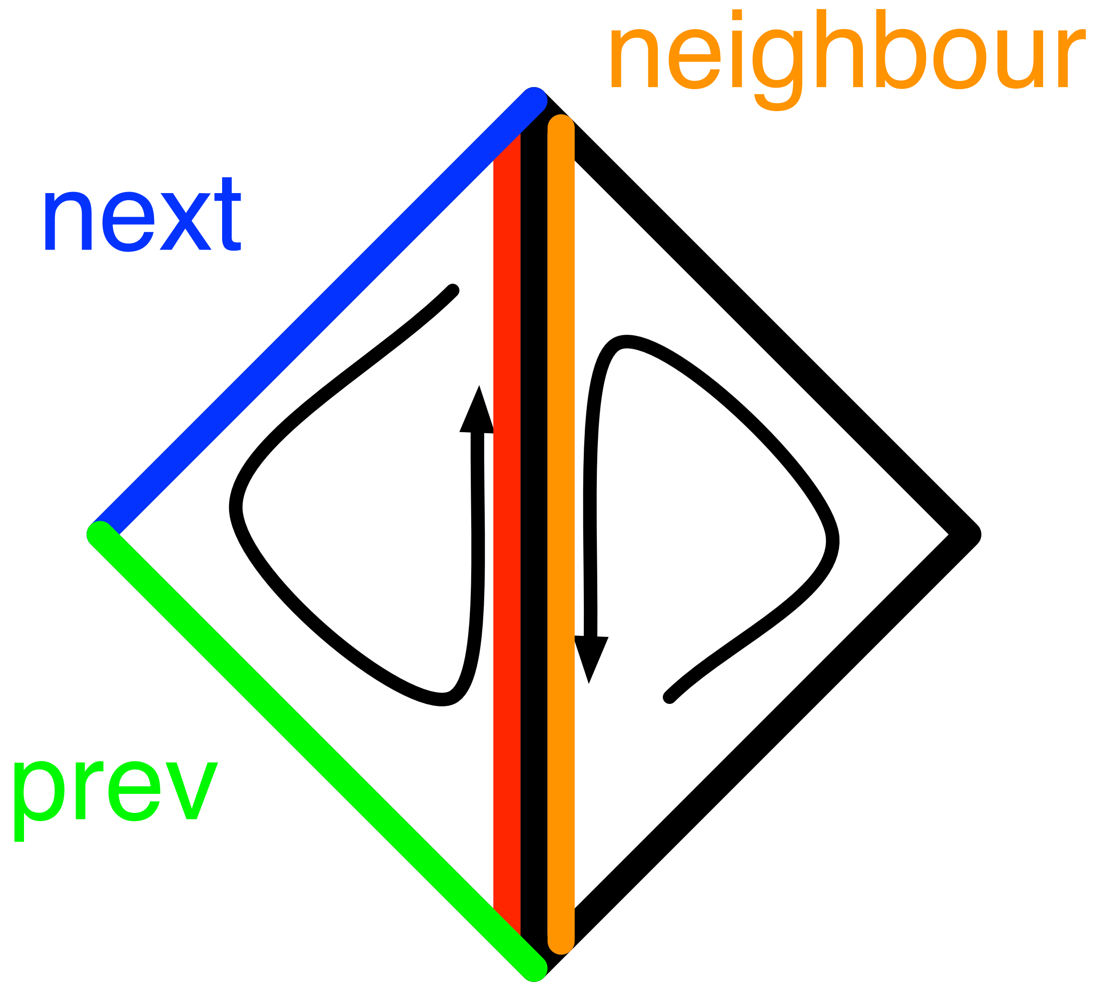
Directed edge data structure

Vertices

x_1	y_1	z_1	e_r
x_2	y_2	z_2	e_s
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Edges

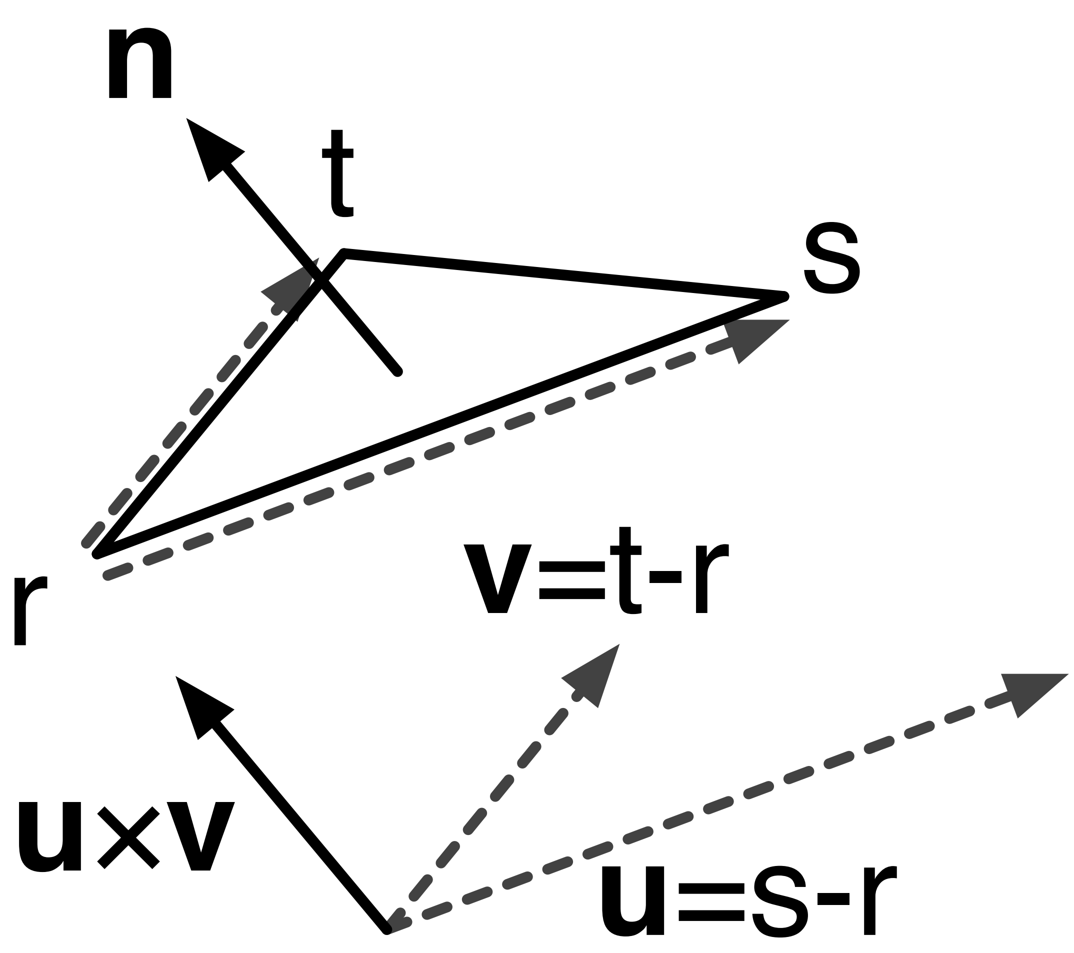
v_a	v_b	$e_{neighbour}$	e_{next}	e_{prev}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

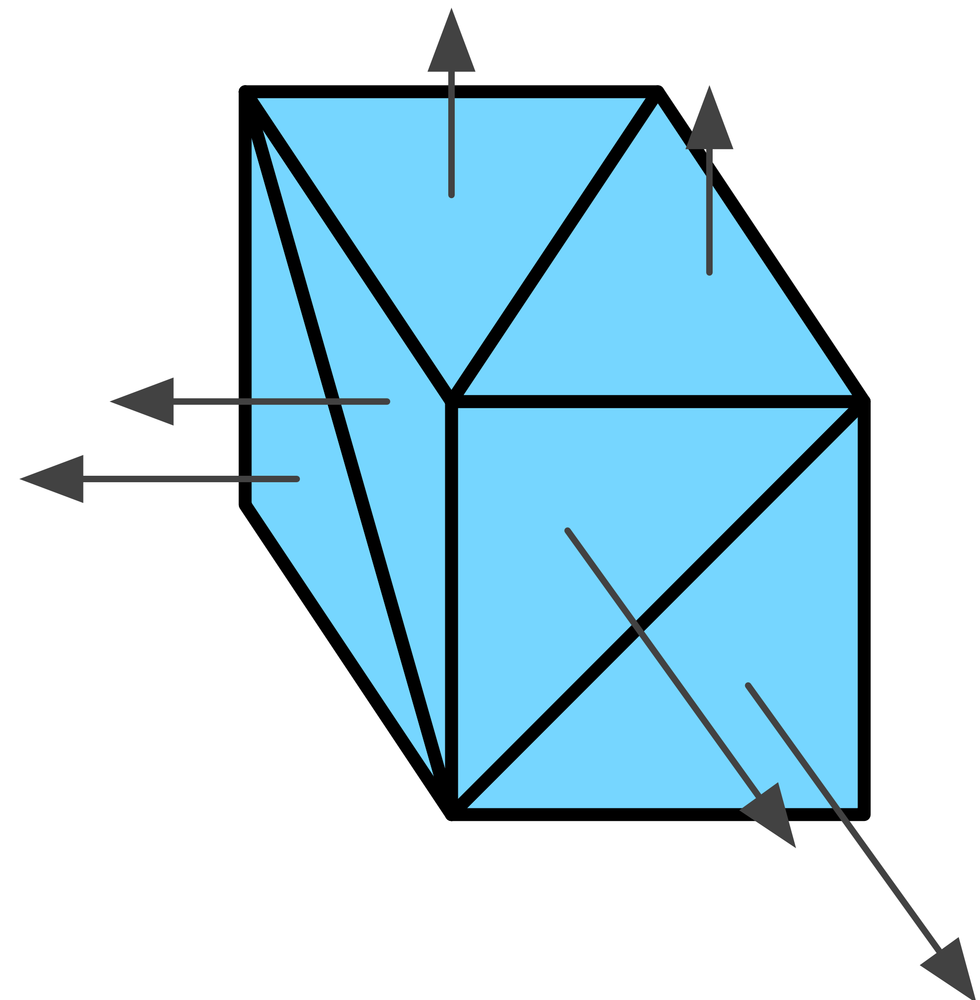


Surface normals

Calculation:

$$\boldsymbol{n} \quad = \quad (t - r) \times (s - r)$$





3D translation

Very simple to extend 2D case to 3D:

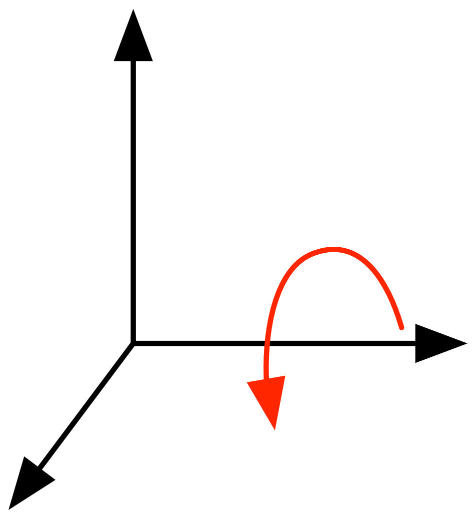
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

3D scaling

Very simple to extend 2D case to 3D:

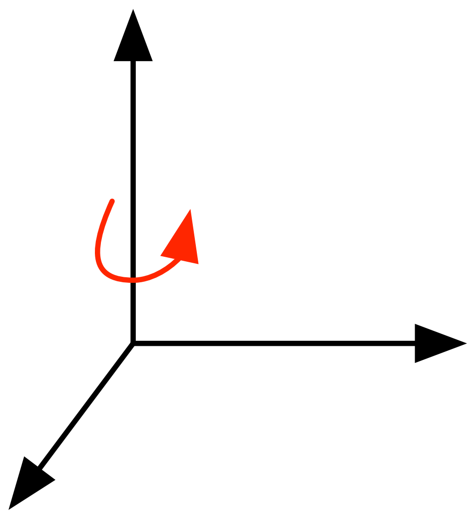
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

3D rotation - X axis



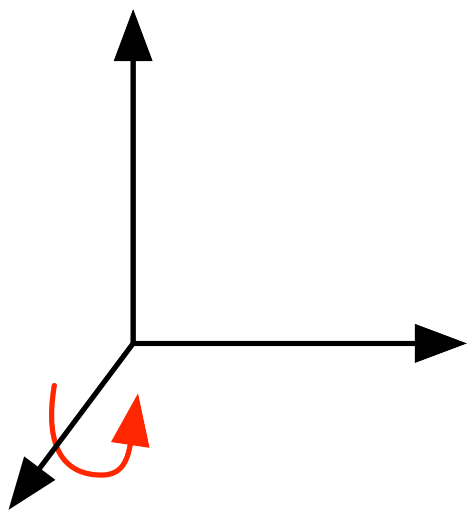
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

3D rotation - Y axis



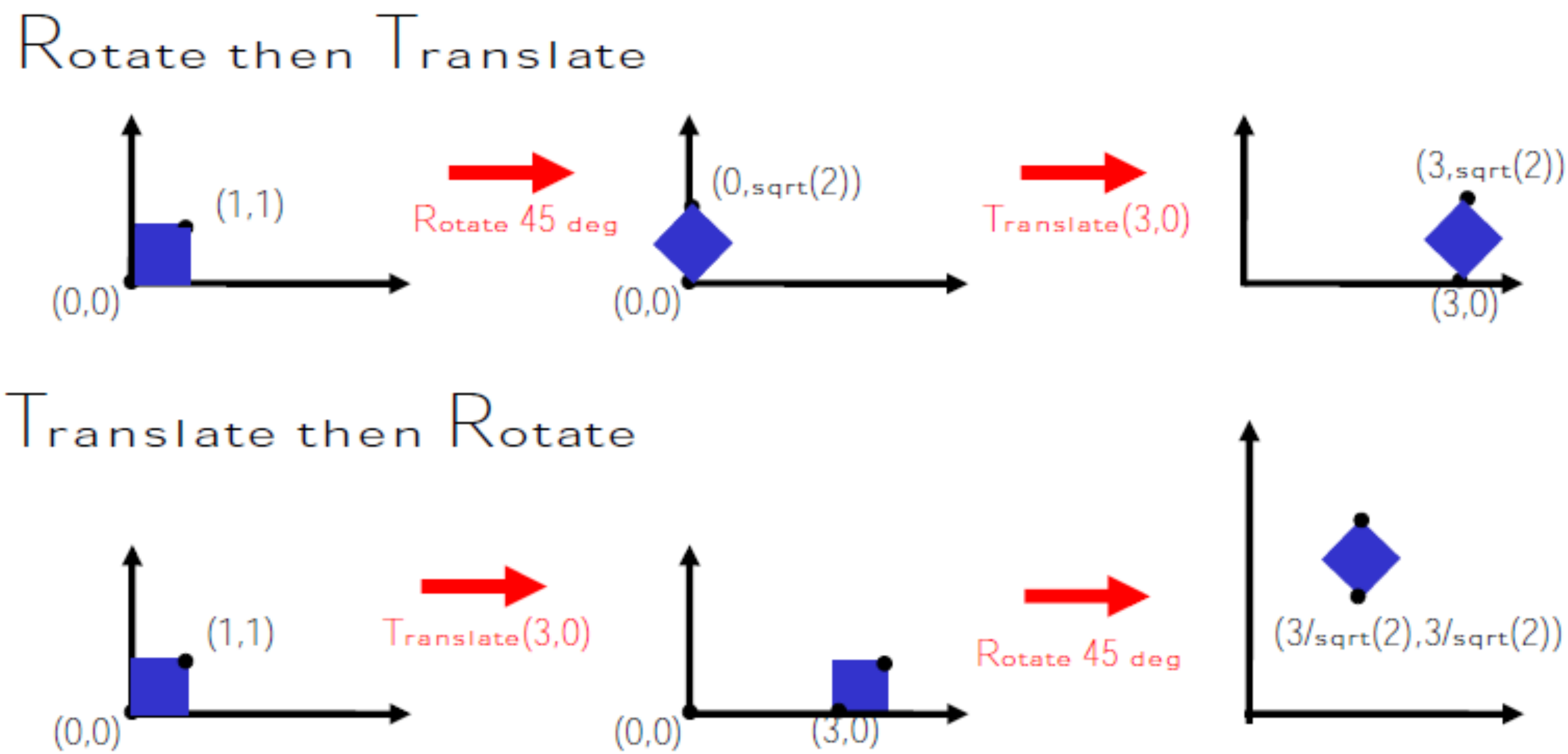
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

3D rotation - Z axis



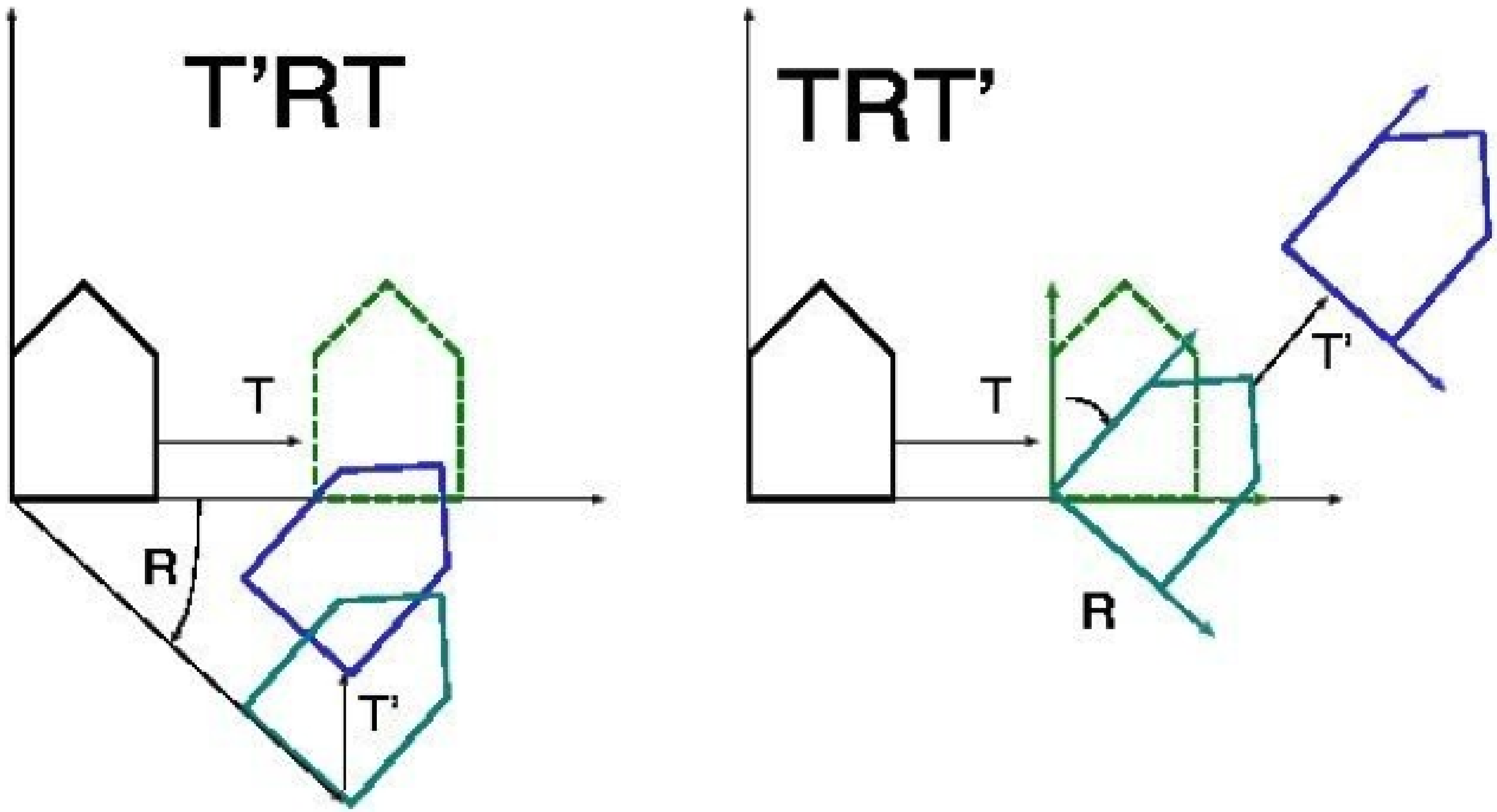
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Example 2



Rotate then Translate
(1,1)
(0,0)
Rotate 45 deg
(0,sqrt(2))
(0,0)
Translate(3,0)
(3,sqrt(2))
(3,0)
Translate then Rotate
(1,1)
(0,0)
Translate(3,0)
(0,0)
(3,0)
Rotate 45 deg
(3/sqrt(2),3/sqrt(2))

Order of multiplication

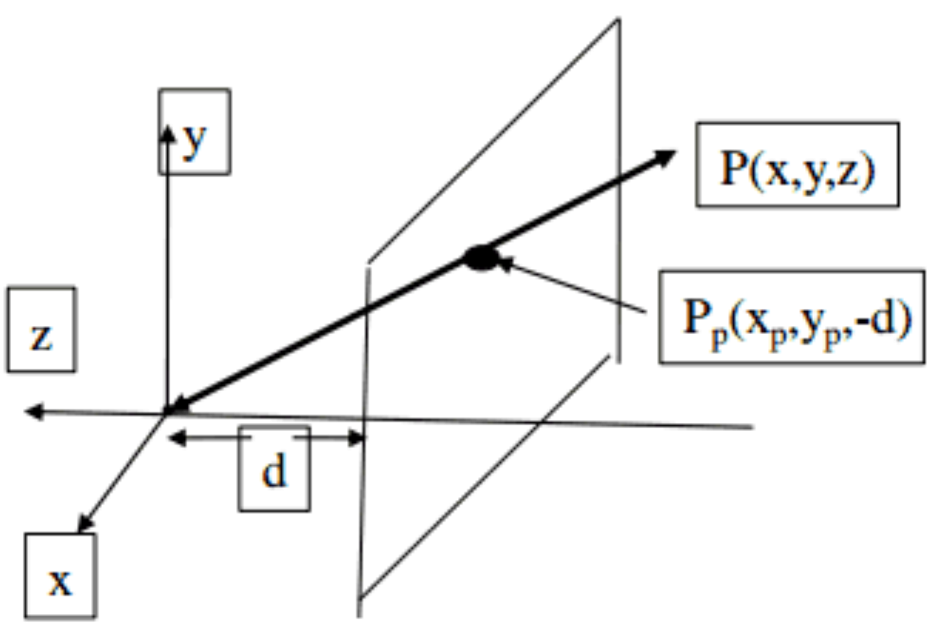


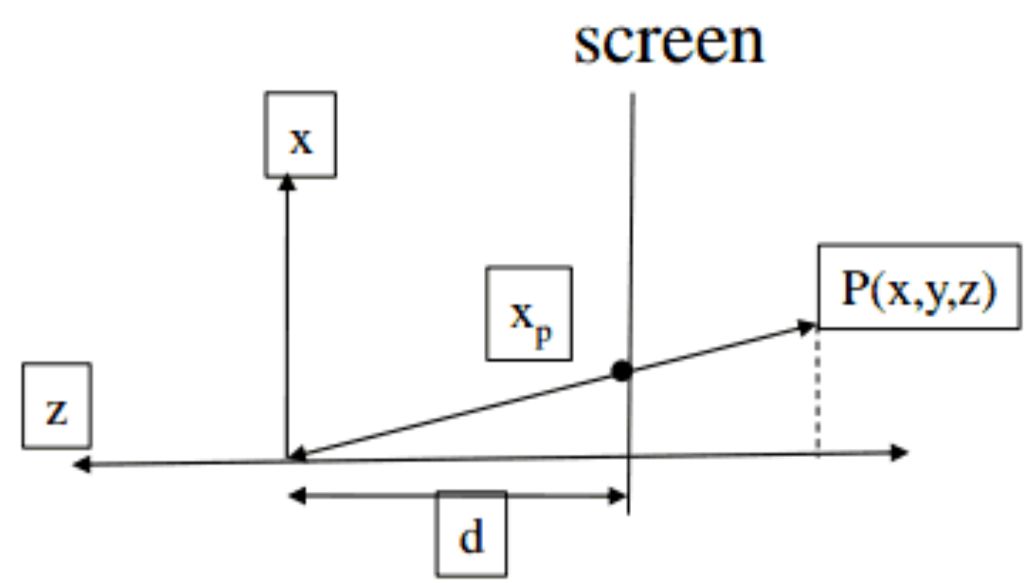
Perspective projection - simple case

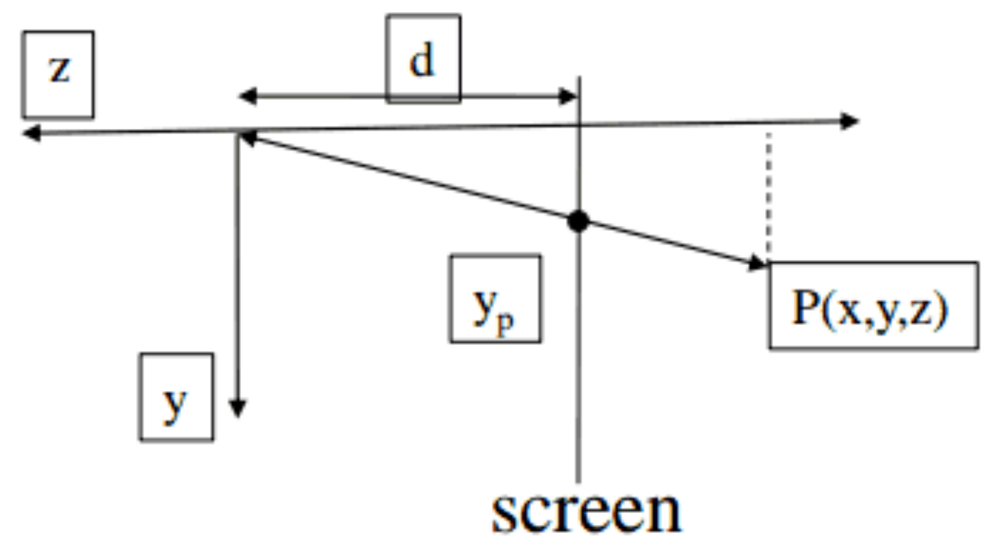
From similar triangles:

$$\frac{x_p}{d} = \frac{x}{-z}; \ \frac{y_p}{d} = \frac{y}{-z}$$

$$x_p = \frac{d \cdot x}{-z} = \frac{x}{-z/d}; \ \ y_p = \frac{d \cdot y}{-z} = \frac{y}{-z/d}$$





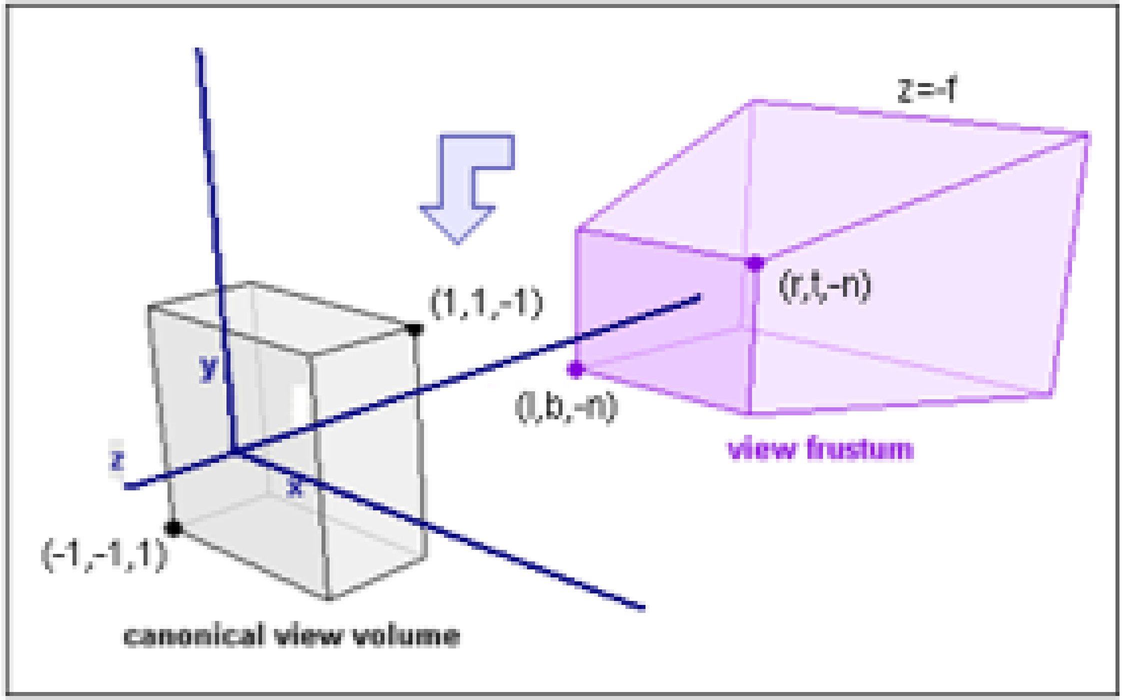


Transforming the view frustum

The frustum is defined by a set of parameters, l, r, b, t, n, f:

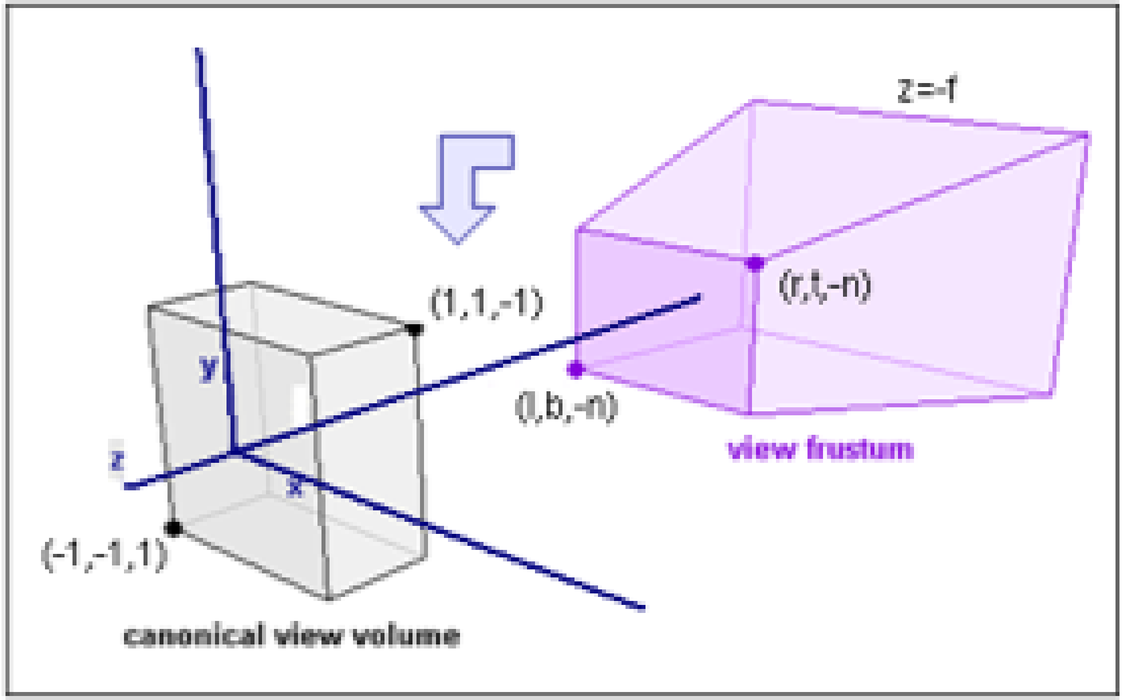
l	Left x coordinate of near plane
r	Right x coordinate of near plane
b	Bottom y coordinate of near plane
t	Top y coordinate of near plane
n	Minus z coordinate of near plane
f	Minus z coordinate of far plane

With $0 < n < f$.



Projection summary

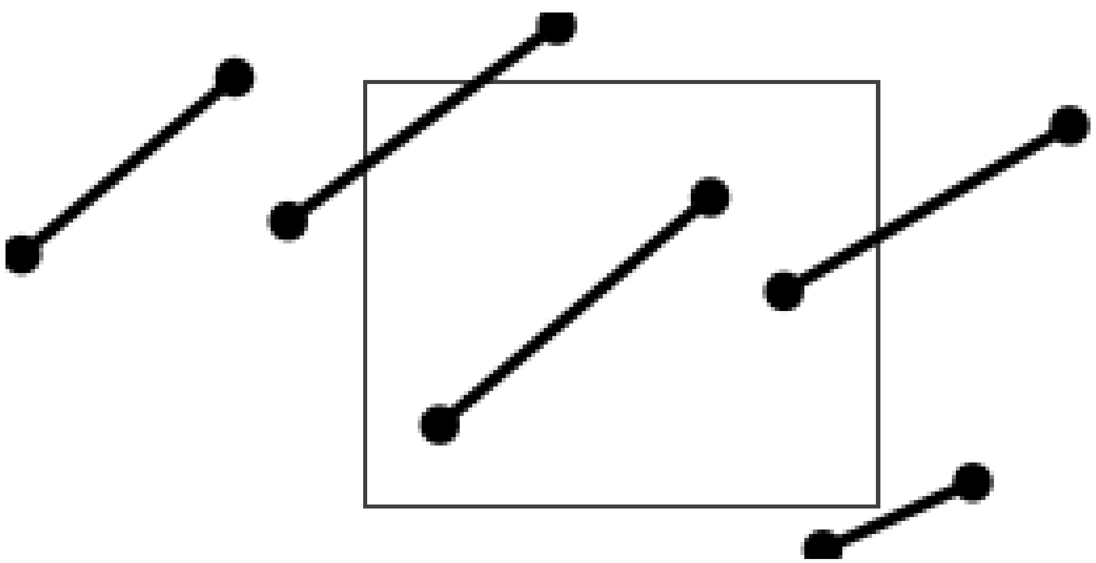
- Parallel and perspective projection
- Projection matrices transform points to 2D coordinates on the screen
- Canonical view volumes can be used for clipping

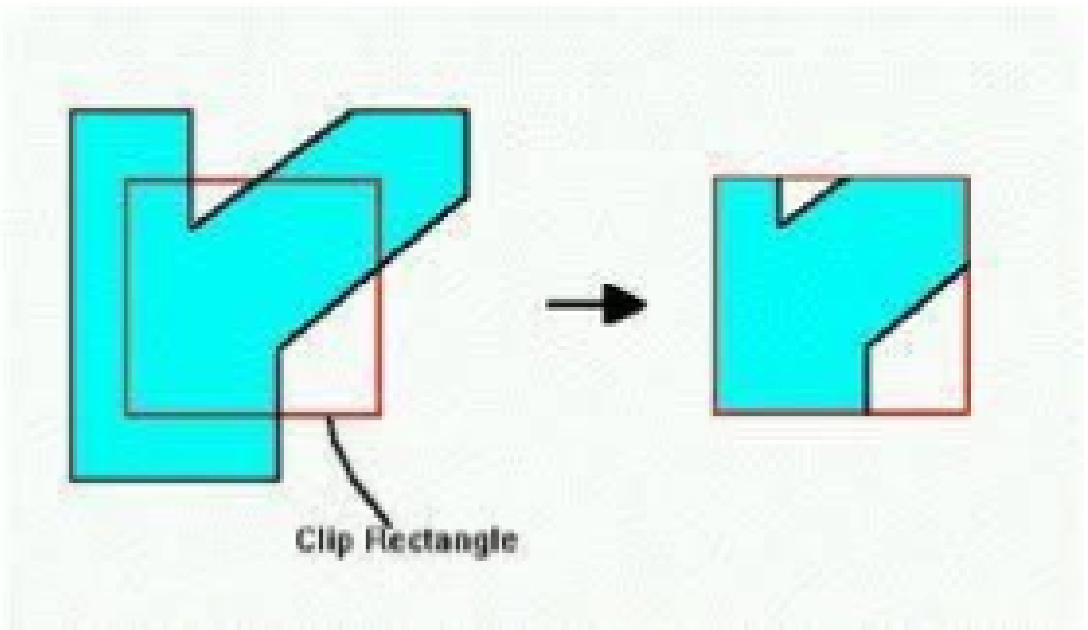


Clipping

They may intersect the canonical view volume, then we need to perform clipping:

- Clipping lines (Cohen-Sutherland algorithm)
- Clipping polygons (Sutherland-Hodgman algorithm)

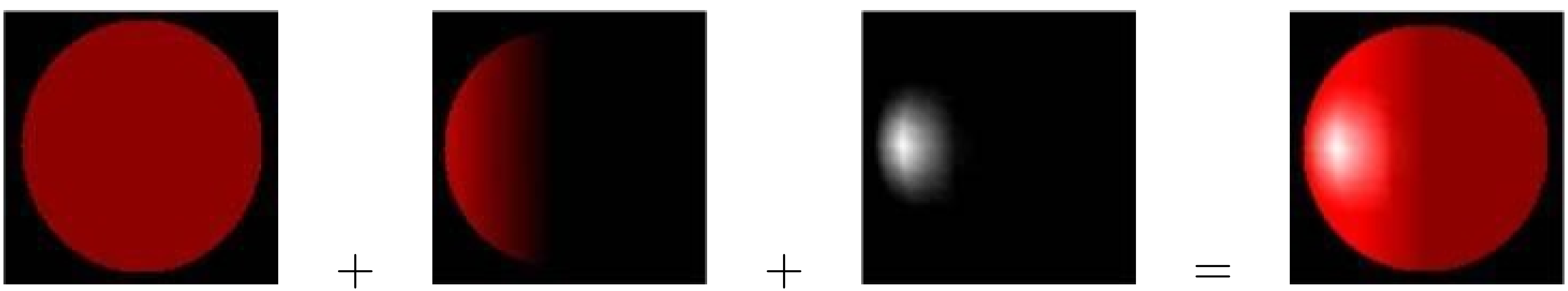




Combined lighting models

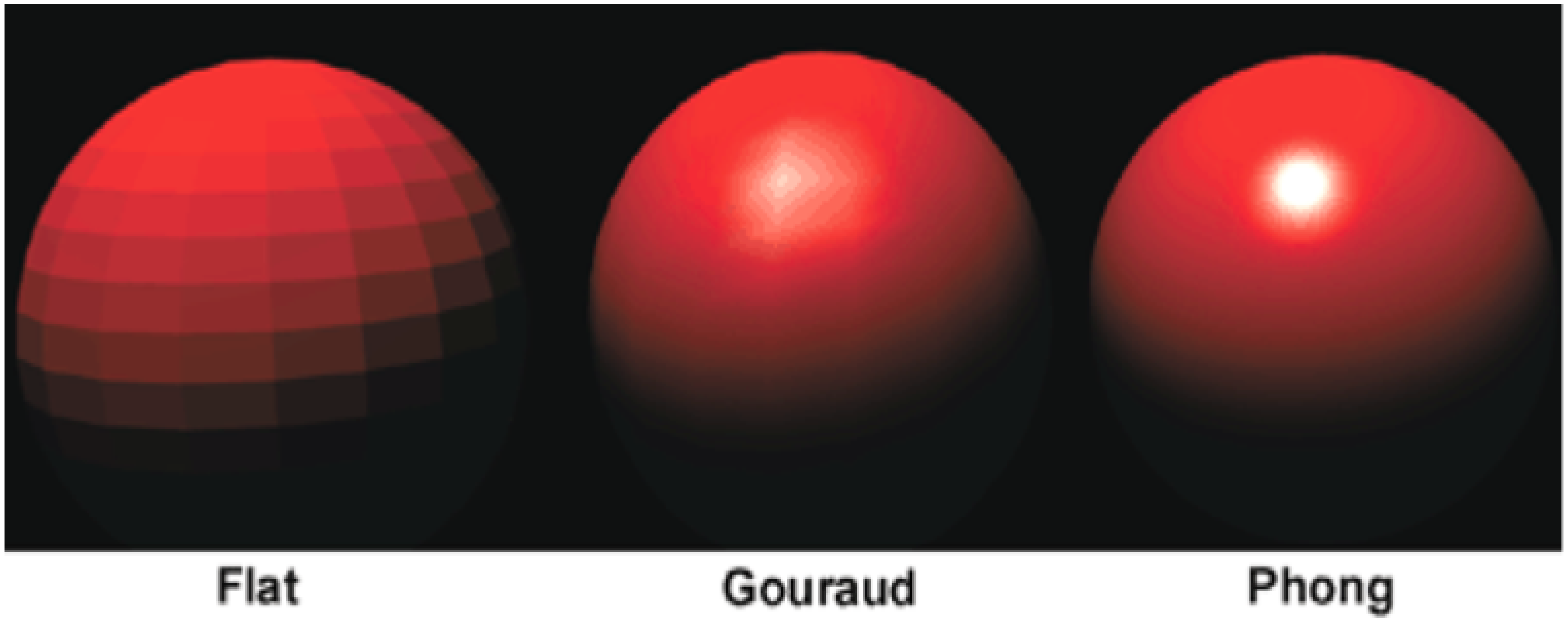
Combining ambient, diffuse and specular highlights gives the Phong Illumination model

$$I \quad = \quad I_a k_a + I_p(k_d \cos\theta + k_s \cos^n \alpha)$$



Ambient + Diffuse + Specular = I

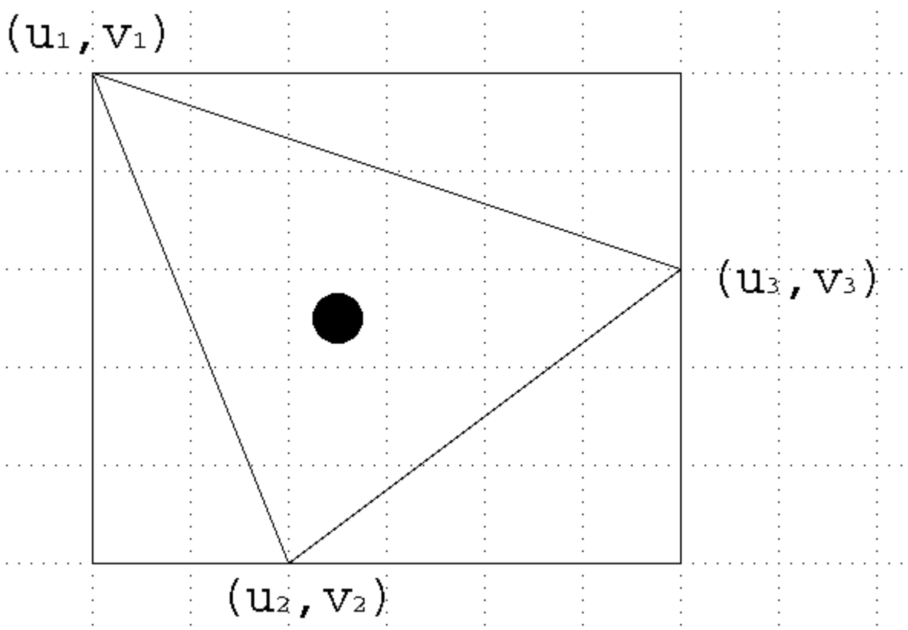
Phong example



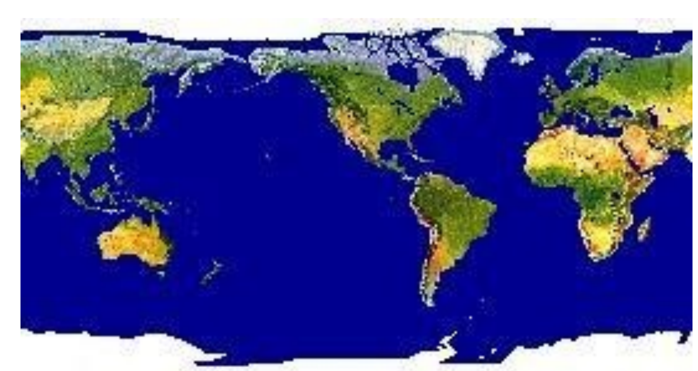
Texture mapping

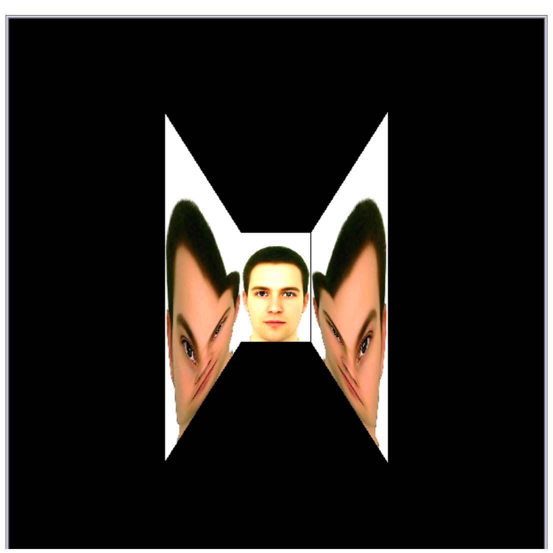
- Barycentric coordinates

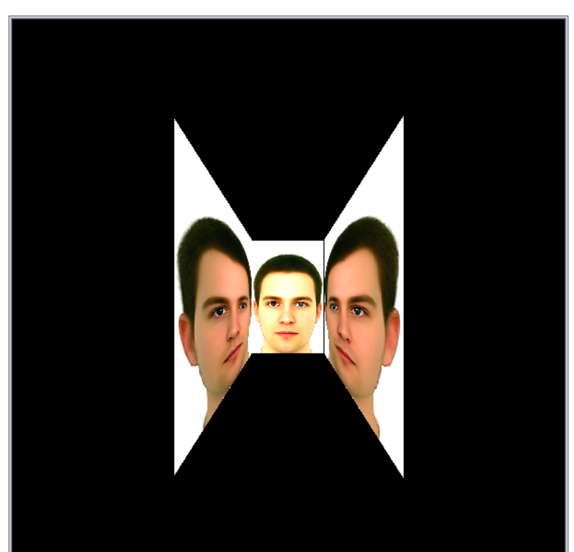
- uv mappings



$$u = \alpha\, u_1 + \beta\, u_2 + \gamma\, u_3$$
$$v = \alpha\, v_1 + \beta\, v_2 + \gamma\, v_3$$



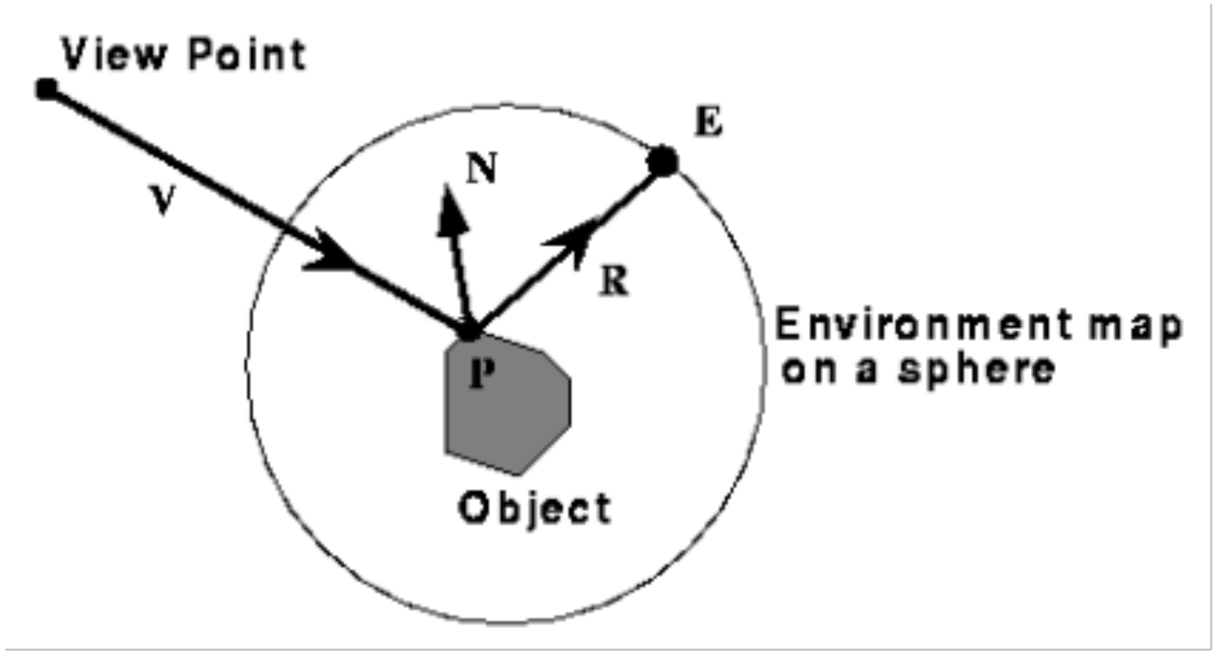


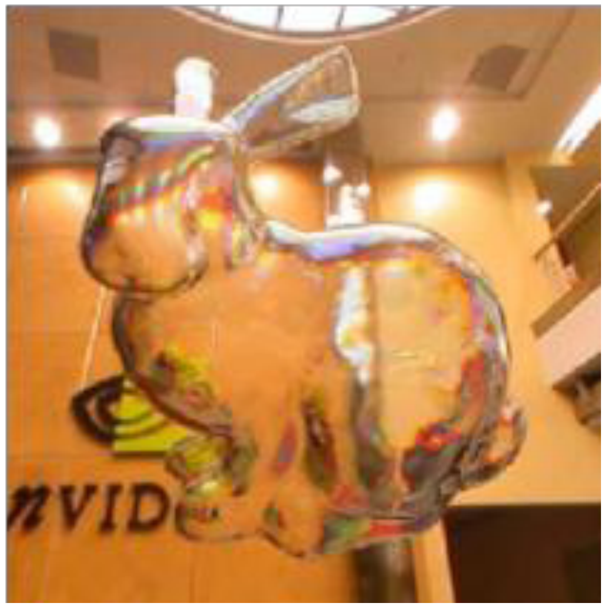


Environment Mapping

- Simple yet powerful method to generate reflections

- Simulate reflections by using the reflection vector to index a texture map at "infinity".





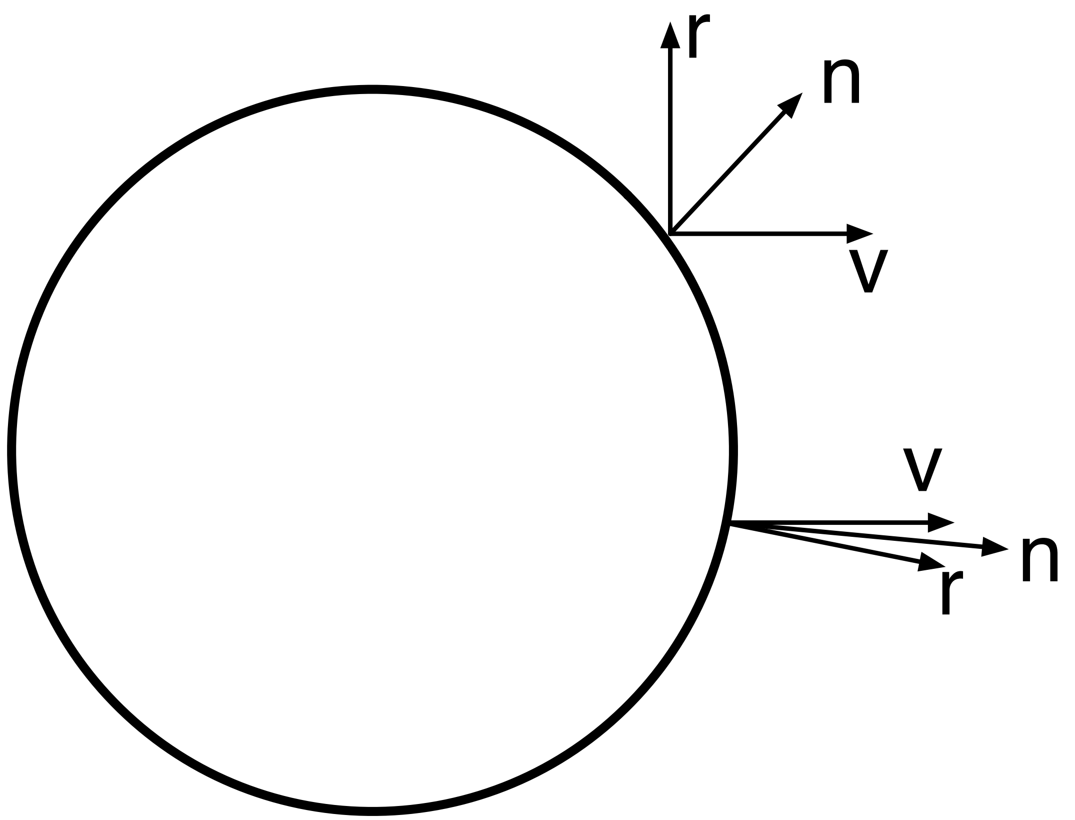
The original environment map was
a sphere [by Jim Blinn '76]

Indexing the sphere map

- Assume that v is fixed at (0,0,1)
- An un-normalised normal vector n is then:

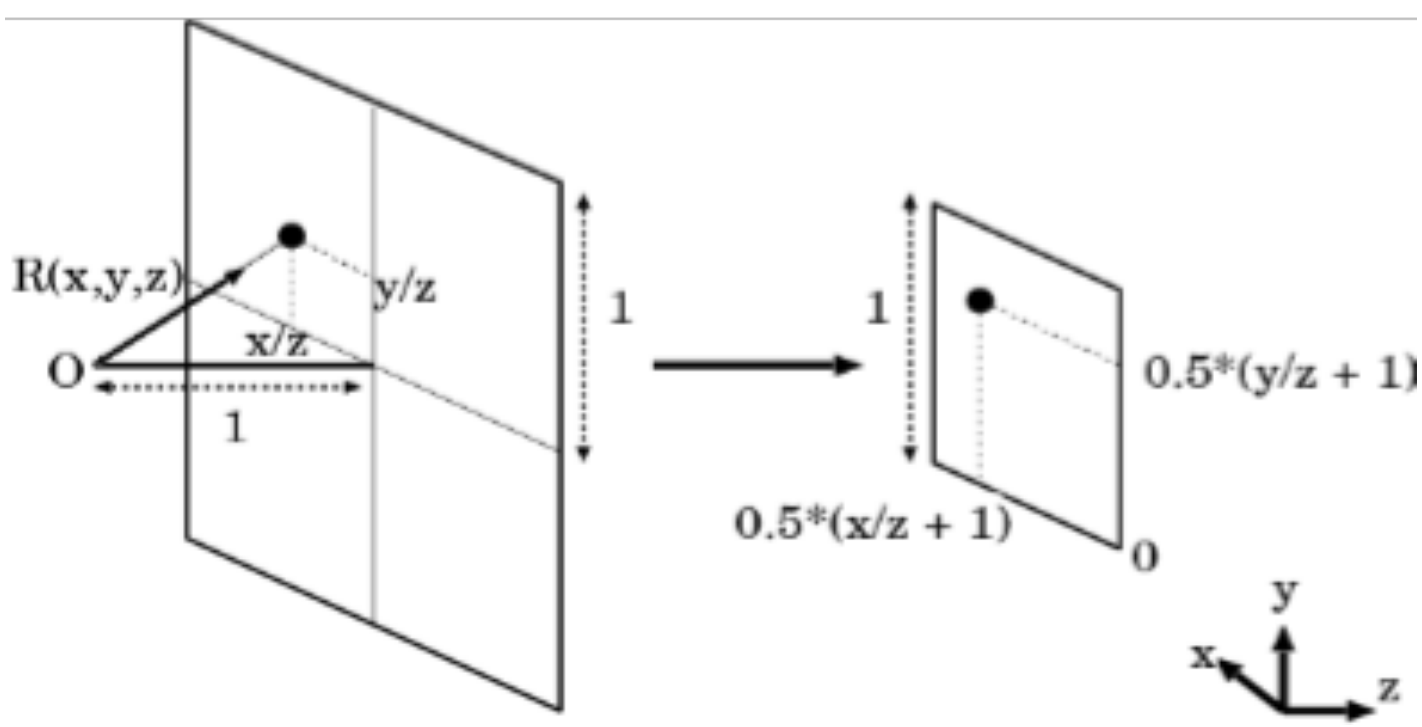
$$n = r + v$$
$$= (r_x, r_y, r_z + 1)$$

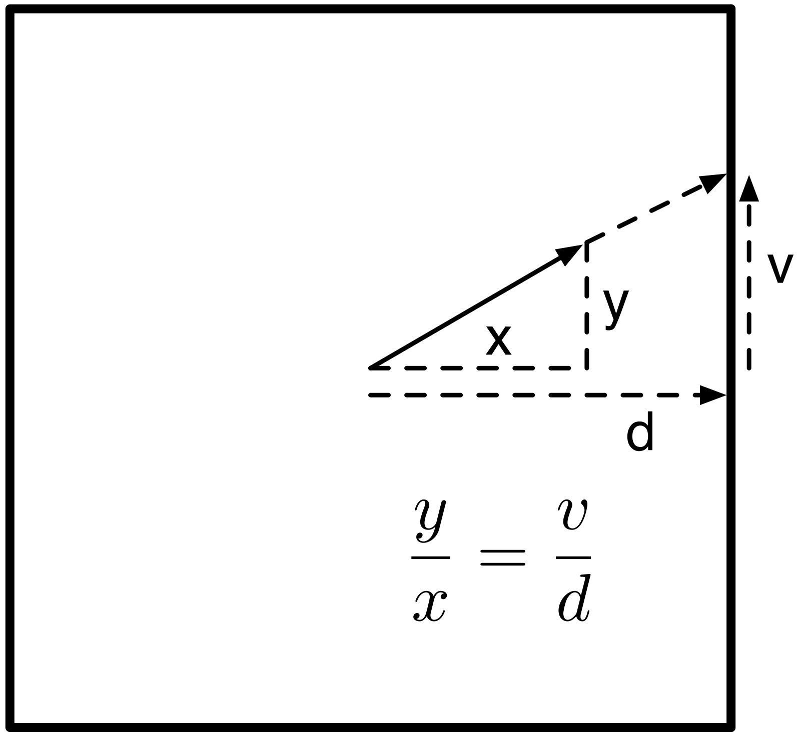
$$\overline{n} = (\frac{r_x}{m}, \frac{r_y}{m}, \frac{r_z + 1}{m})$$
$$m = \sqrt{r_x^2 + r_y^2 + (r_z + 1)^2}$$



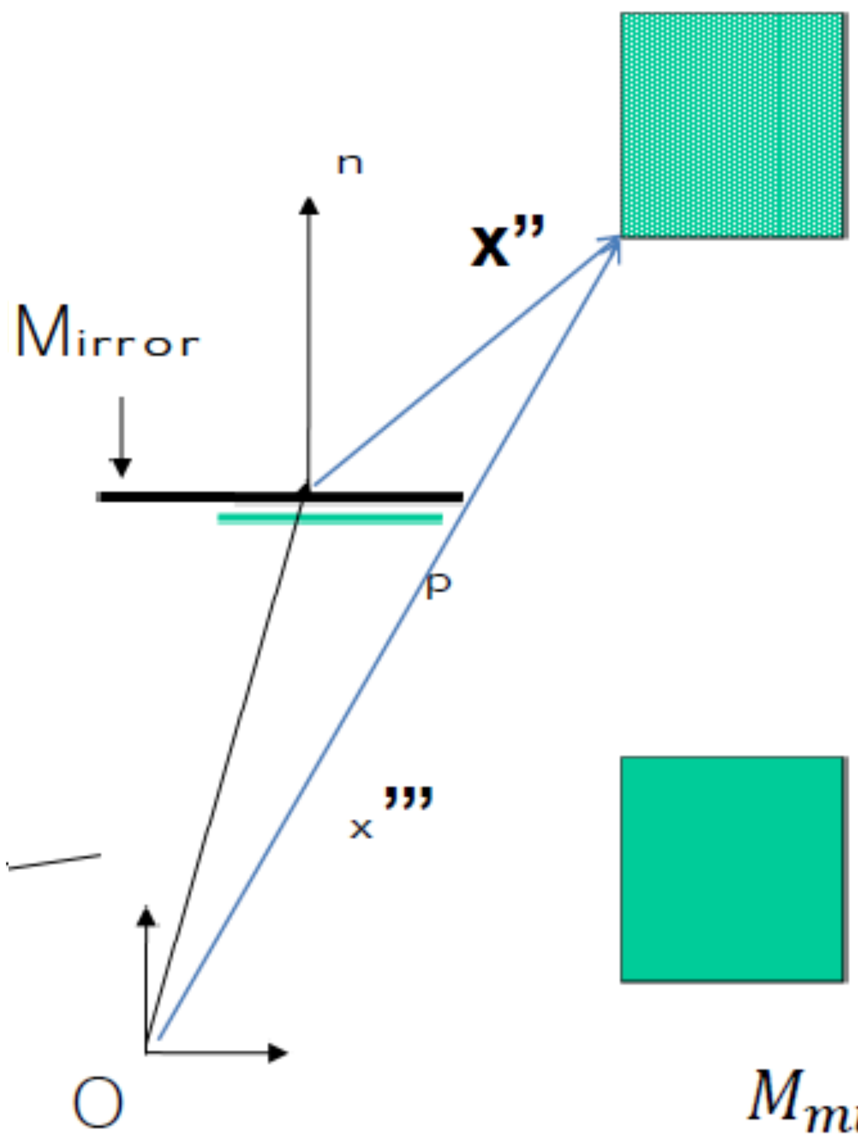
Indexing Cubic Maps

- How do you decide which texture coordinates to use?
- Divide by the coordinate with the largest magnitude
- Now have a value in the range [-1,1]
- Remapped to a value between 0 and 1.





Flat Mirrors



$$x' = R(n)^{-1}T(-p)\,x$$

$$x'' = S(1,1,-1)\,x'$$

$$x''' = T(p)R(n)\,x''$$

$$x''' = T(p)R(n)S(1,1,-1)R(n)^{-1}T(-p)\,x$$

$$M_{mirrror} = T(p)R(n)S(1,1,-1)R(n)^{-1}T(-p)$$

Stencil buffer mirrors

- First pass:
  - Render the scene without the mirror
- For each mirror:
  - Second pass:
    - Clear the stencil, disable the write to the colour buffer, render the mirror, setting the stencil to 1 if the depth test passes
  - Third pass:
    - Clear the depth buffer with the stencil active, passing things inside the mirror only
    - Reflect the world and draw using the stencil test. Only things seen in the mirror will be drawn
    - Combine it with the scene made during the first pass

Stencil buffer after the second pass

Render the mirrored scene into the stencil

Shadows

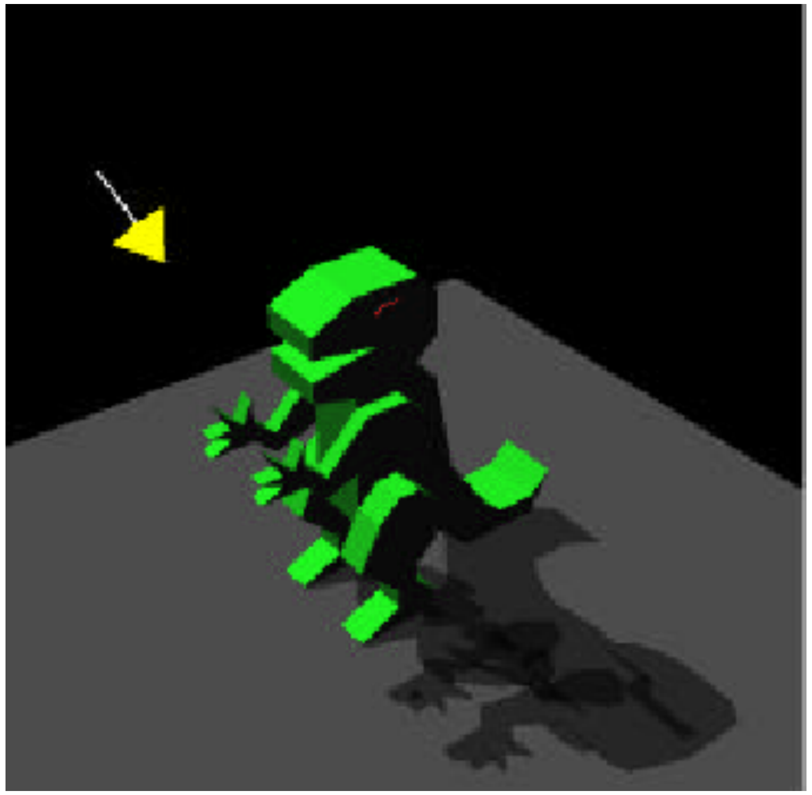
- Planar:

$$v' = M_s M_{g \leftarrow l} v$$

shadow vertex

shadow matrix

point in local coordinates



- Shadow texture:

Shadows

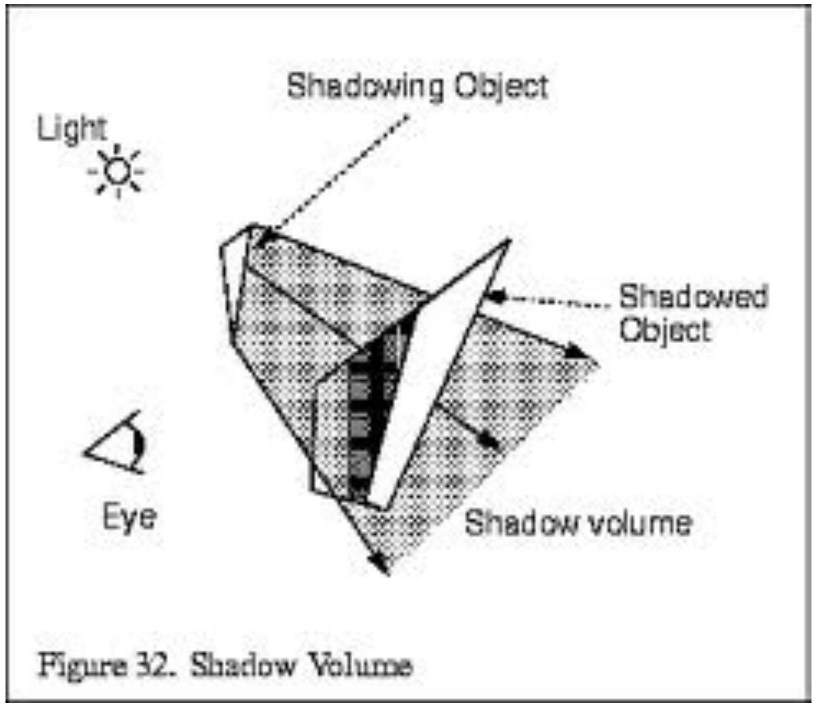
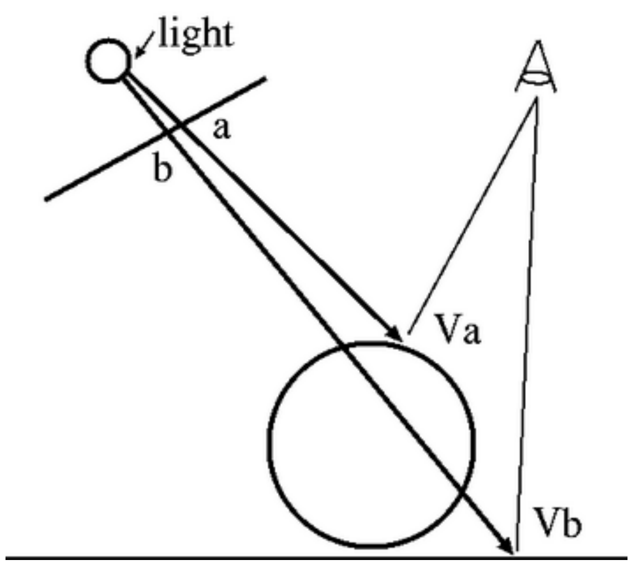
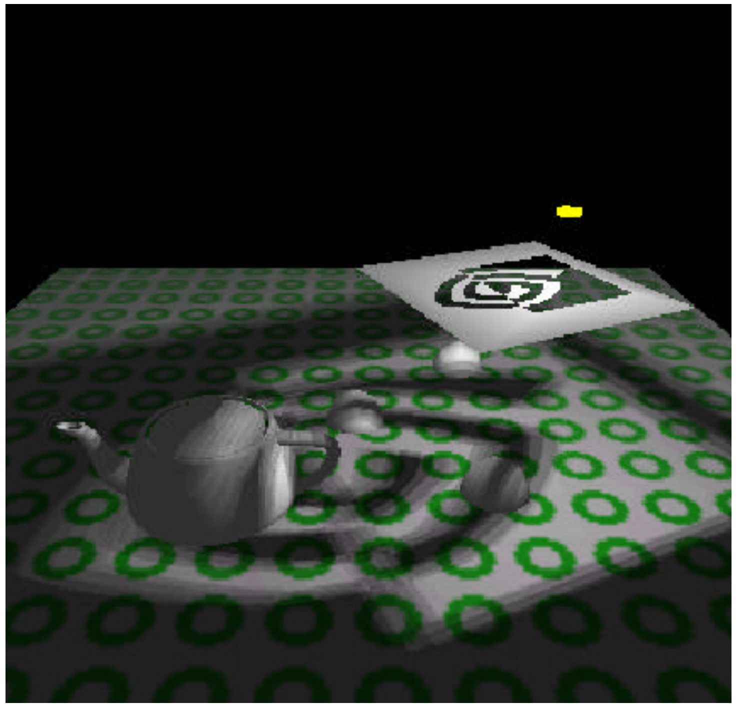
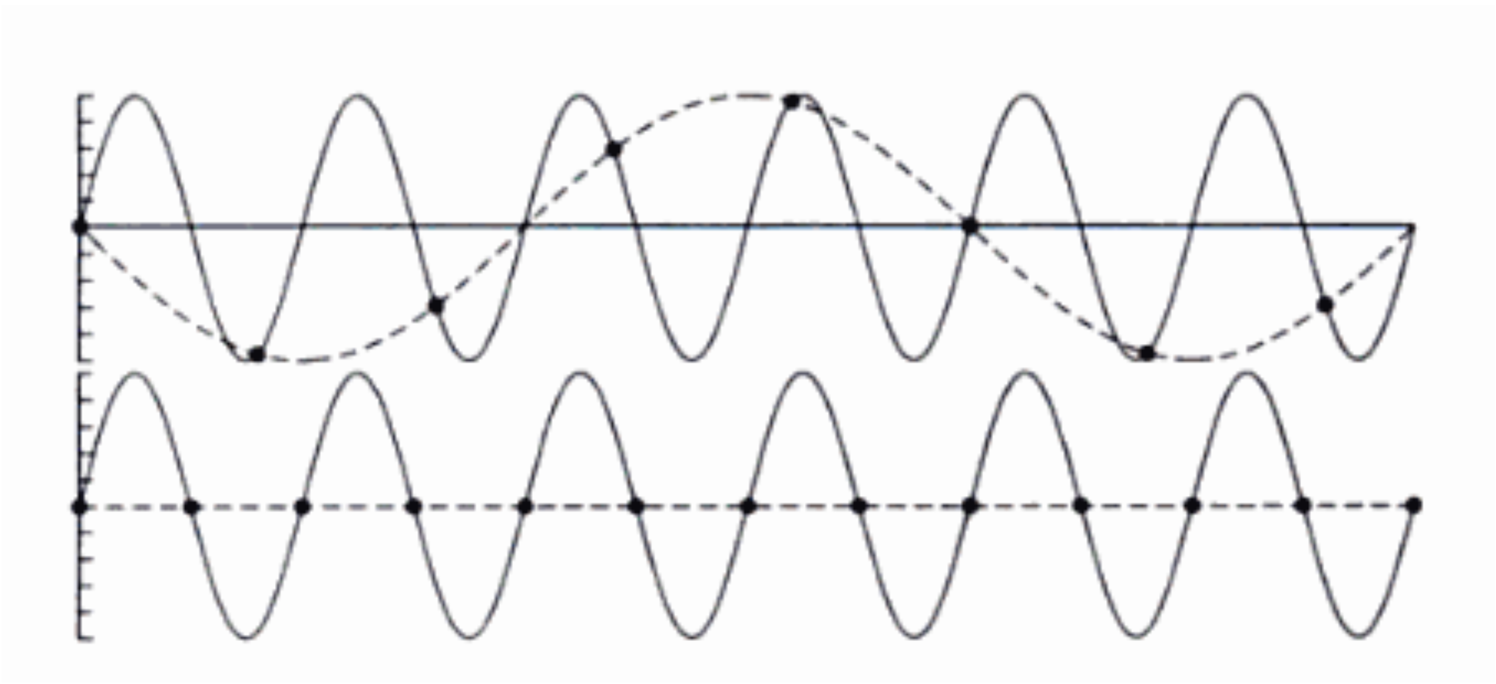


Figure 32. Shadow Volume





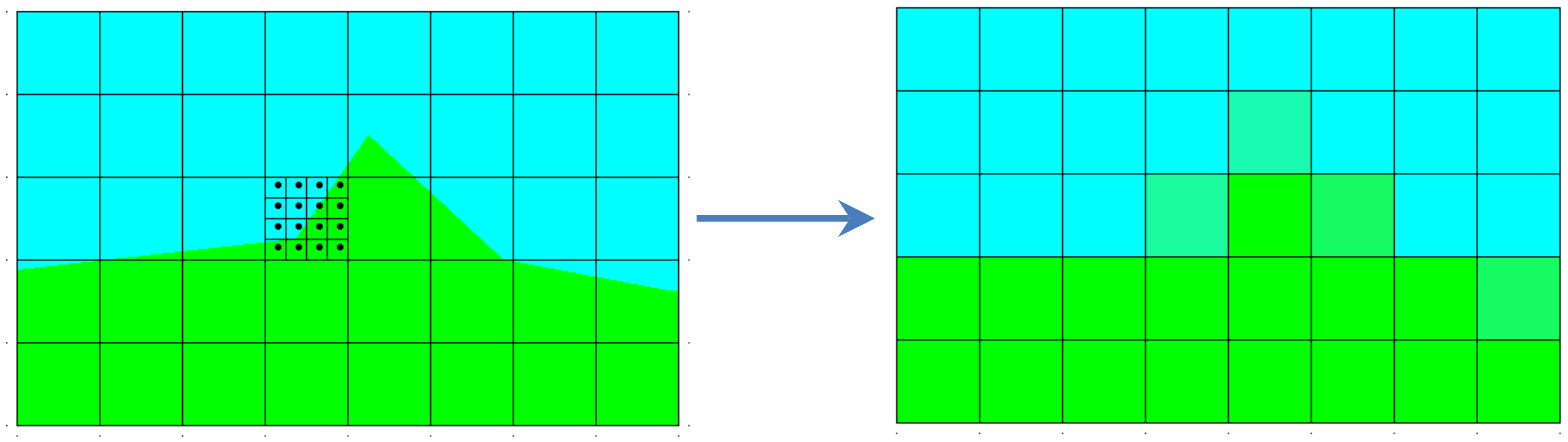
Antialiasing



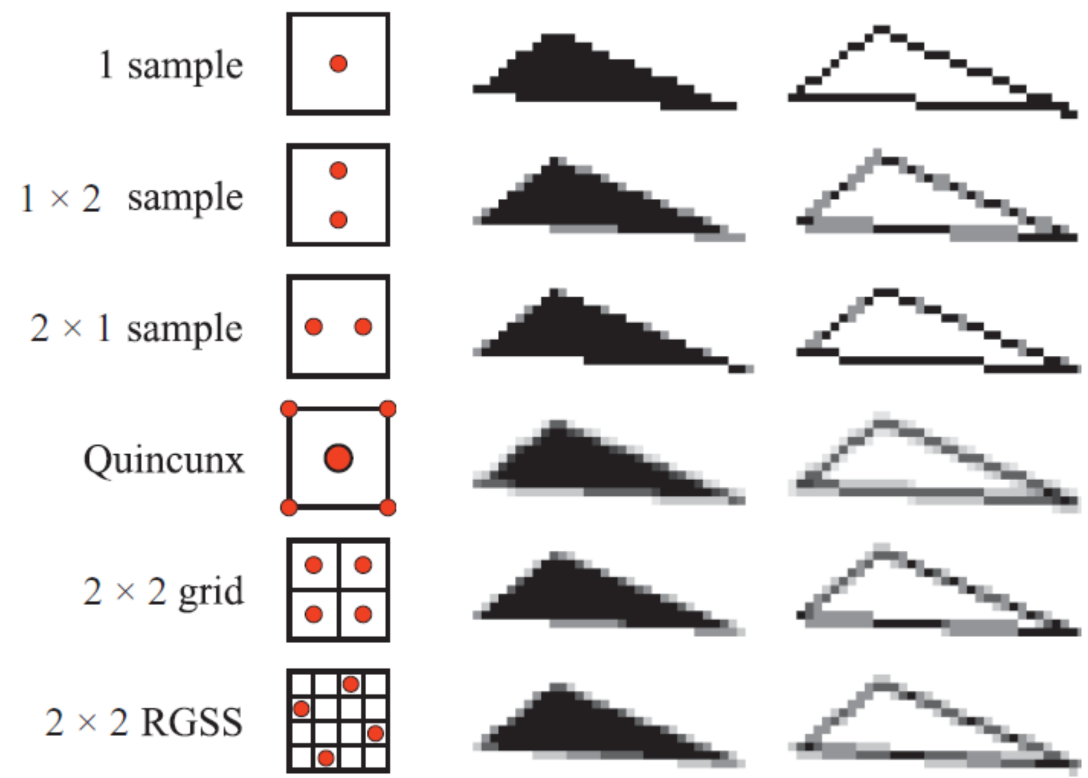
$$f_{signal} = 0.8 f_{sample}$$

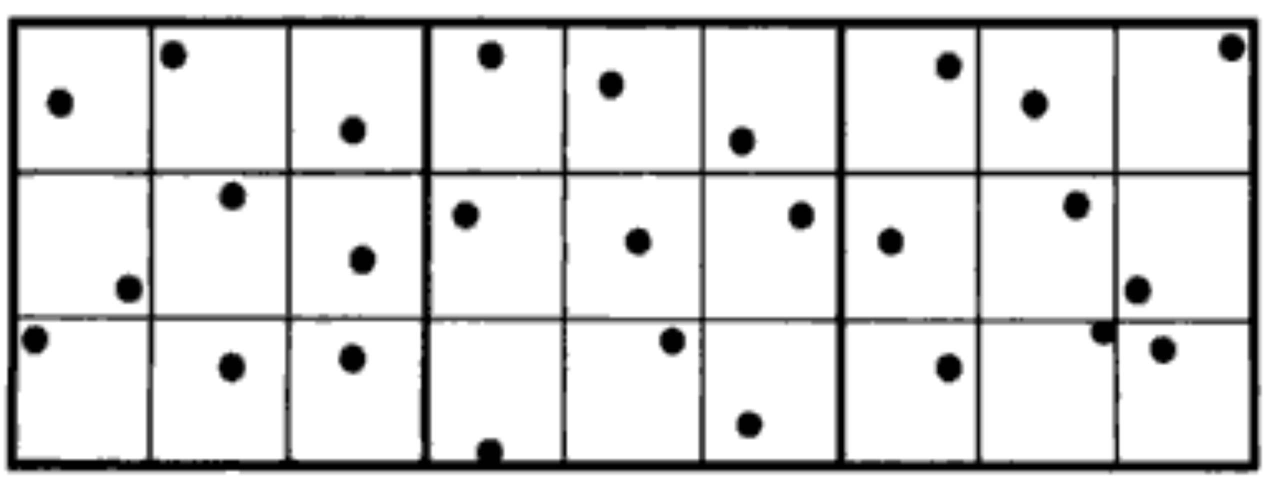
$$f_{signal} = 0.5 f_{sample}$$

$$f_{signal} < 0.5 f_{sample}$$

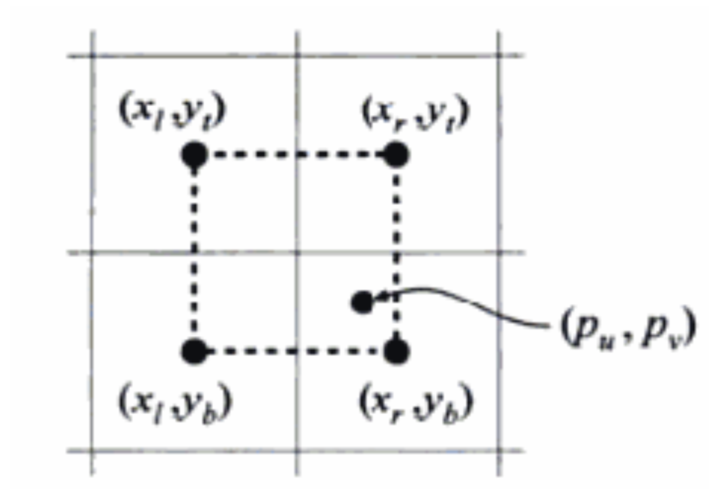


Subsampling schemes





Antialiasing textures



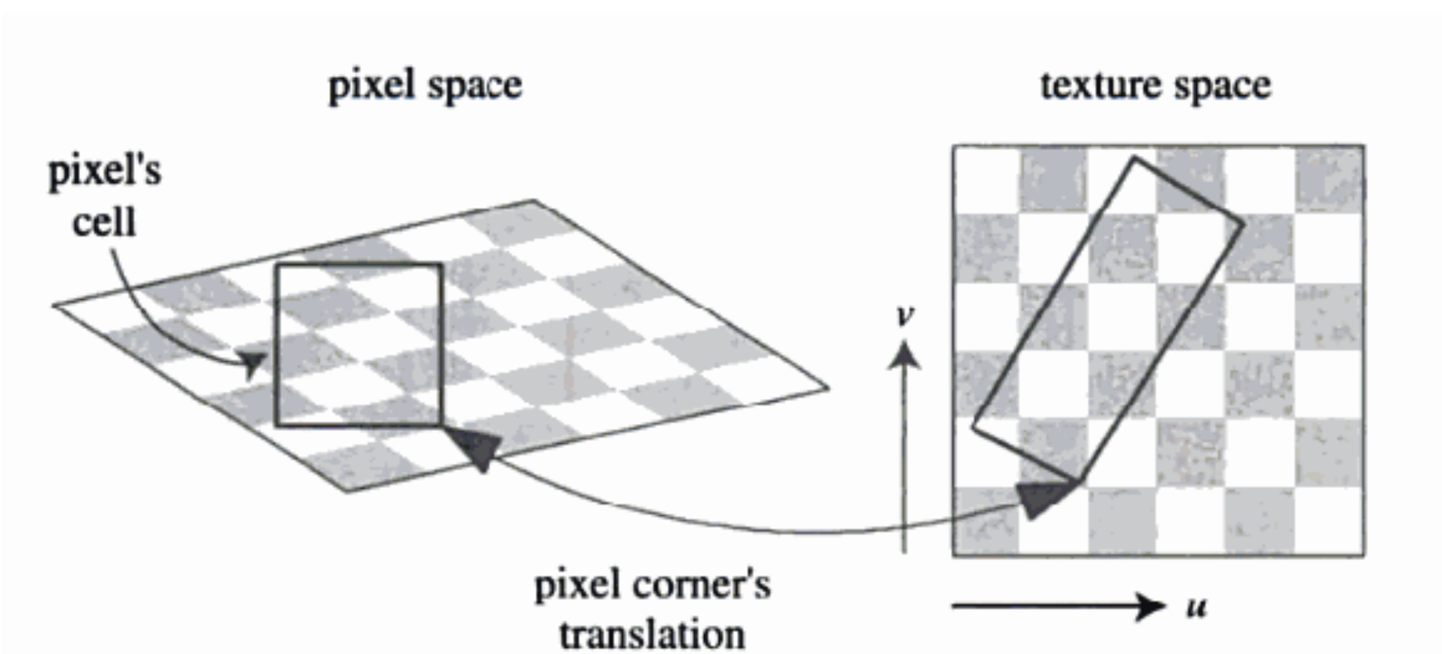
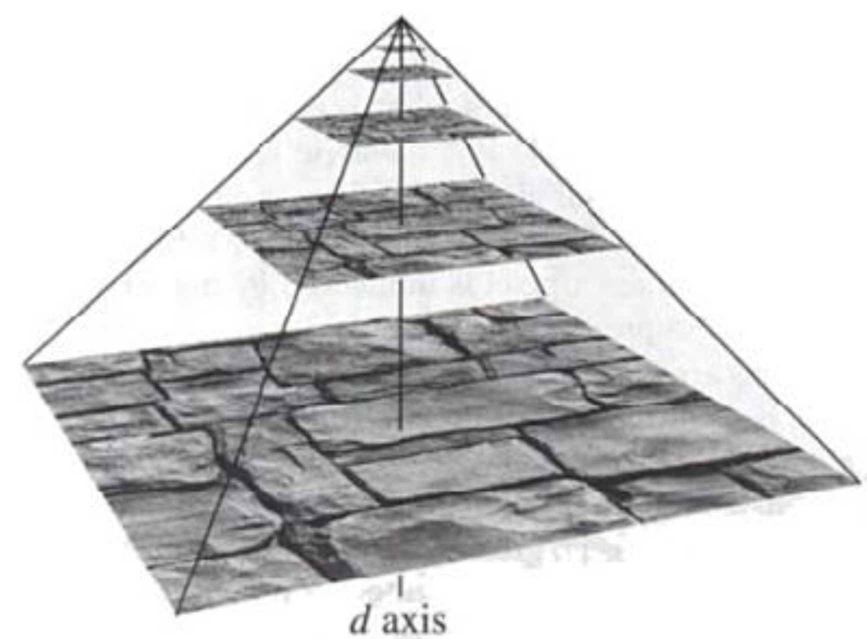
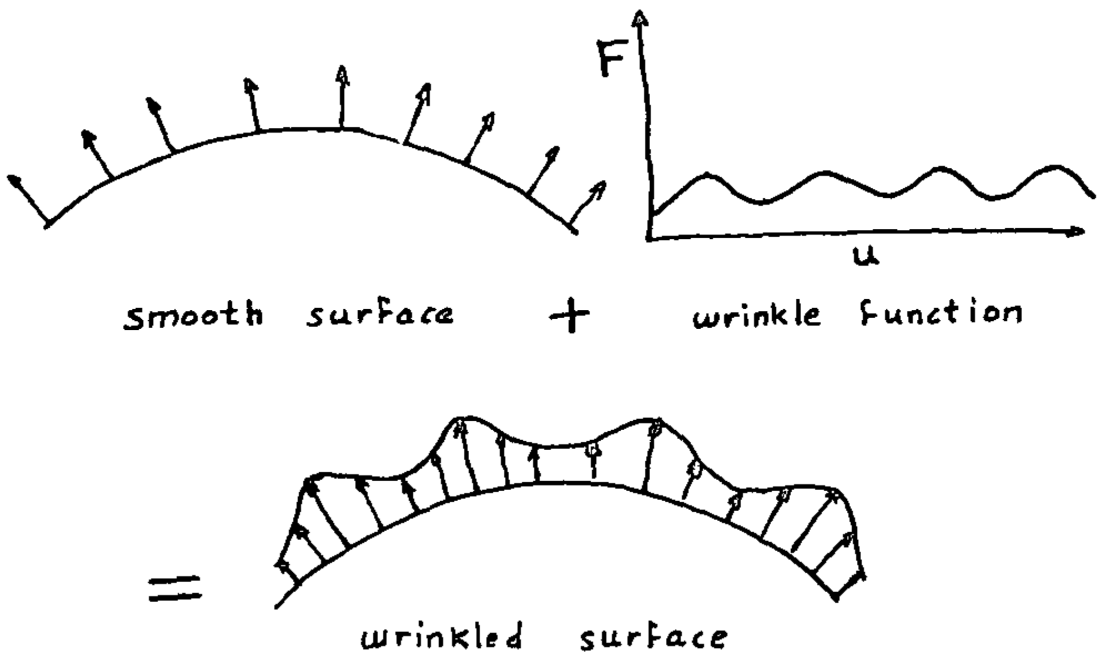
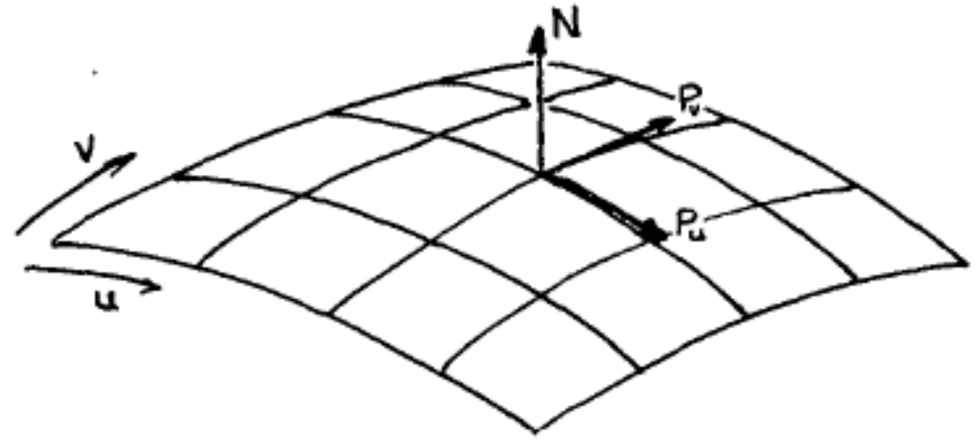


Figure 5.13. On the left is a square pixel cell and its view of a texture. On the right is the projection of the pixel cell onto the texture itself.

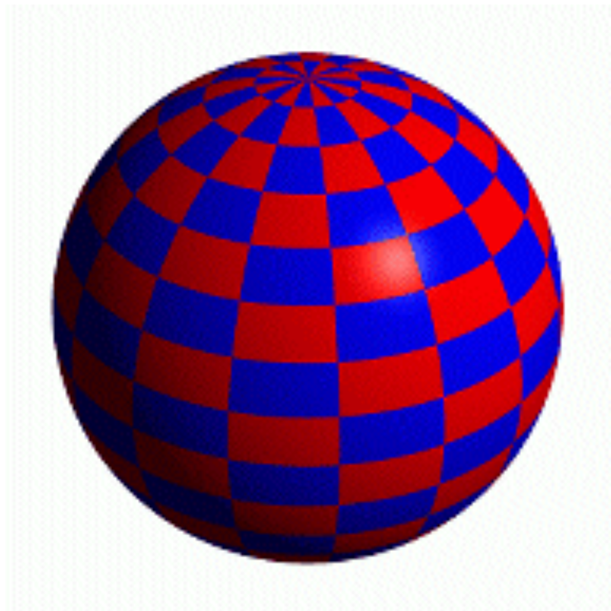


Bump mapping



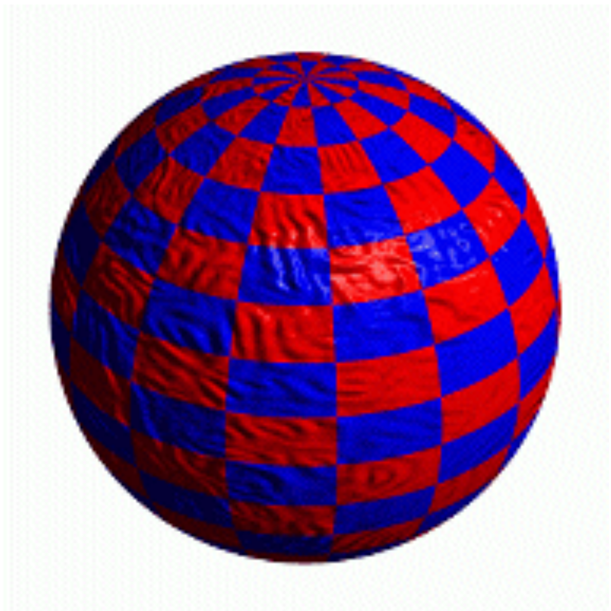


$$\mathbf{n}' = \mathbf{n} + \frac{F_u(\mathbf{n} \times \mathbf{P_v}) - F_v(\mathbf{n} \times \mathbf{P_u})}{||\mathbf{n}||}$$



Sphere w/Diffuse Texture

Swirly Bump Map

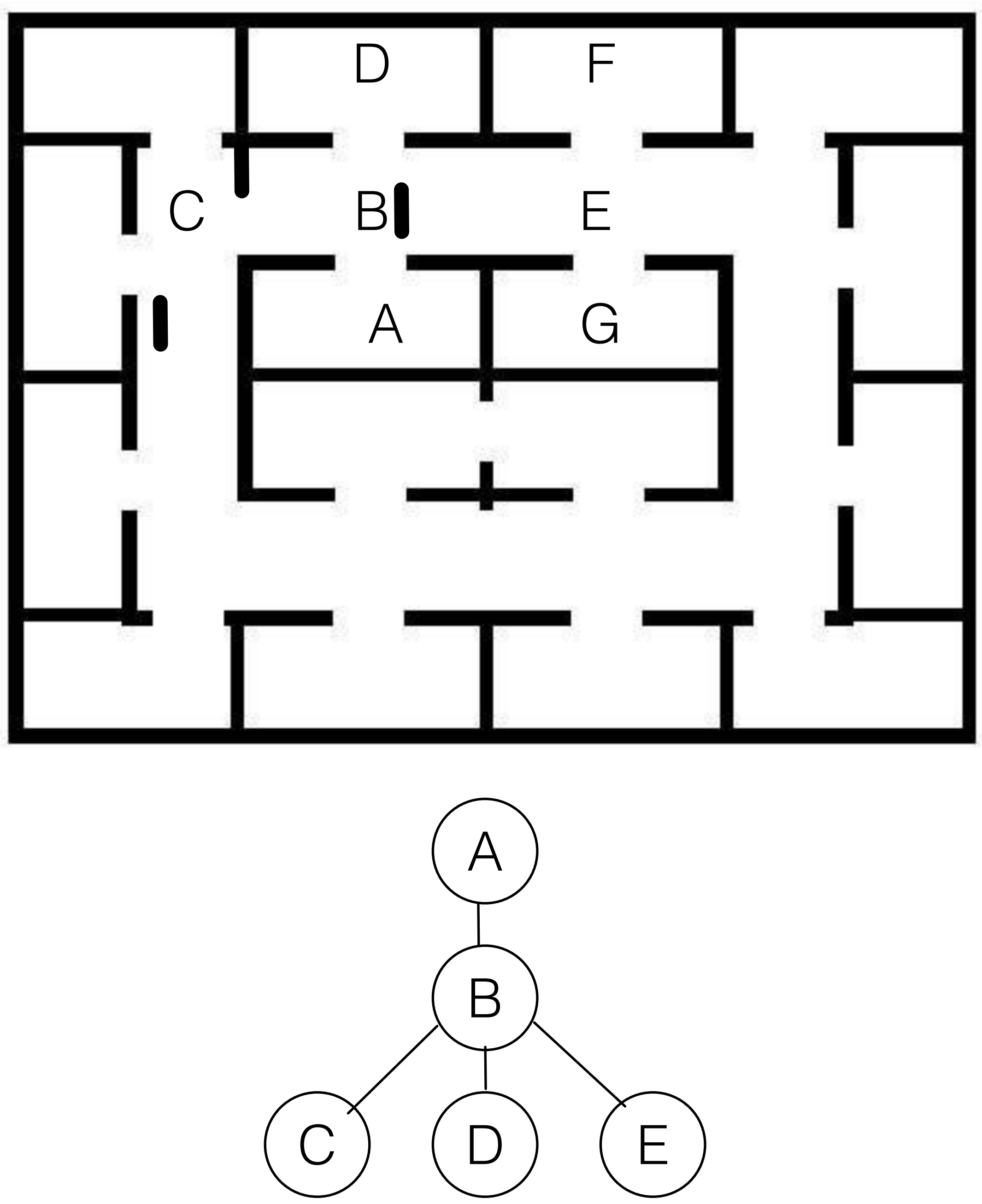


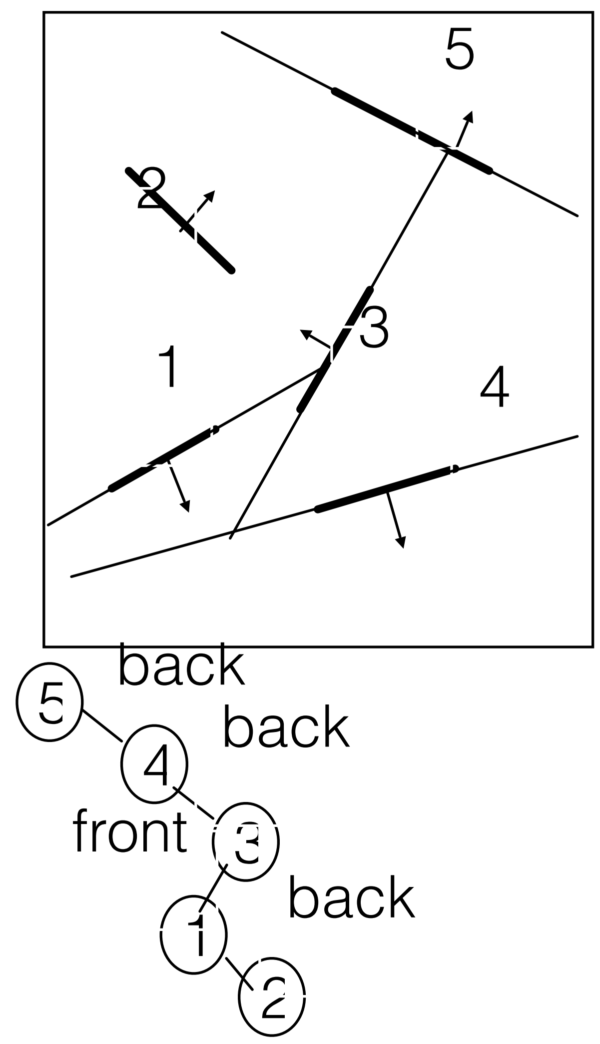
Sphere w/Diffuse Texture & Bump Map

Z-buffer

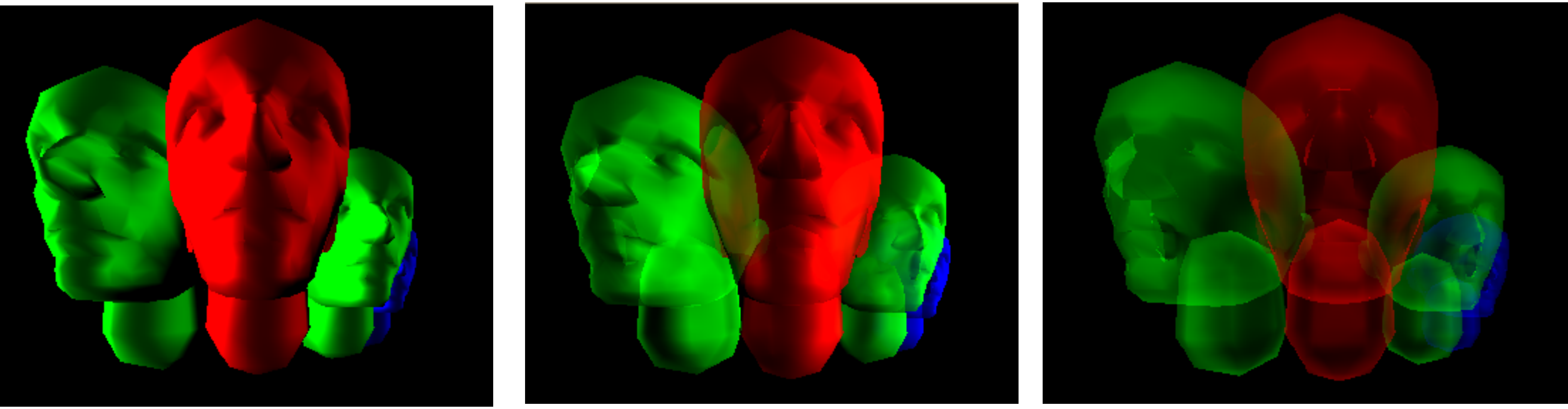
- Advantages:
  - Simple to implement in hardware
  - Memory is relatively cheap
  - Works with any primitives
  - Unlimited complexity
  - No need to sort objects or calculate intersections
- Disadvantages:
  - Wasted time drawing hidden objects
  - Z-precision errors (aliasing)

Hidden surface removal





Transparency

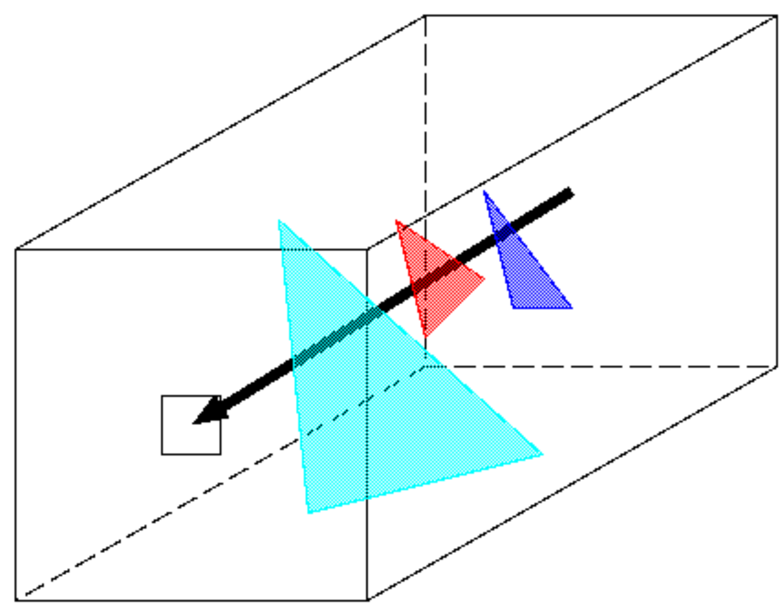


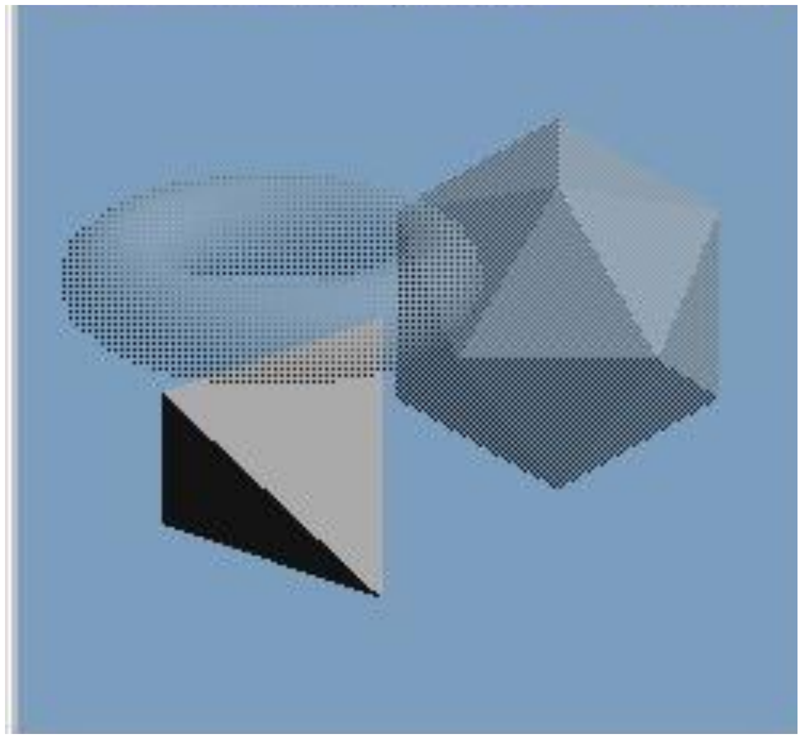
$$C_o = \alpha C_s + (1-\alpha)C_d$$

C_o = New pixel colour

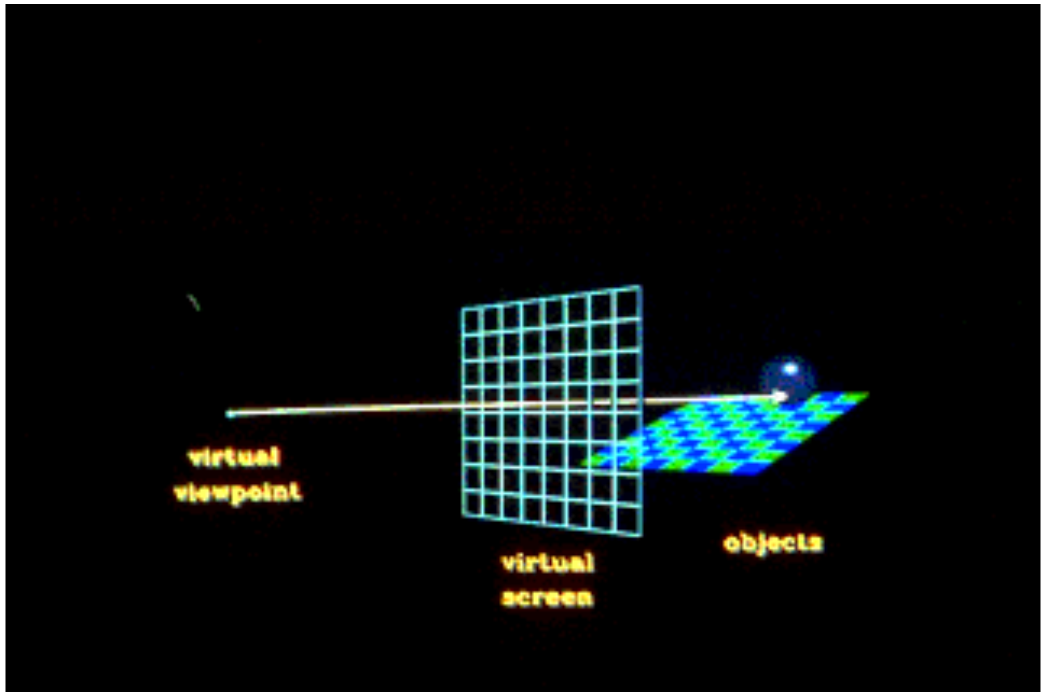
C_s = Transparent object colour

C_d = Current pixel colour

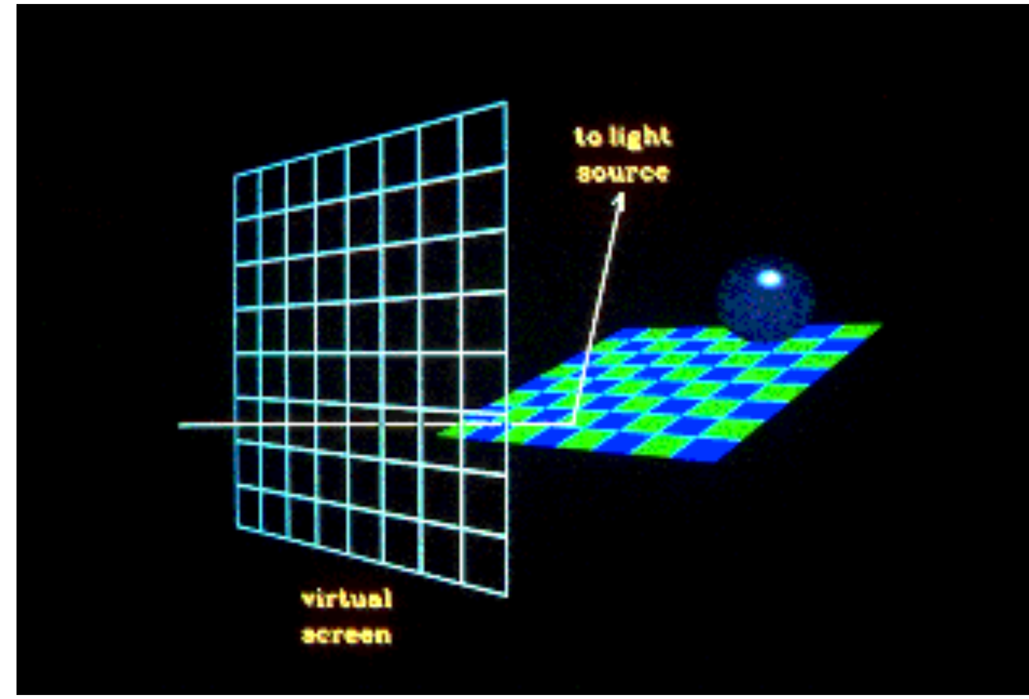




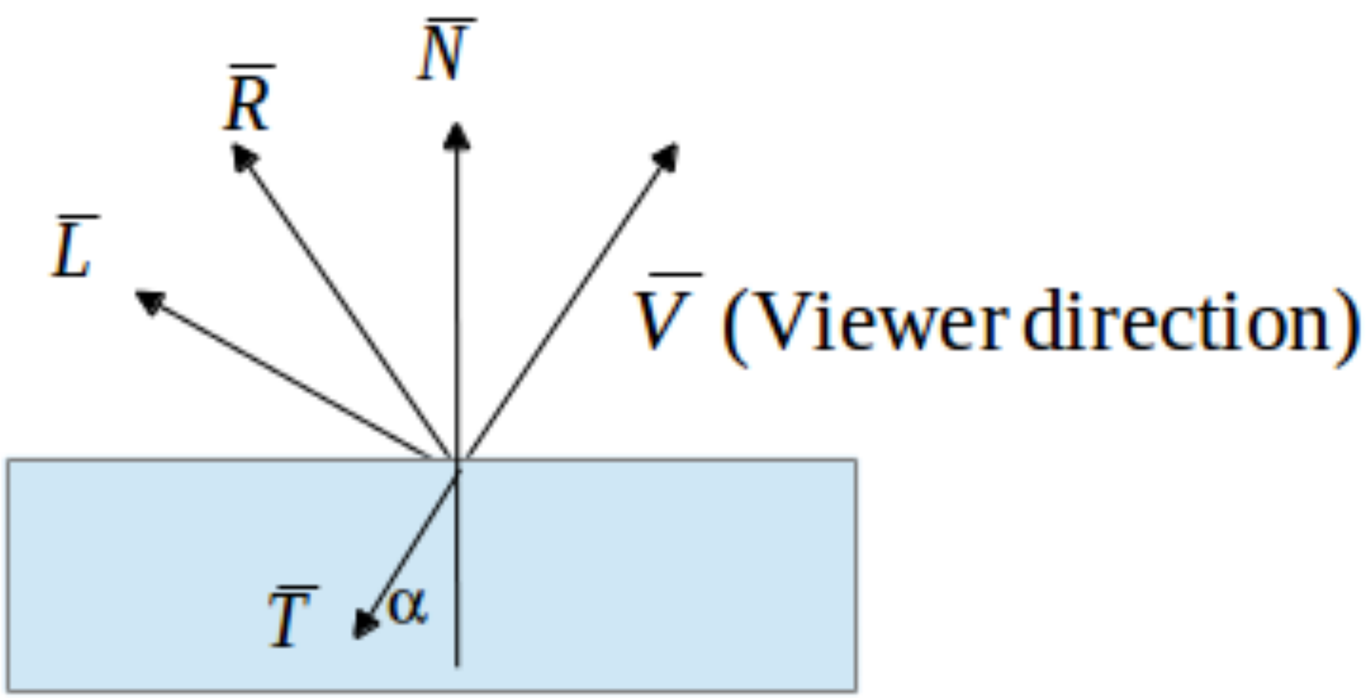
Ray tracing



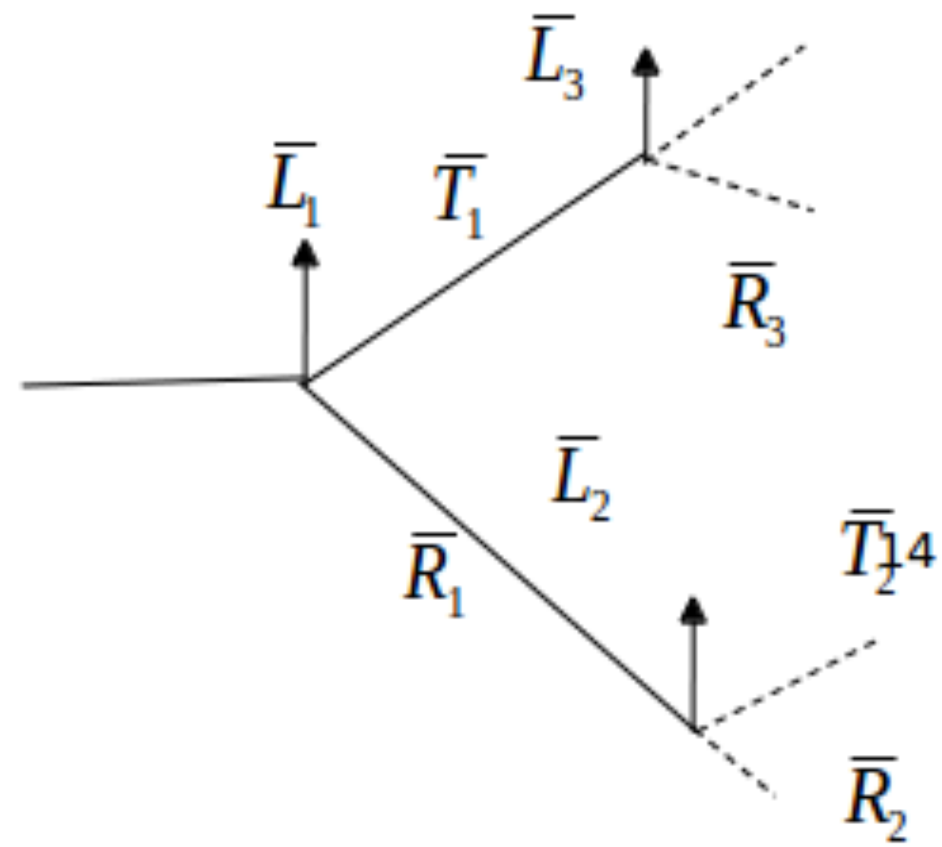
virtual viewpoint
virtual screen
objects



to light source
virtual screen



$\bar{N}$
$\bar{R}$
$\bar{L}$
$\bar{V}$ (Viewer direction)
$\bar{T}$
α



$\bar{L}_3$
$\bar{L}_1$
$\bar{T}_1$
$\bar{R}_3$
$\bar{L}_2$
$\bar{R}_1$
$\bar{T}_2$
$\bar{R}_2$

Ray/sphere intersection

This gives us the solution for t as:

$$t = \frac{-2\boldsymbol{d} \cdot (\boldsymbol{e} - \boldsymbol{s}) \pm \sqrt{(2\boldsymbol{d} \cdot (\boldsymbol{e} - \boldsymbol{s}))^2 - 4(\boldsymbol{d} \cdot \boldsymbol{d})((\boldsymbol{e} - \boldsymbol{s}) \cdot (\boldsymbol{e} - \boldsymbol{s}) - r^2)}}{2(\boldsymbol{d} \cdot \boldsymbol{d})}$$

With the number of solutions determined by the value in the square root.

- If $b^2 - 4ac > 0$ there are two intersections of the ray with the sphere
- If $b^2 - 4ac = 0$ the ray grazes the sphere and there is a single intersection
- If $b^2 - 4ac < 0$ the ray misses the sphere completely.

Ray/plane intersections

To calculate the intersection of a ray with a plane we substitute the equation for the points on the ray into the implicit plane equation:

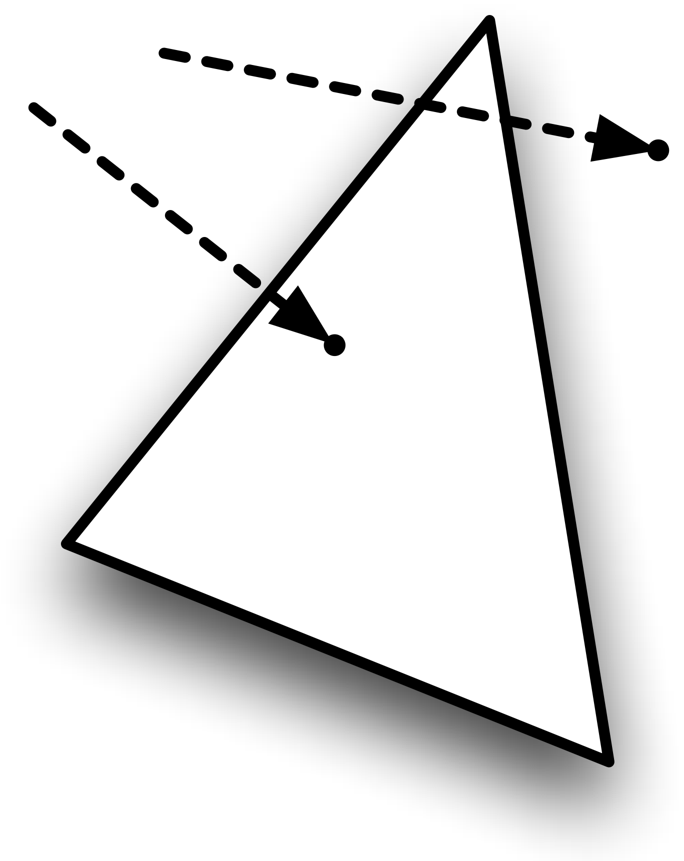
$$
\begin{aligned}
(\boldsymbol{e} + t\boldsymbol{d} - \boldsymbol{s}) \cdot \boldsymbol{n} &= 0 \\
(\boldsymbol{e} - \boldsymbol{s}) \cdot \boldsymbol{n} + t\boldsymbol{d} \cdot \boldsymbol{n} &= 0 \\
t &= \frac{(\boldsymbol{s} - \boldsymbol{e}) \cdot \boldsymbol{n}}{\boldsymbol{d} \cdot \boldsymbol{n}}
\end{aligned}
$$

In the case where $\boldsymbol{d} \cdot \boldsymbol{n} = 0$ the ray is parallel to the plane, and so does not intersect it.

Ray/triangle intersection

First perform intersection with the plane:

$$t = \frac{(\boldsymbol{s} - \boldsymbol{e}) \cdot \boldsymbol{n}}{\boldsymbol{d} \cdot \boldsymbol{n}}$$

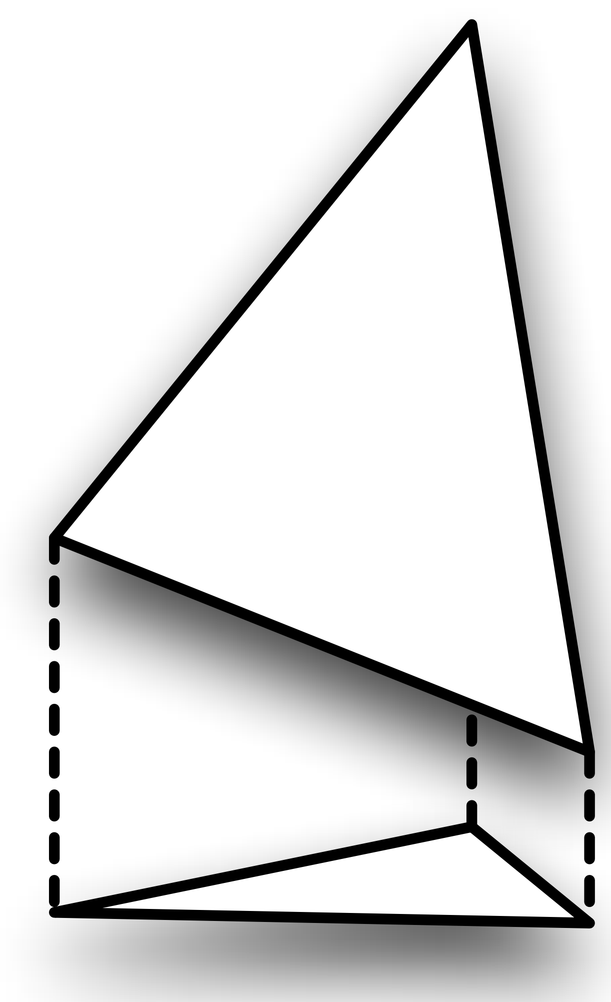


Then test if the point $\boldsymbol{r(t)} = \boldsymbol{e} + t\boldsymbol{d}$ lies within the triangle.

Projection onto primary planes

To make things simpler, we project the triangle onto one of the planes corresponding to a pair of axes (*xy*, *yz* or *xz*).

- We chose the plane on which the triangle has the largest projection, using the normal vector $\boldsymbol{n}$.
- The largest component of $\boldsymbol{n}$ is dropped e.g. if $|n_y|$ is the largest we project onto the *xz* plane, dropping the *y* coordinate.



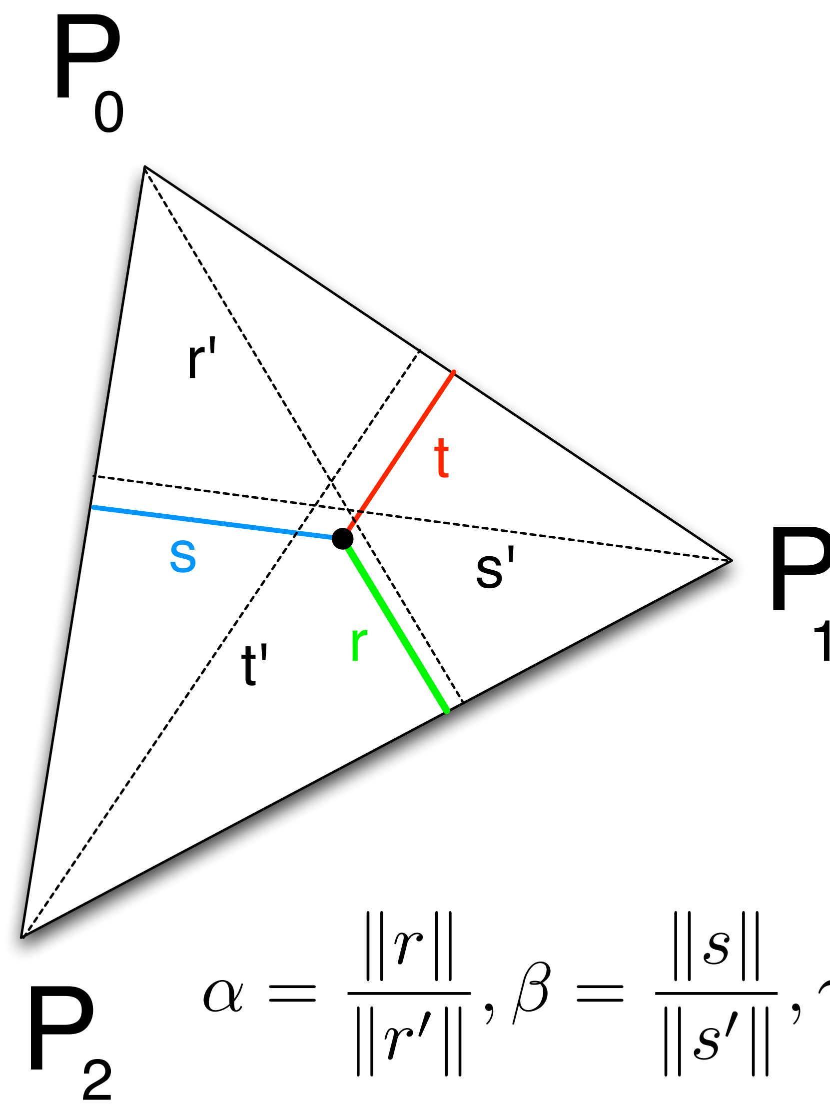
Projection onto primary planes

After projection to a 2D plane we can test for a point being inside the triangle using barycentric coordinates:

$$\alpha = \frac{f_{P_1P_2}(x,y)}{f_{P_1P_2}(x_0,y_0)}$$

$$\beta = \frac{f_{P_2P_0}(x,y)}{f_{P_2P_0}(x_1,y_1)}$$

$$\gamma = \frac{f_{P_0P_1}(x,y)}{f_{P_0P_1}(x_2,y_2)},$$



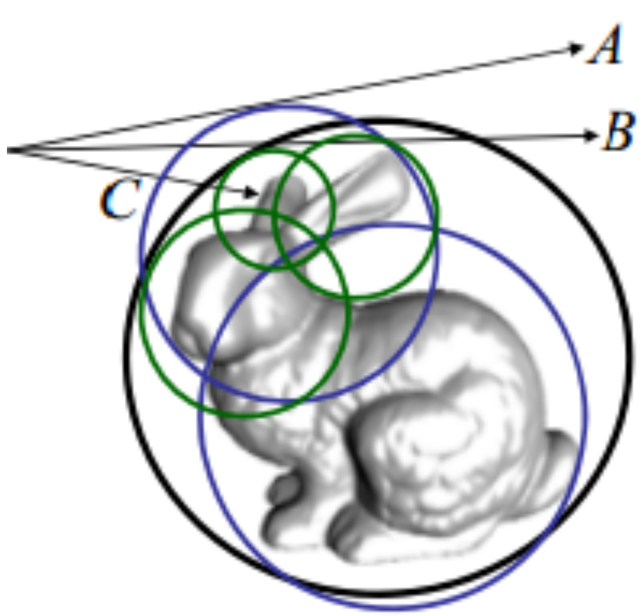
$$\alpha = \frac{\|r\|}{\|r'\|}, \beta = \frac{\|s\|}{\|s'\|}, \gamma = \frac{\|t\|}{\|t'\|}$$

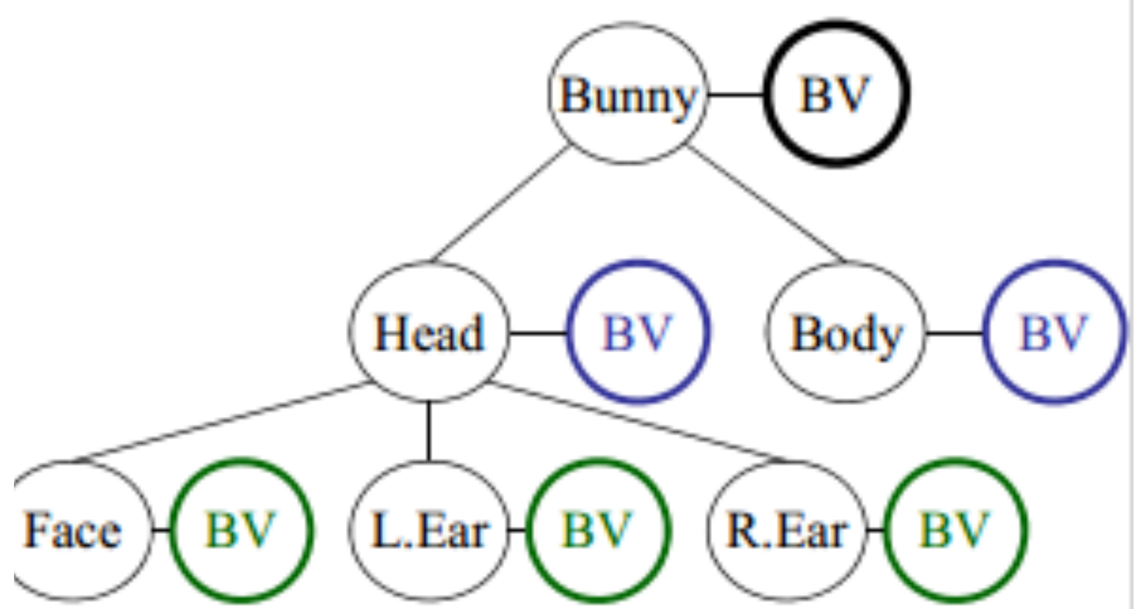
where

$$f_{pq}(x,y) = (y_q - y_p)x - (x_q - x_p)y + x_q y_p - y_q x_p$$

Bounding volume hierarchy

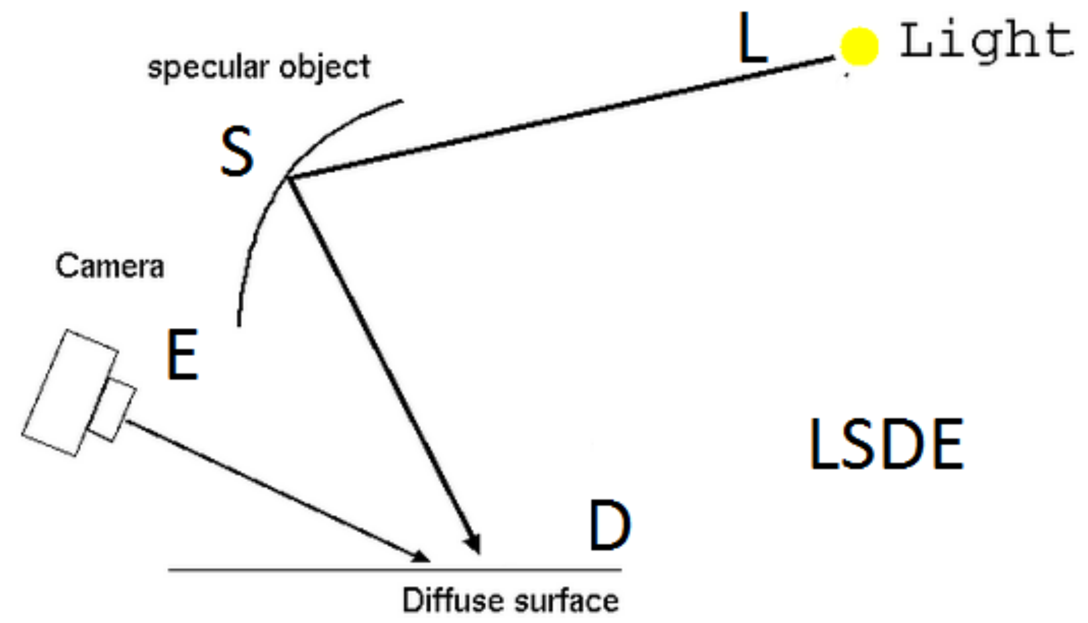
- Give each object a bounding volume
- The bounding volume does not partition
- The bounding volumes can overlap each other
- The volume higher in the hierarchy contains their children
- If a ray misses a bounding volume, no need to check for intersection with children
- If we intersect a bounding volume, check intersection with children

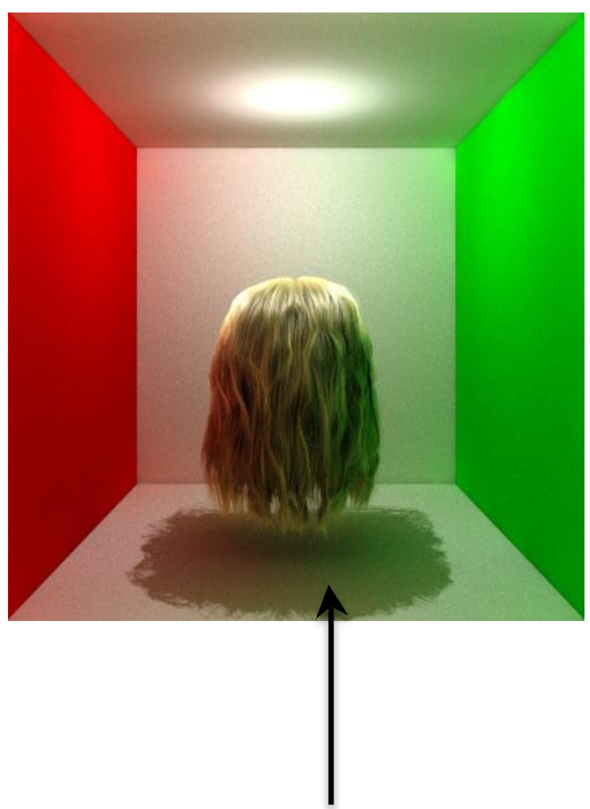




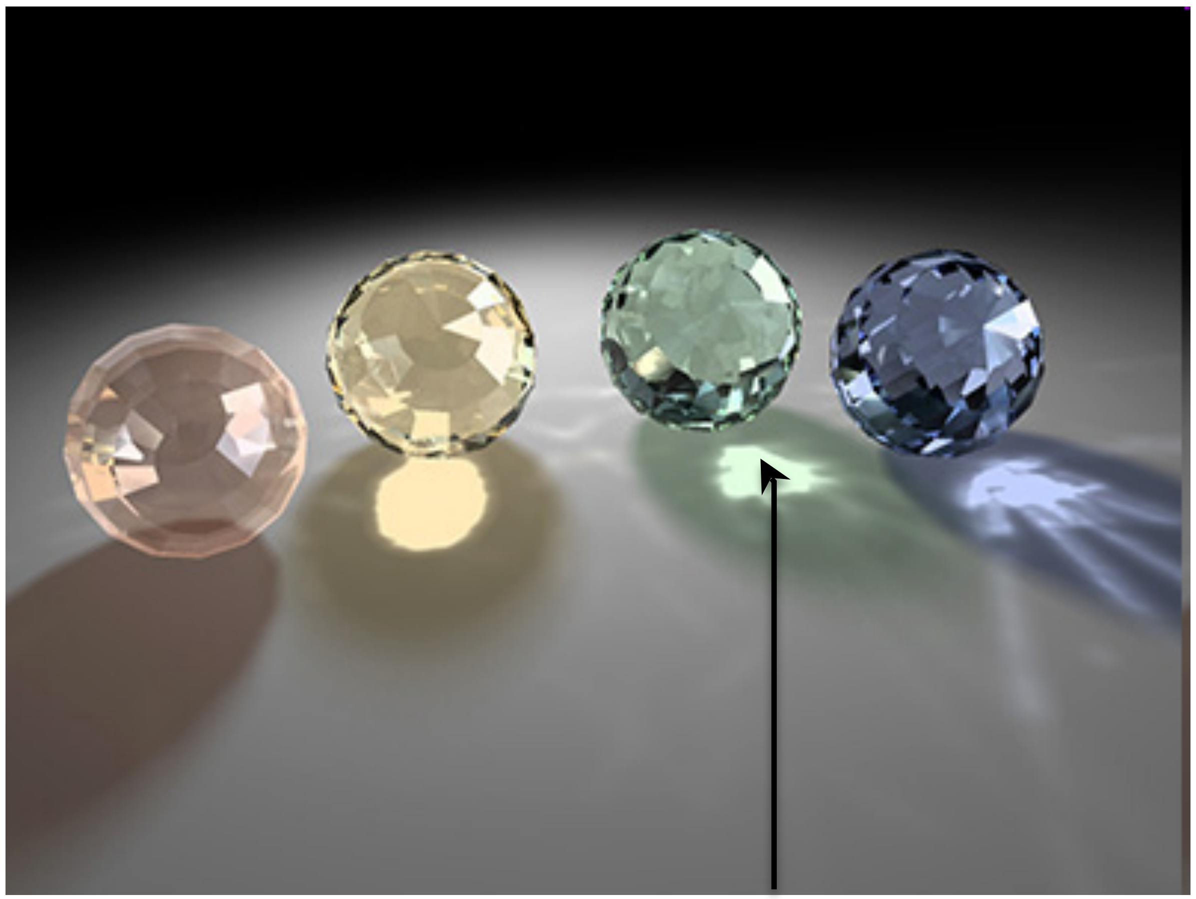
Light transport notations

- L light source
- E the eye
- S specular reflection or refraction
- D diffuse reflection



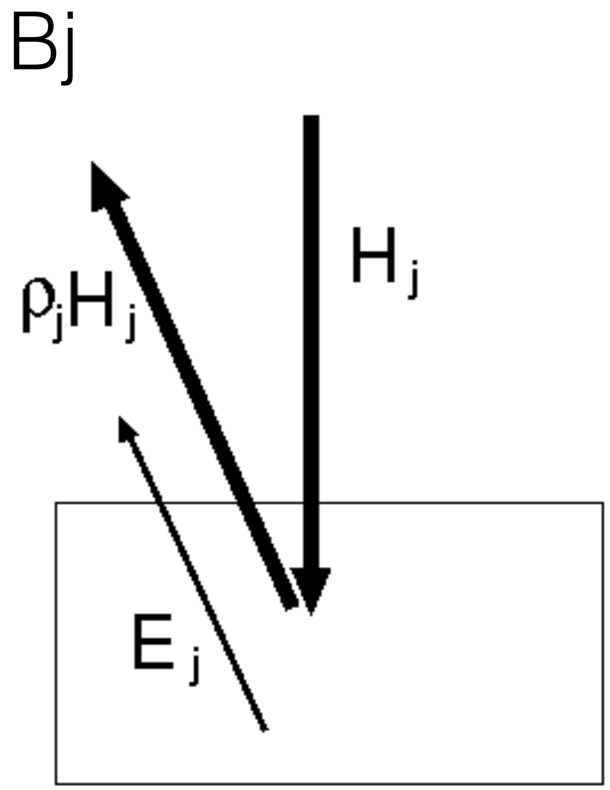


LDDE



LSDE

The Radiosity model



$$\boxed{B_j = \rho_j H_j + E_j}$$

B_j is the radiosity of surface j,

ρ_j is the reflectivity of surface j,

E_j is the energy emitted by surface j.

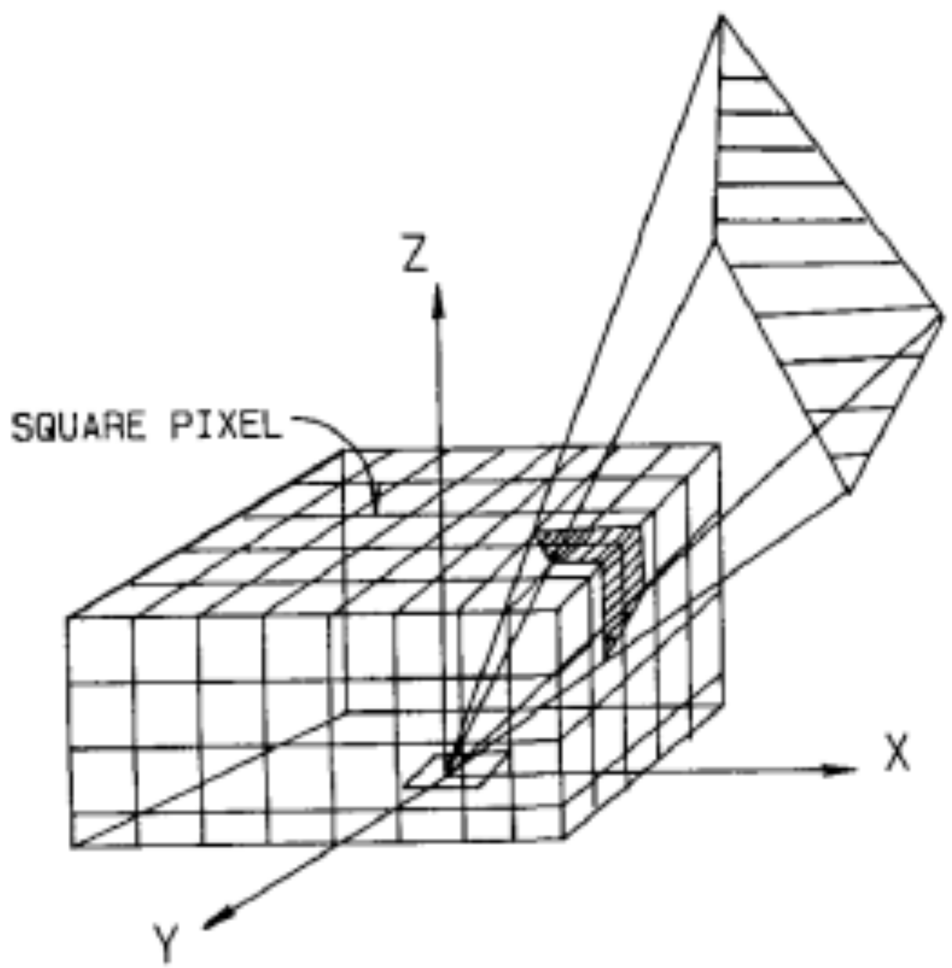
H_j is the energy incident on surface j

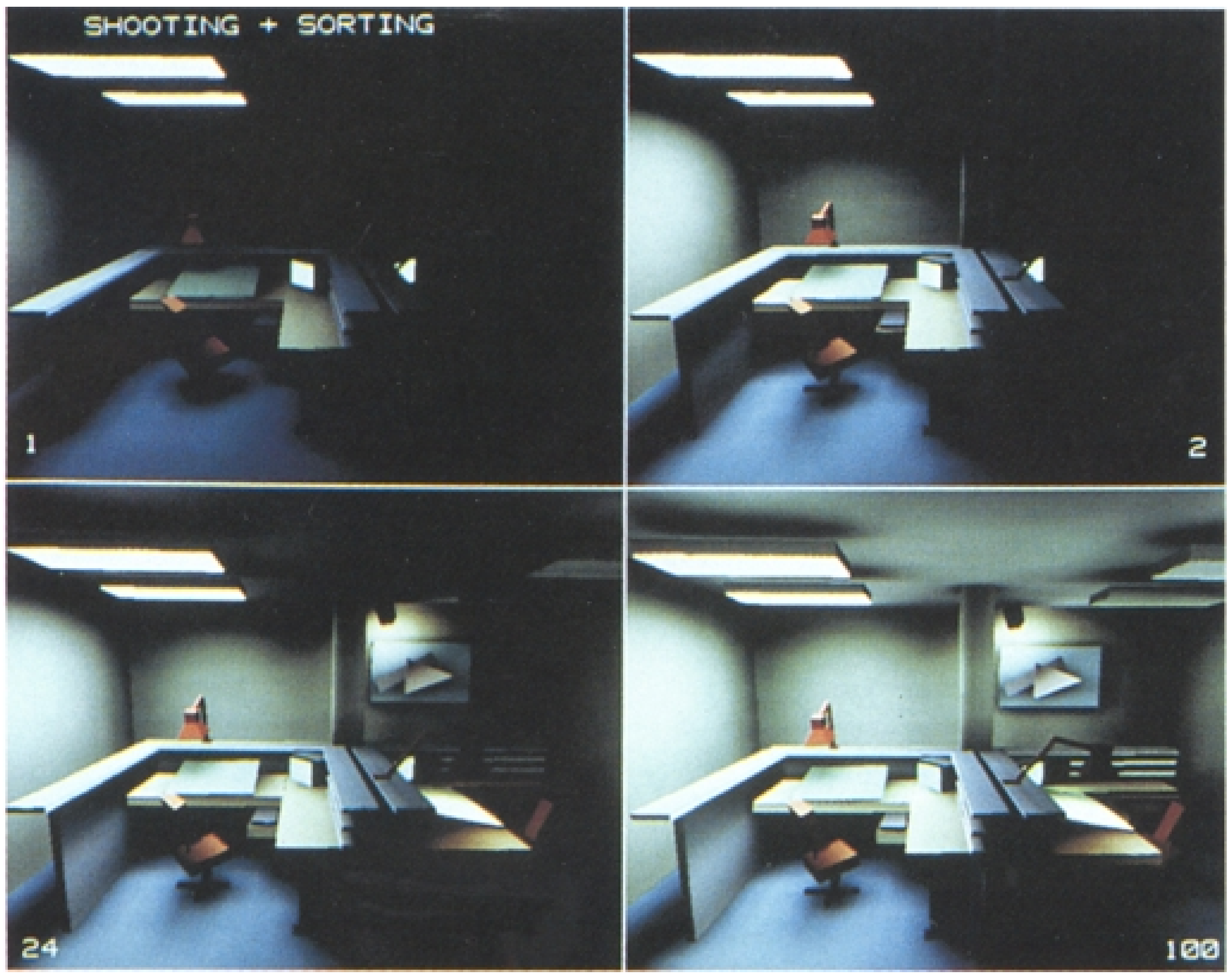
$$F_{ij} = \sum_i \sum_j \frac{\cos\phi_i \cos\phi_j}{\pi |r|^2} dA_i dA_j$$

$$B_j = E_j + \rho_j \sum_{i=1}^{N} B_i F_{i,j}$$

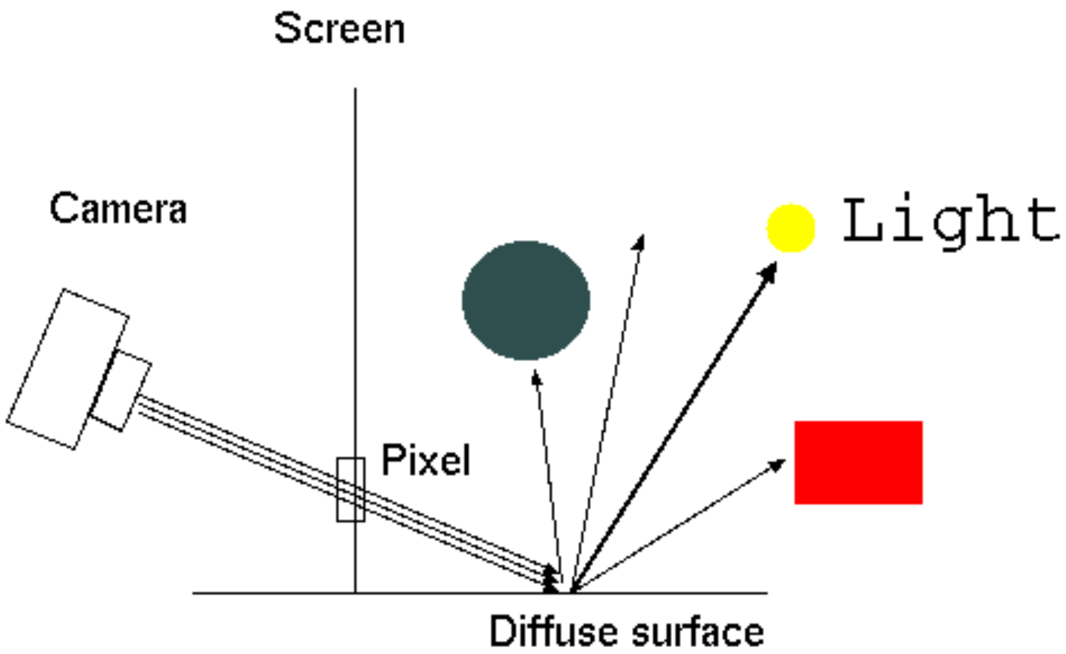
Radiosity

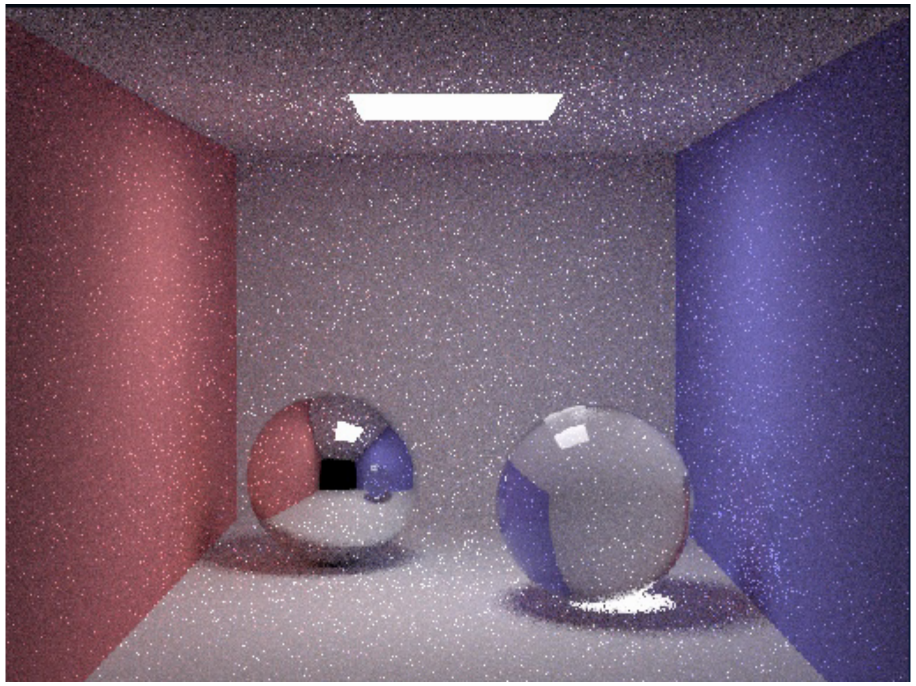
$$\begin{pmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \dots & -\rho_1 F_{1N} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \dots & -\rho_2 F_{2N} \\ \vdots & \vdots & \dots & \vdots \\ -\rho_N F_{N1} & -\rho_N F_{N2} & \dots & 1-\rho_N F_{NN} \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{pmatrix}$$

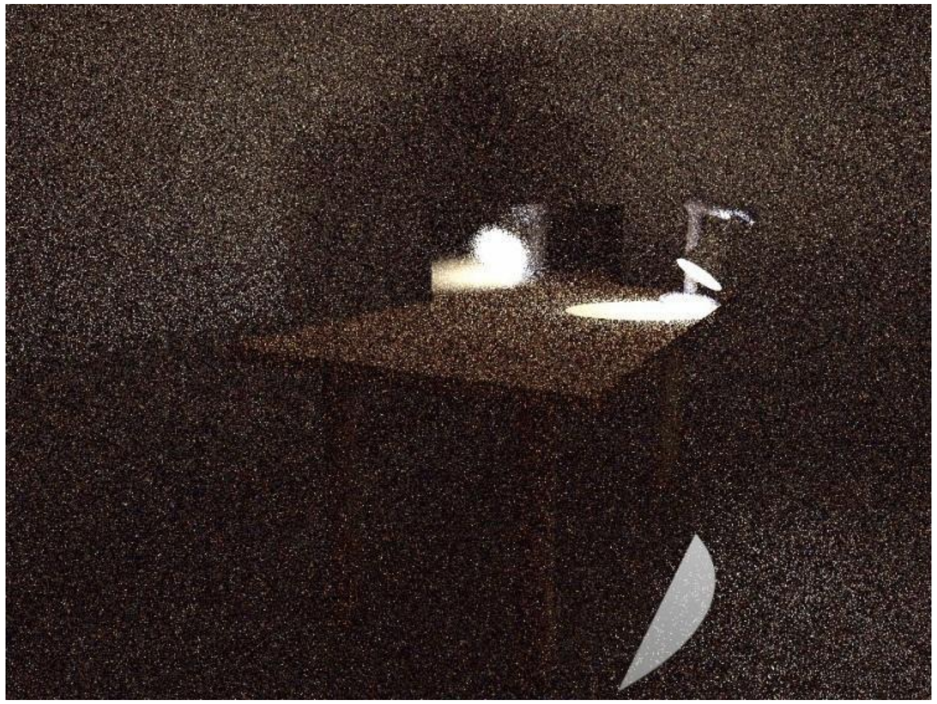


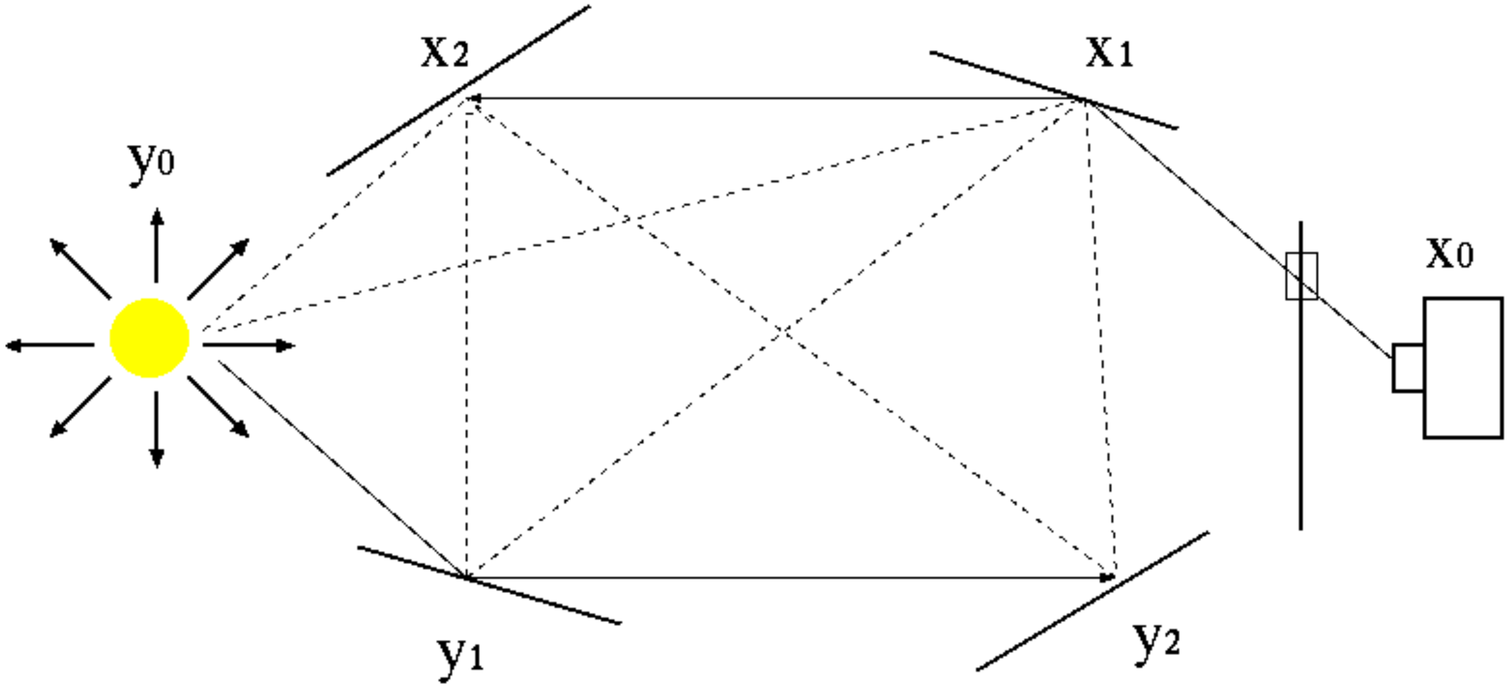


Path Tracing



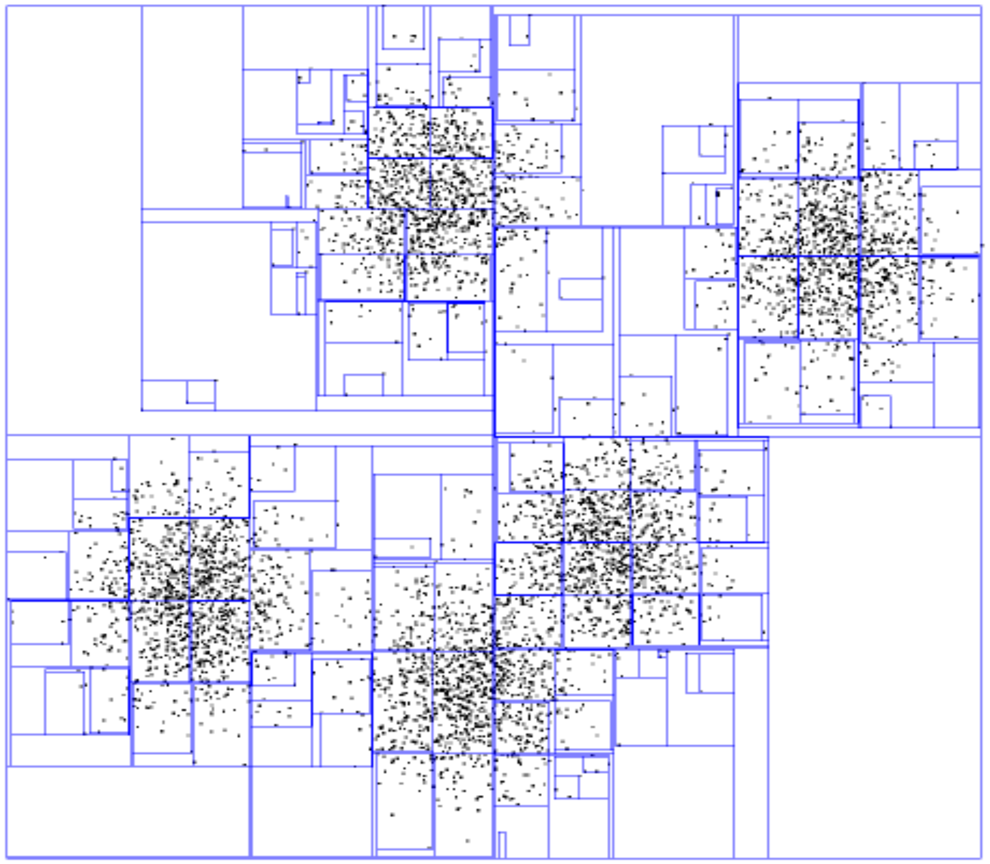




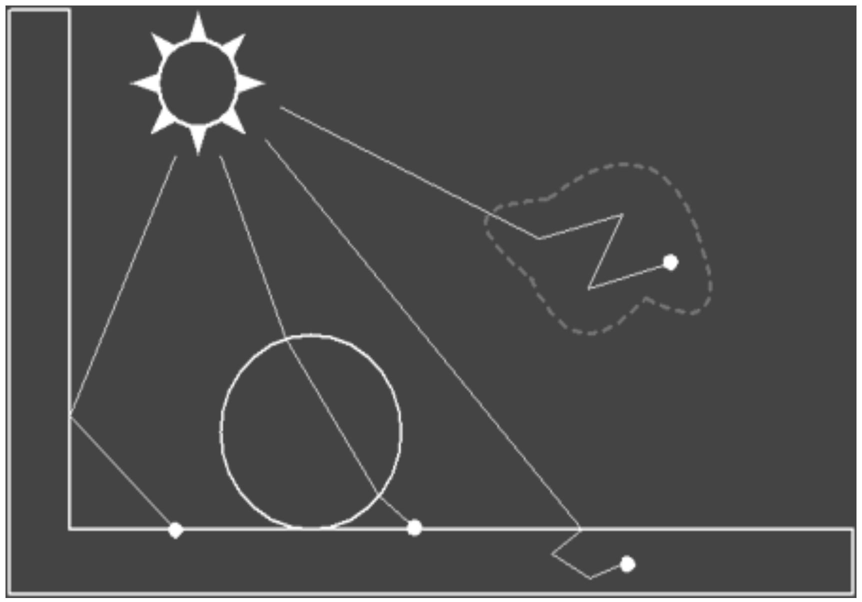


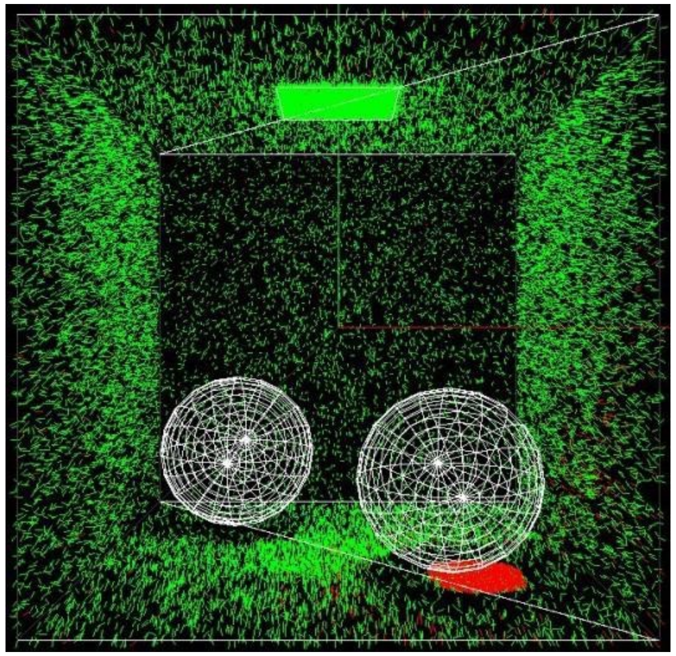
*

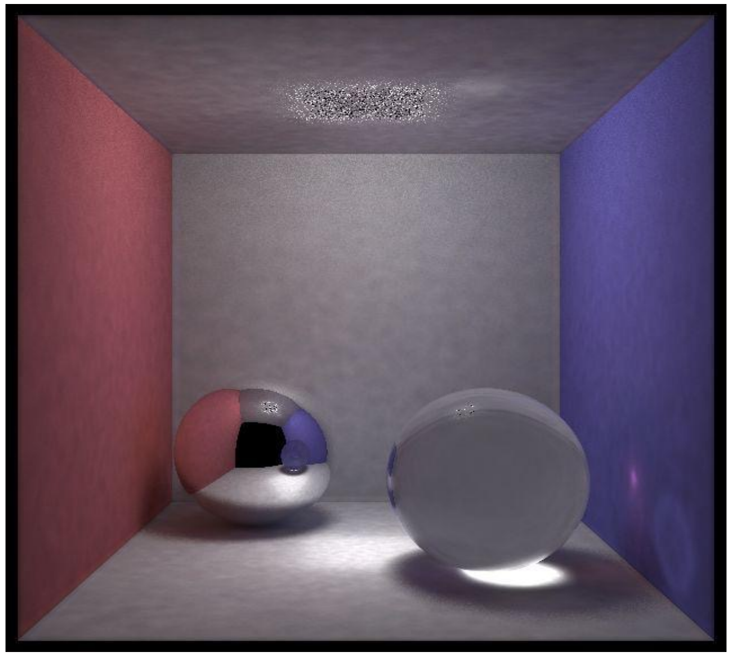
Photon Mapping



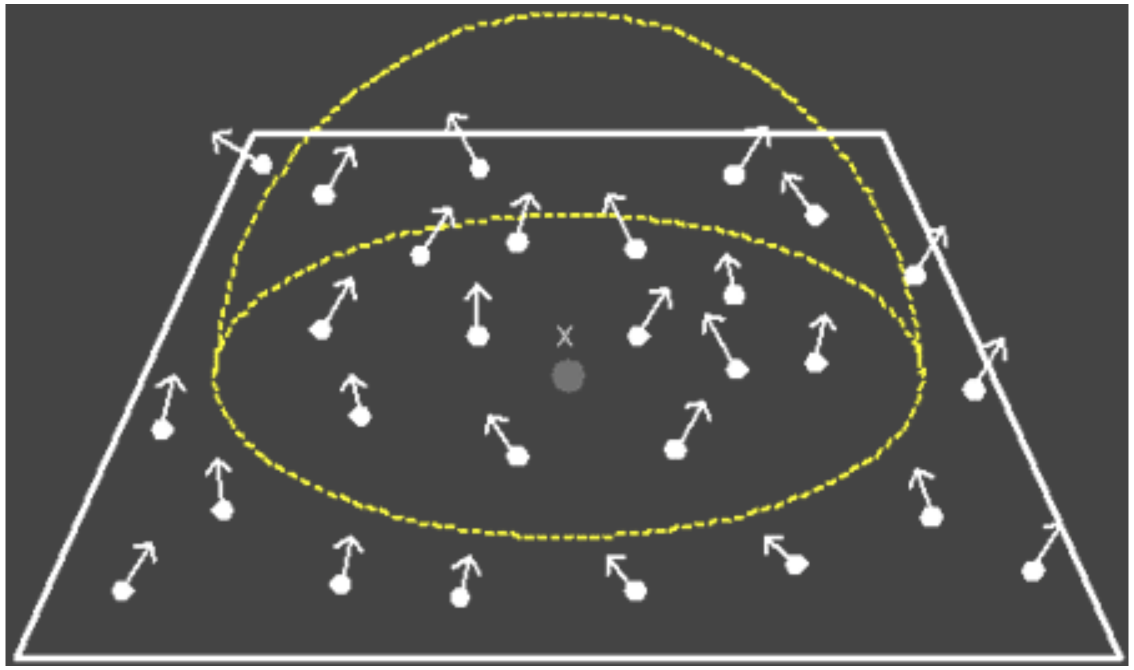
- A two pass global illumination algorithm
  - First Pass - photon tracing:
    - Casting photons from the light source
    - Storing photon positions in the "photon map",
  - Second Pass – rendering (radiance estimate):
    - the shading of pixels is estimated from the photon map

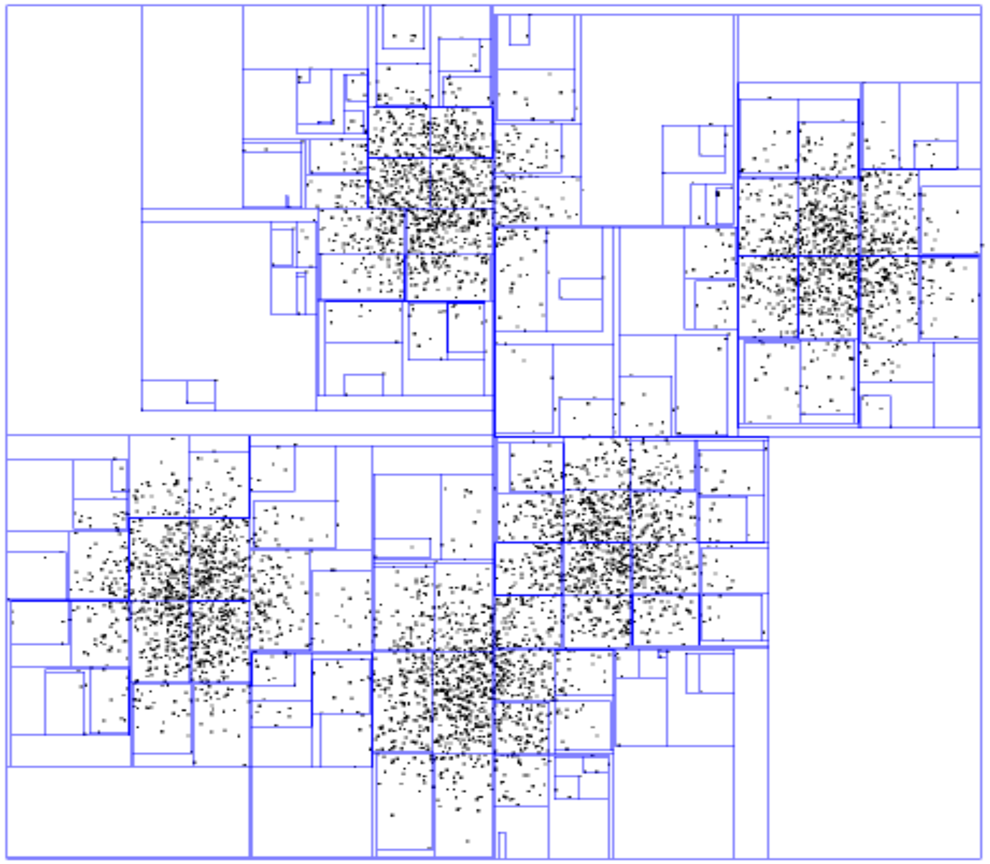






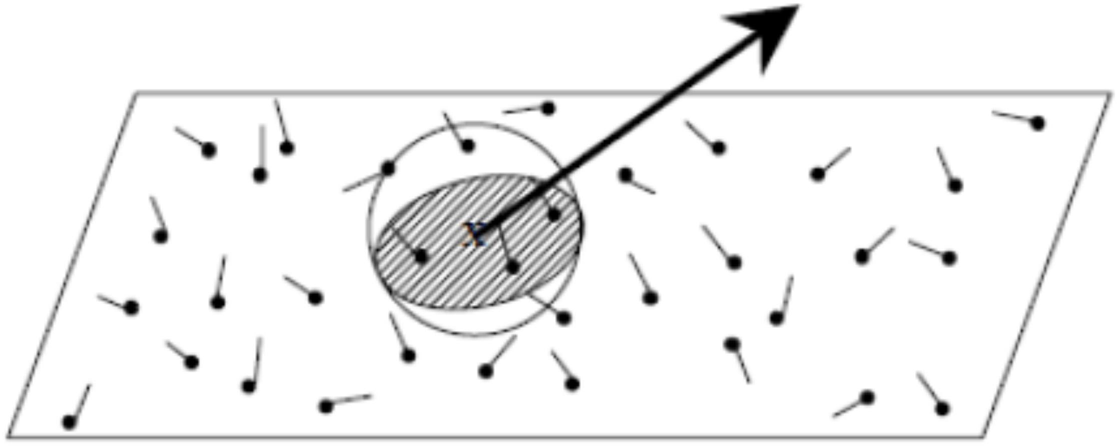
Second Pass – Rendering





- The radiance estimate can be written by the following equation

$$L_r(x,\vec{\omega}) = \sum_{p=1}^{N} f_r(x,\overrightarrow{\omega_p},\vec{\omega}) \frac{\Delta\Phi_p(x,\overrightarrow{\omega_p})}{\Delta A}$$

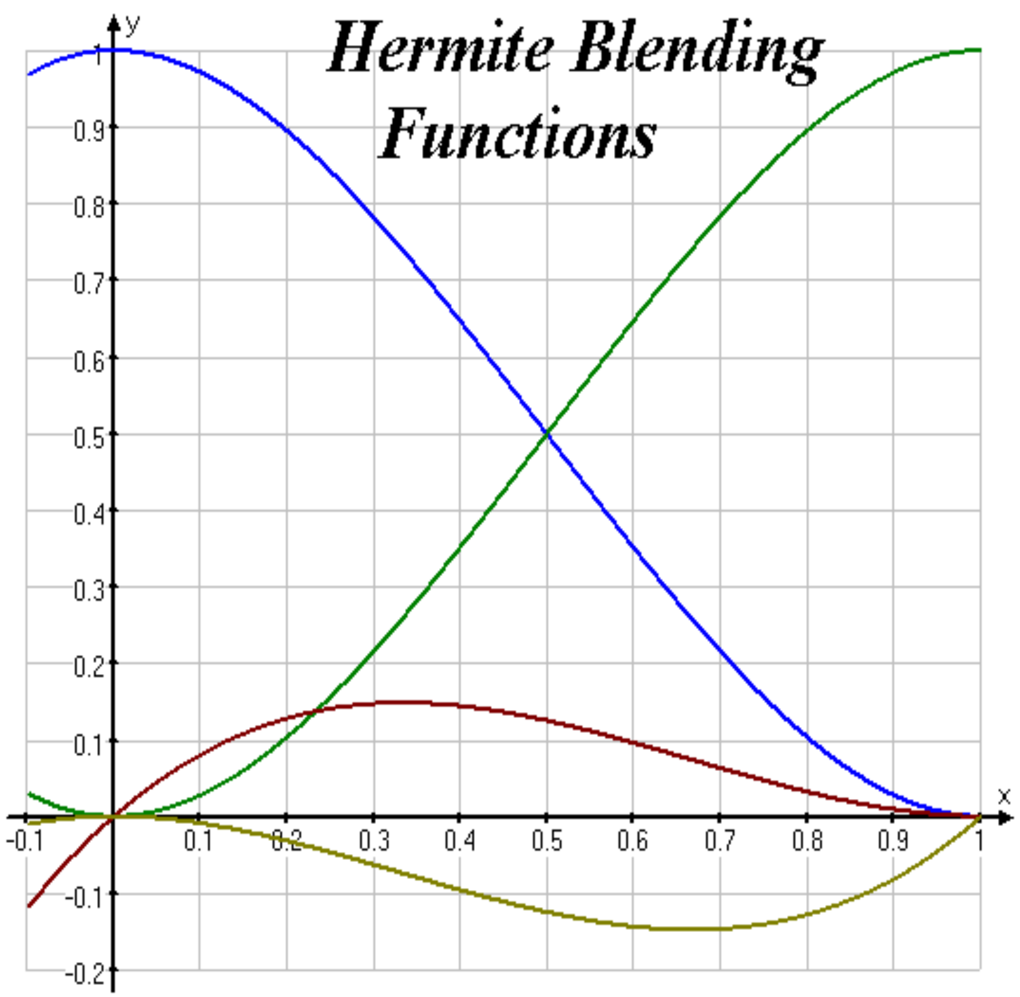


Hermite curves

Hermite Specification

$$x(t) = (2x_0 + x_0' - 2x_1 + x_1')t^3 + (-3x_0 - 2x_0' + 3x_1 - x_1')t^2 + x_0't + x_0$$

$$X(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & -2 & 1 \\ -3 & -2 & 3 & -1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_0' \\ x_1 \\ x_1' \end{bmatrix}$$

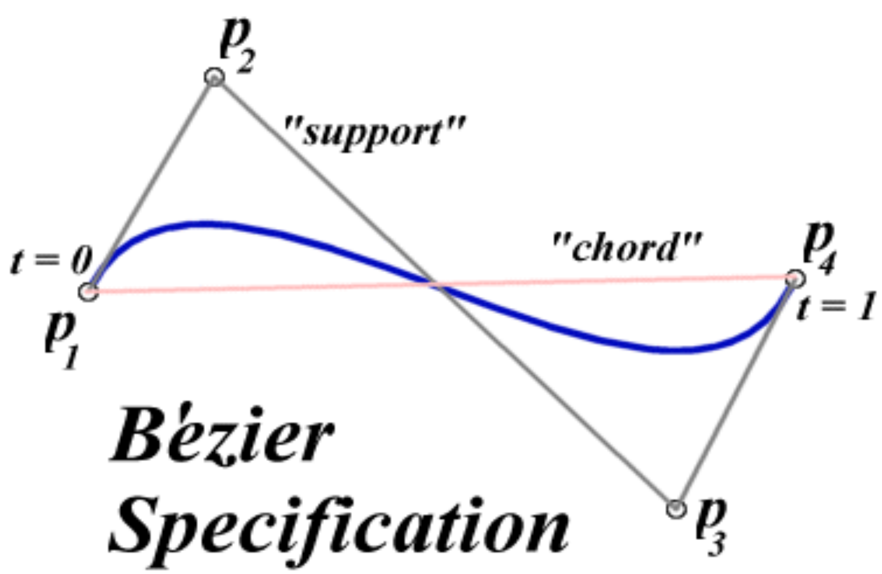


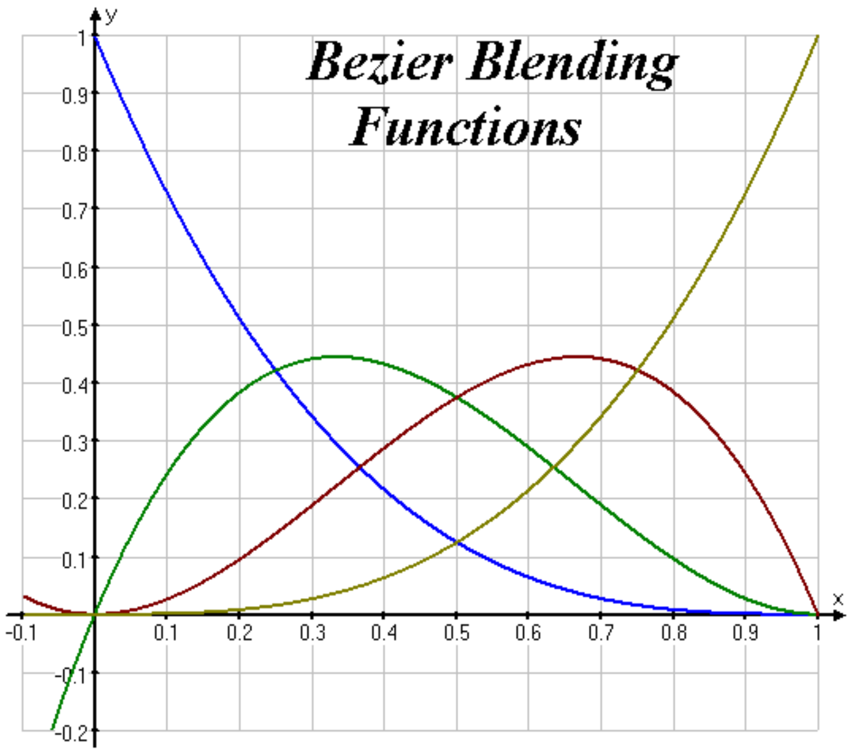
$$X(t) = \underline{\underline{(2t^3 - 3t^2 + 1)}}x_0 + \underline{\underline{(t^3 - 2t^2 + t)}}x_0' + \underline{\underline{(-2t^3 + 3t^2)}}x_1 + \underline{\underline{(t^3 - t^2)}}x_1'$$

Bézier Curves

$$X(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

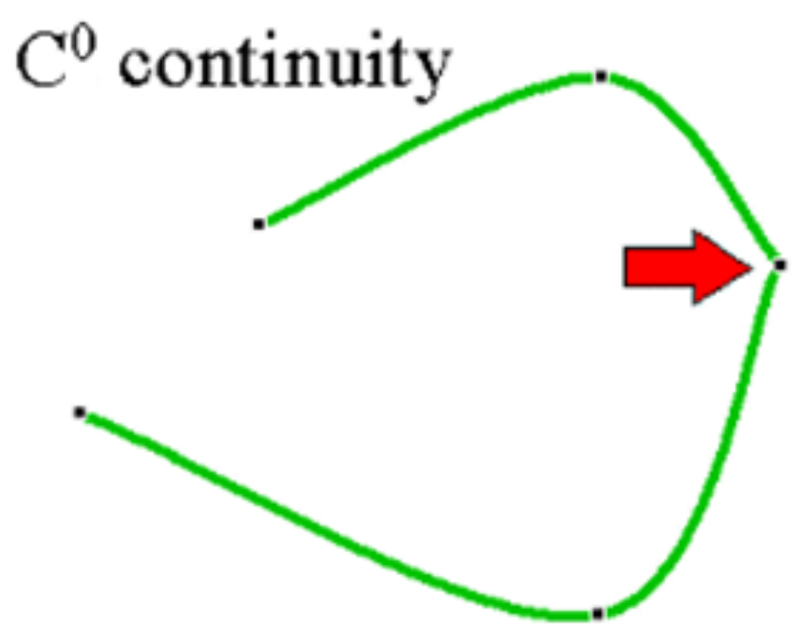
$$X(t) = (-t^3 + 3t^2 - 3t + 1)q_0 + (3t^3 - 6t^2 + 3t)q_1 + (-3t^3 + 3t^2)q_2 + (t^3)q_3$$

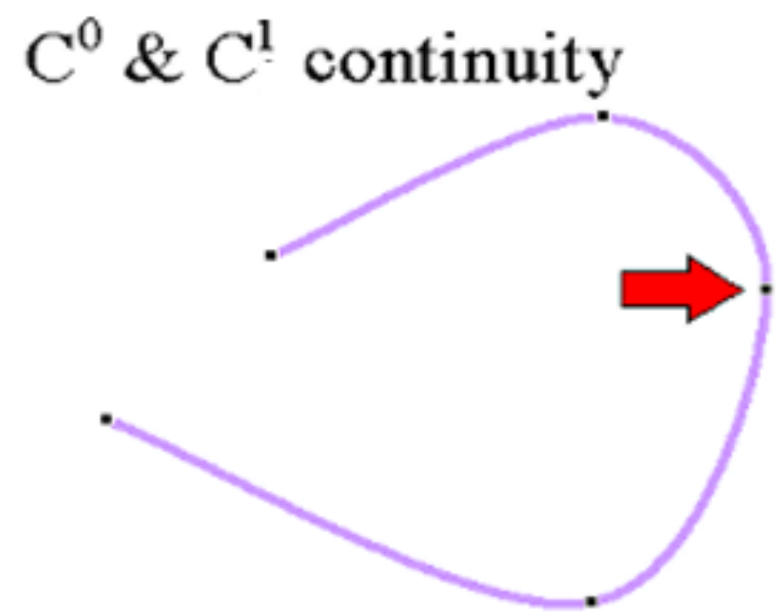


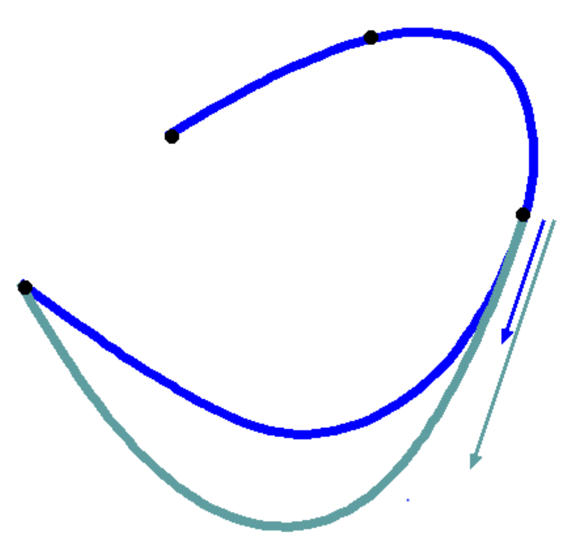


Continuity between curve segments

- If the direction and magnitude of $\frac{d^n X(t)}{dt^n}$ are equal at the join point, the curve is called C^ncontinuous







Uniform cubic B-splines

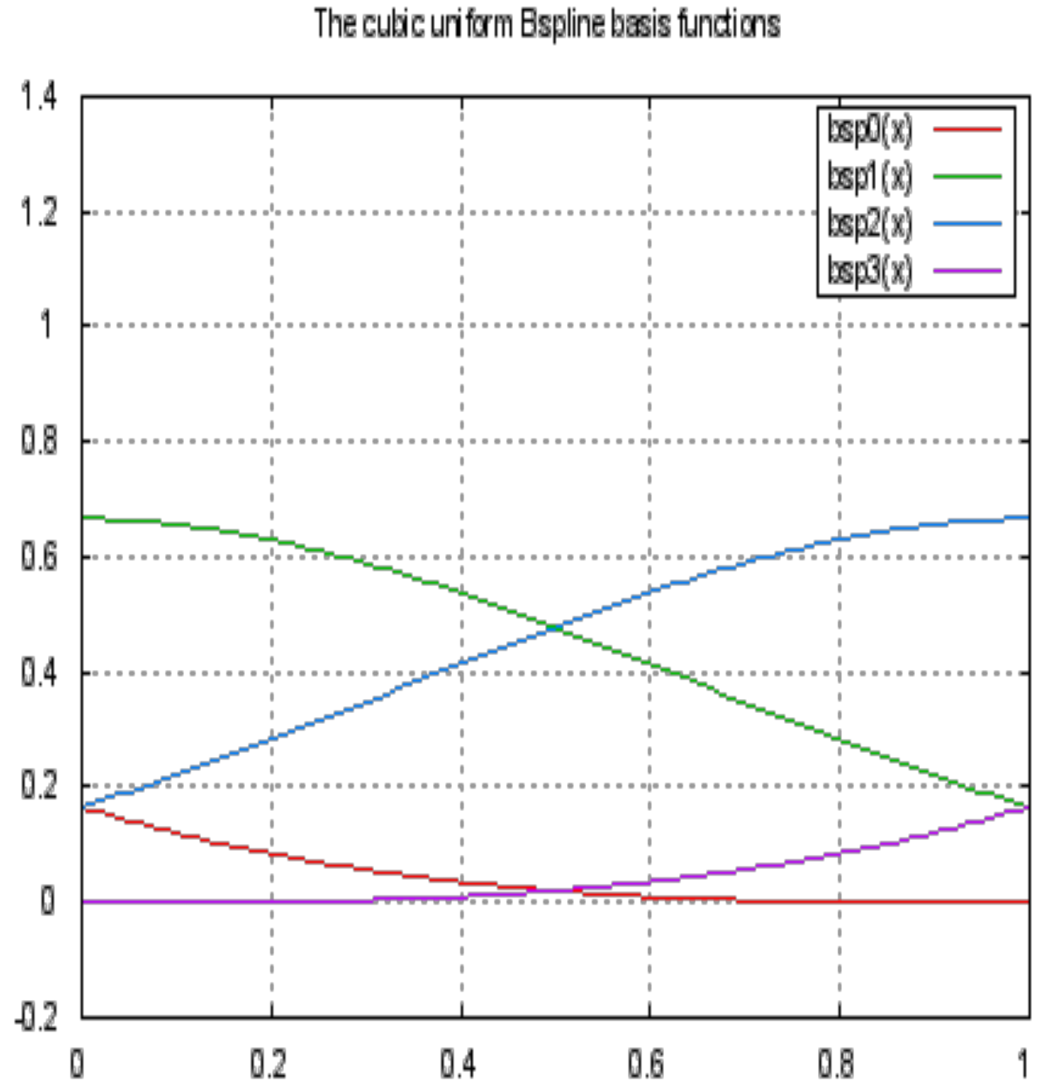
$$X(t) = \mathbf{t}^{\mathbf{T}}\mathbf{M}\mathbf{Q}^{(\mathbf{i})} \qquad for \quad t_i \le t \le t_{i+1}$$

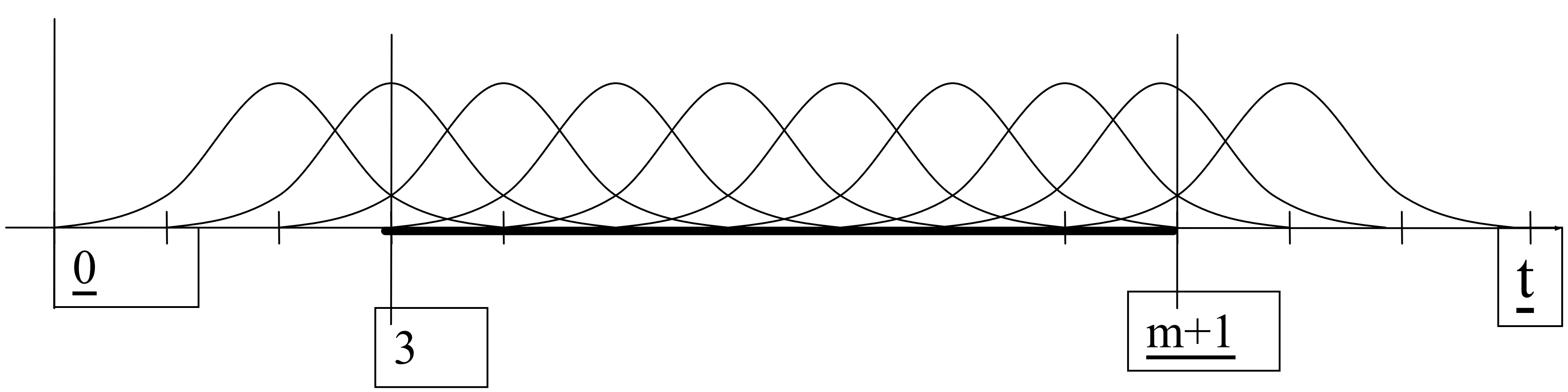
$$where \qquad \mathbf{Q}^{(\mathbf{i})} = (x_{i-3}, \dots, x_i)$$

$$\mathbf{M} = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix},$$

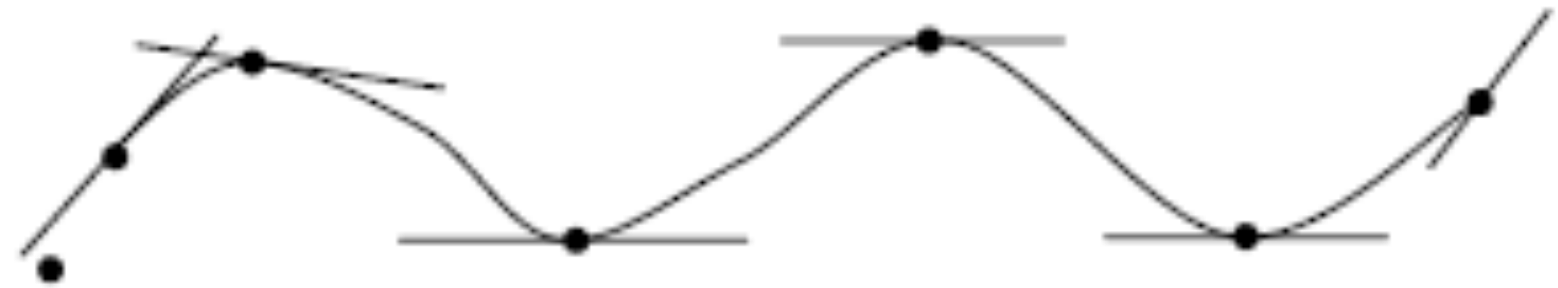
$$\mathbf{t}^{\mathbf{T}} = \left((t-t_i)^3, (t-t_i)^2, t-t_i, 1\right)$$

$$t_i : \text{knots}, \;\; 3 \le i$$





Catmull-Rom Spline



Hermite Specification

$$P^i(t) = T \cdot M_{CR} \cdot G_B$$

$$= \frac{1}{2} \cdot T \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{bmatrix}$$

Bicubic patches

- Now we assume q_i to vary along a parameter s,
- $Q_i(s,t) = t^T M[q_1(s), q_2(s), q_3(s), q_4(s)]$
- $q_i(s)$ are themselves cubic curves
- Bicubic patch has degree 6

$$x(s,t) = t^T.M_B.q_x.M_B^T.s$$

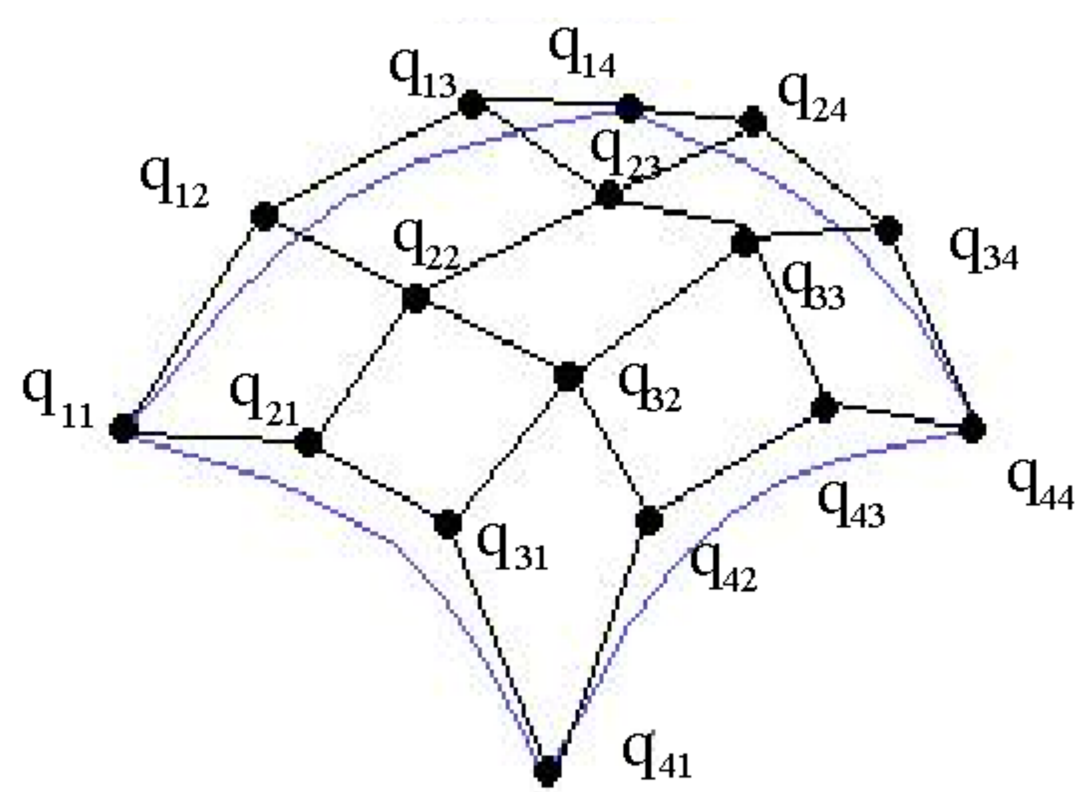
q_x *is* 4×4 *array of* x *coords*

$$y(s,t) = t^T.M_B.q_y.M_B^T.s$$

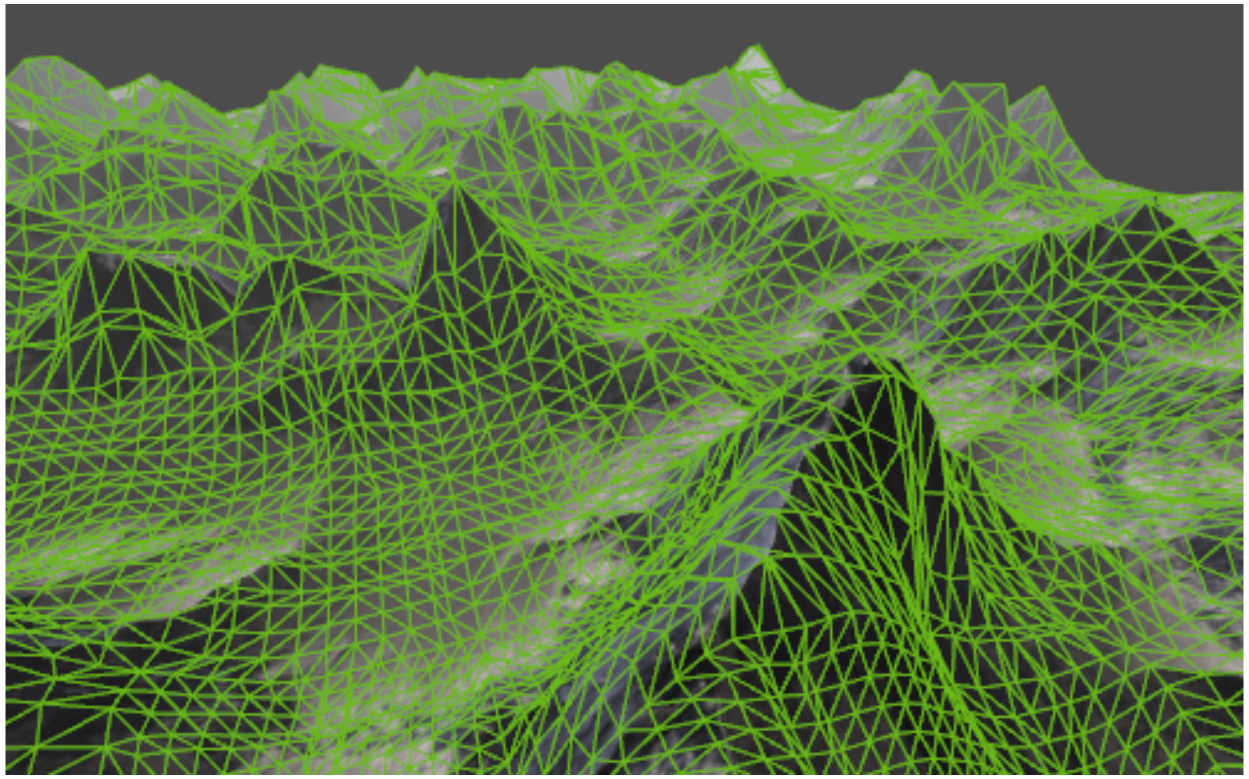
q_y *is* 4×4 *array of* y *coords*

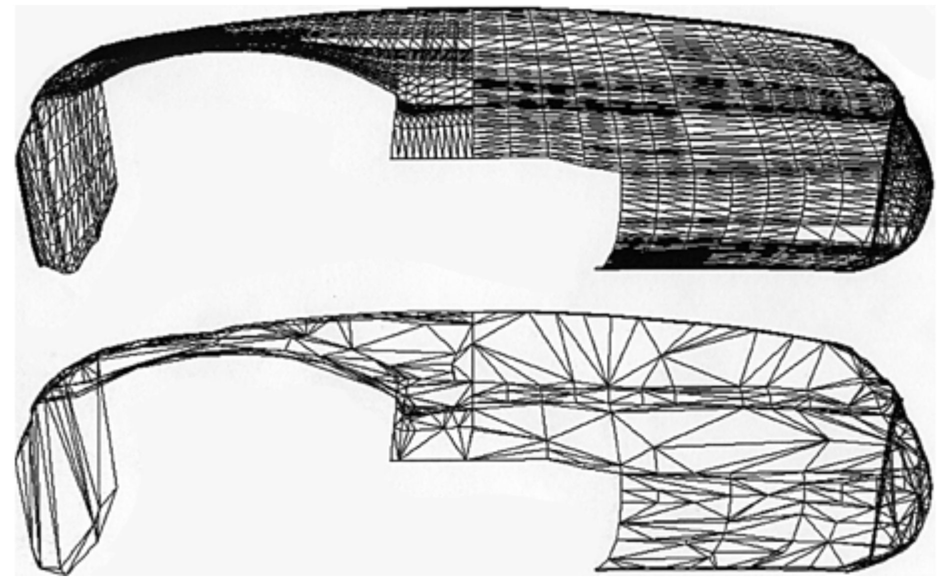
$$z(s,t) = t^T.M_B.q_z.M_B^T.s$$

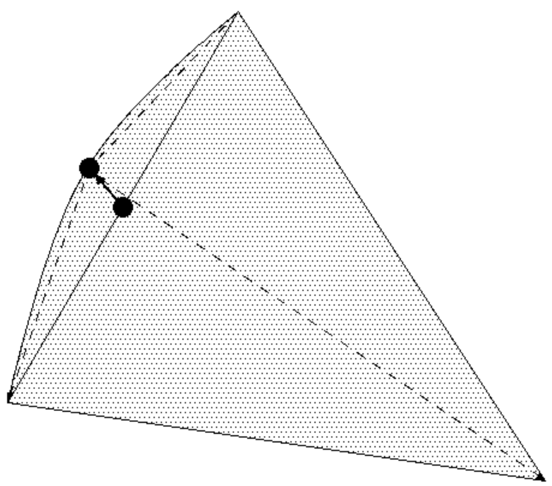
q_z *is* 4×4 *array of* z *coords*

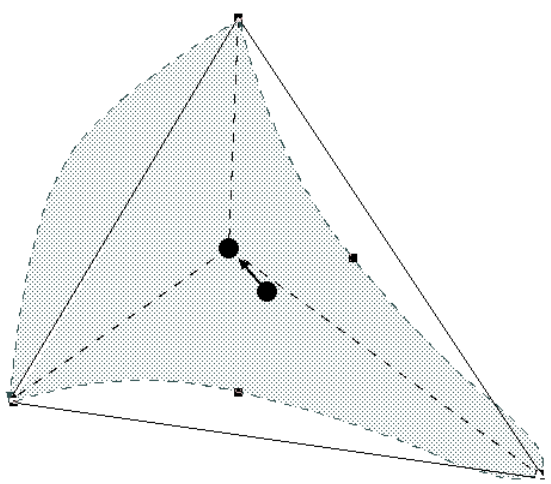


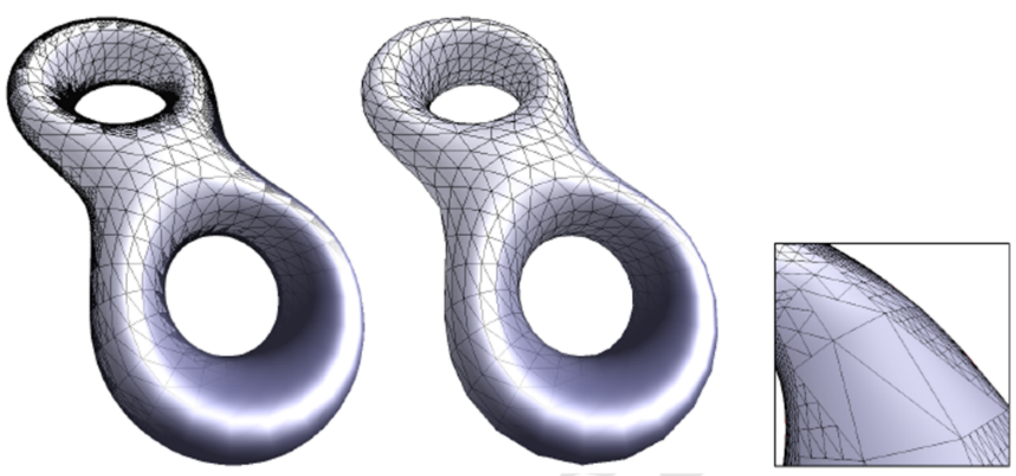
Tessellation











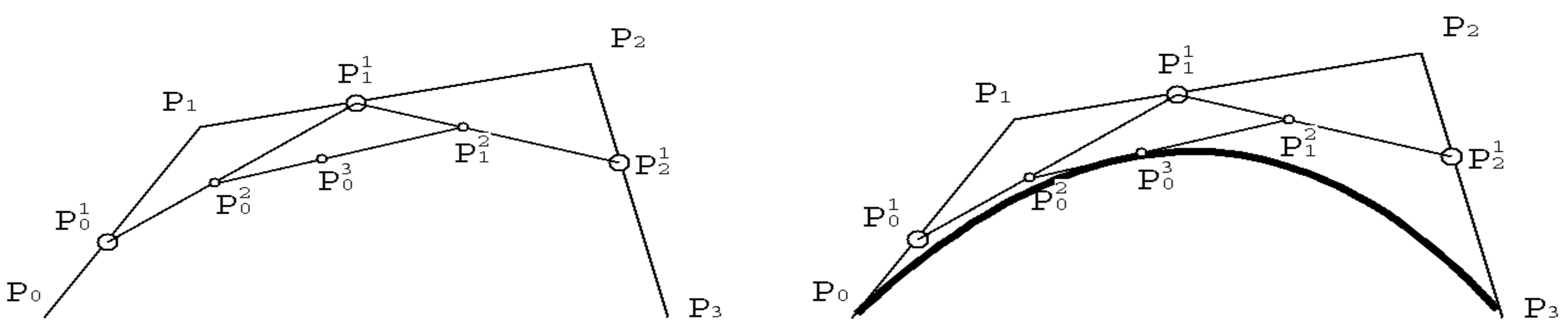
de Casteljau's algorithm

- Given the control points $P_1, \ldots, P_n$ and the parameter value $0 \leq t \leq 1$
- Repeat the following procedure:

$$P_i^r(t) = (1-t)P_i^{r-1}(t) + tP_{i+1}^{r-1}(t)$$

$$P_i^0(t) = P_i$$

- Then $P_0^n(t)$ is the point with parameter value t on the Bézier curve



B-Splines: general form

a B-spline of order *k* (polynomial of degree k-1) is a parametric curve composed of a linear combination of basis B-splines $B_{i,k}$:

$P_i \, (i = 0, \ldots, m)$ are the control points

$$p(t) = \sum_{i=0}^{m} P_i B_{i,k}(t)$$

Knots: $t_0 \le t_1 \le \ldots \le t_{k+m}$ - the knots subdivide the domain of the B-spline curve into a set of knot spans $[t_i, t_{i+1})$

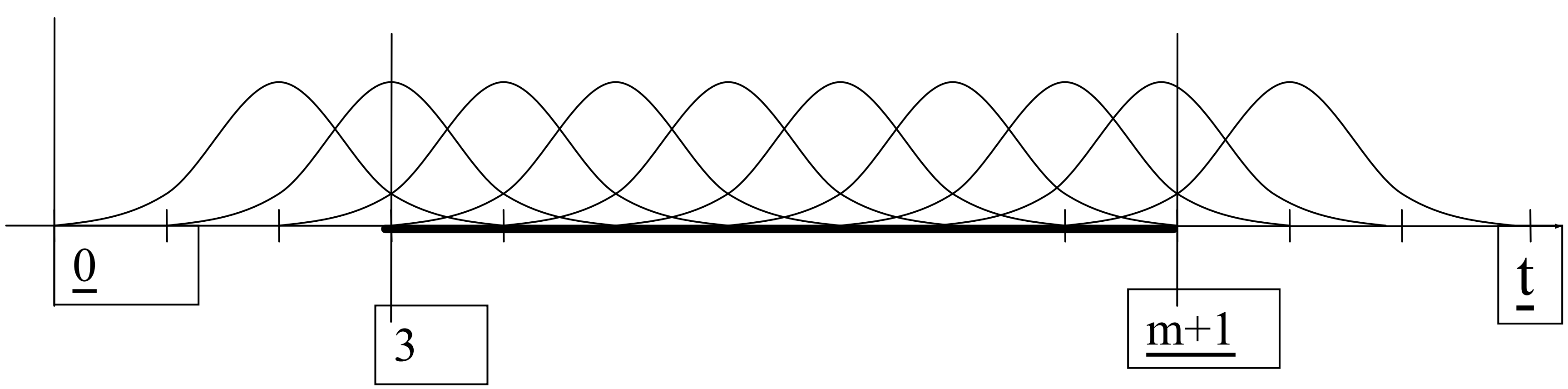
The B-splines can be defined by

$$B_{i,1}(t) = \begin{cases} 1, \ \mathrm{t_i} \le \mathrm{t} < \mathrm{t_{i+1}} \\ 0, \ \text{otherwise} \end{cases}$$

$$B_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k-1} - t_i} B_{i+1,k-1}(t)$$

B-spline terms

- Order k : the number of control points affecting the sampled value
- Degree $k-1$: the degree of the basis function polynomial
- Control points $P_i \quad i = (0, \ldots, m)$
- Knots $t_j \quad (j = 0, \ldots, n)$
- An important rule: $n - m = k$
- The domain of the curve is $t_{k-1} \leq t \leq t_{m+1}$
- Below, k=4, m=9, domain is $t_3 \leq t \leq t_{10}$

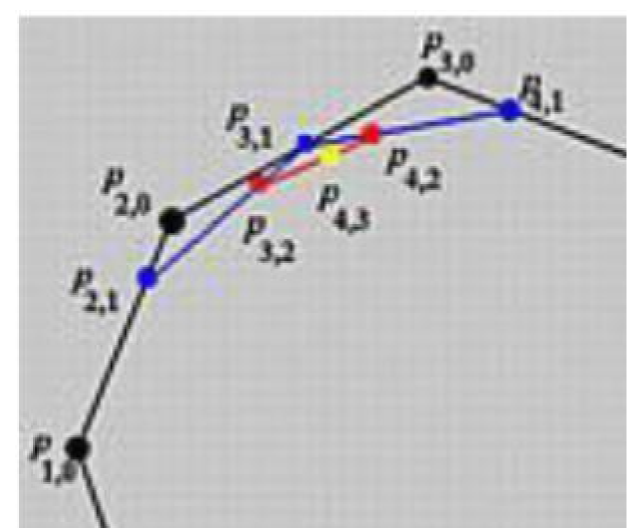


de Boor's algorithm

- B-spline version of de Casteljau's algorithm
- A precise method to evaluate the curve
- Starting from control points and parameter value t, recursively solve:

$$\mathbf{P}_i^r = (1 - a_{i,r})\mathbf{P}_{i-1}^{r-1} + a_{i,r}\mathbf{P}_i^{r-1}$$

$$a_{i,r} = \frac{t - t_i}{t_{i+k-1-r} - t_i}$$



Knot insertion

- If the new knot t is inserted into the span $[t_j, t_{j+1})$,

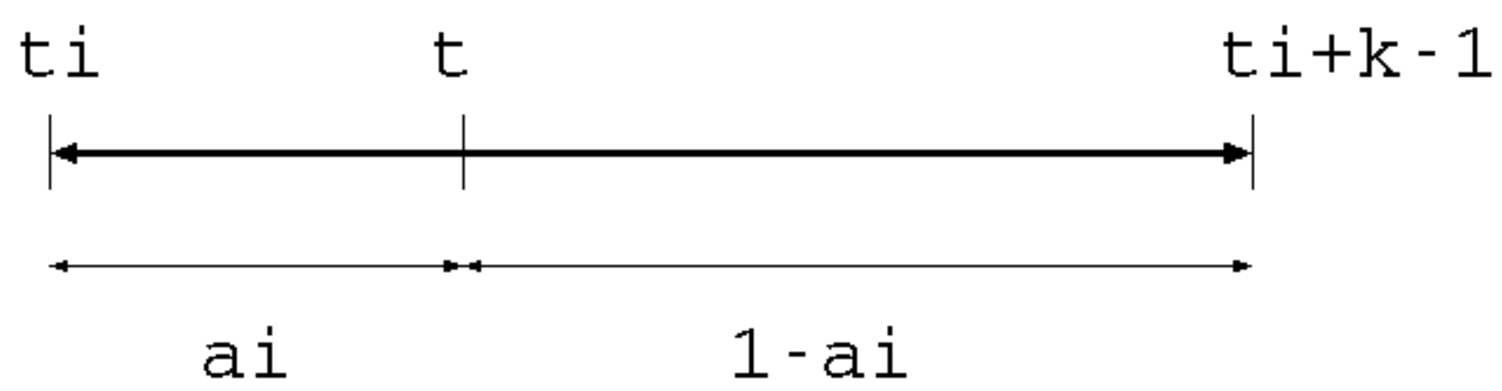
the new control points can be computed by

$$\mathbf{Q_i} = (1 - a_i)\mathbf{P_{i-1}} + a_i\mathbf{P_i}$$

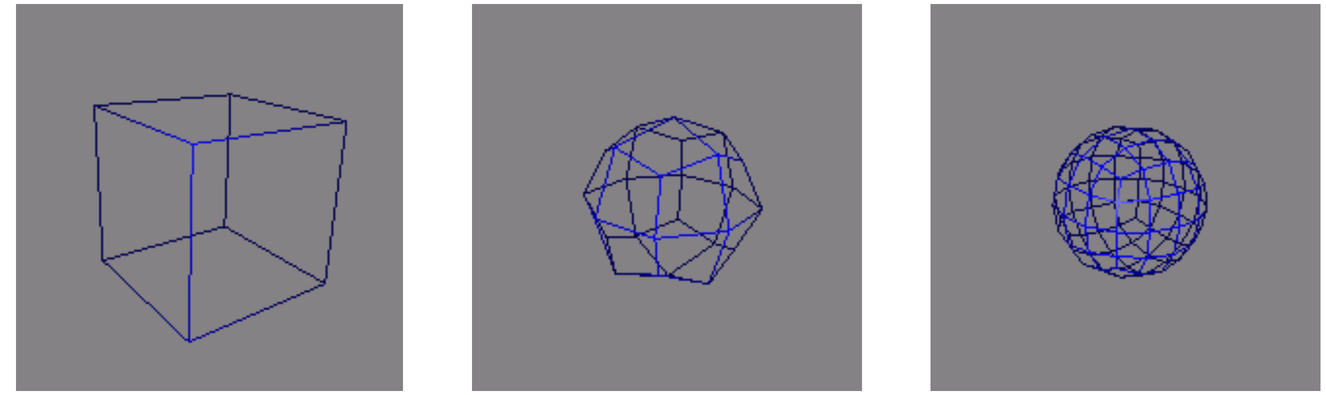
where Qi is the new control point and a_i is computed by

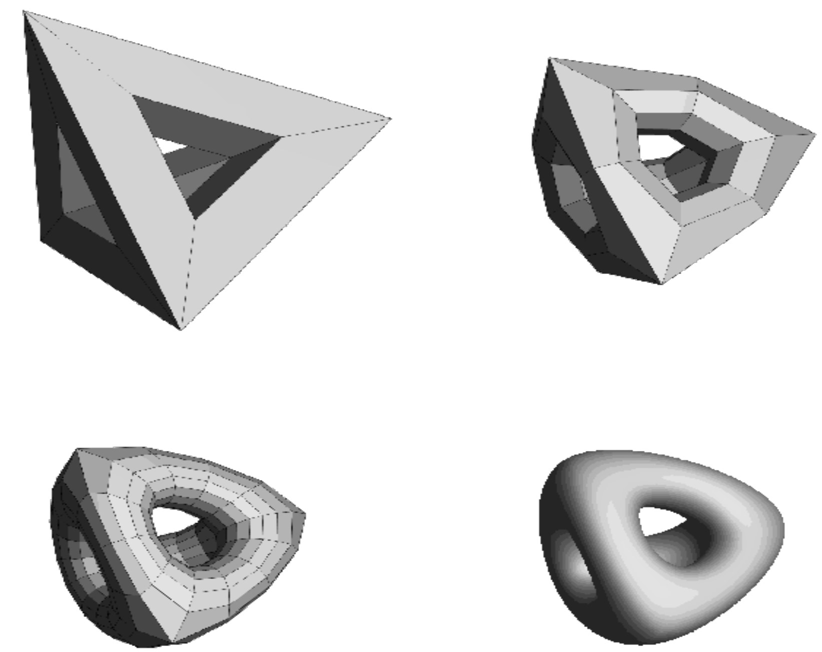
$$a_i = \frac{t - t_i}{t_{i+k-1} - t_i} \quad \text{for } j\text{-}k + 2 \le i \le j$$

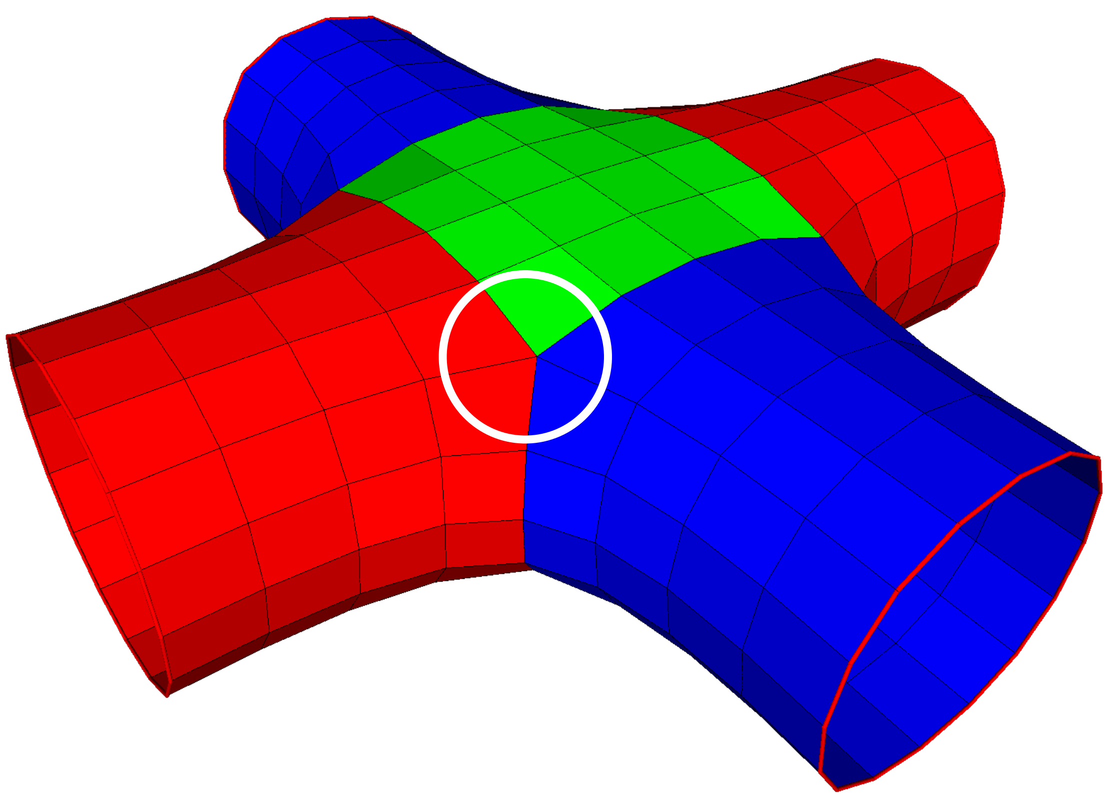
Pj-k+1, Pj-k+2, ..., Pj-1, Pj is replaced with Pj-k+1, Qj-k+2, ..., Qj-1, Qj ,Pj.

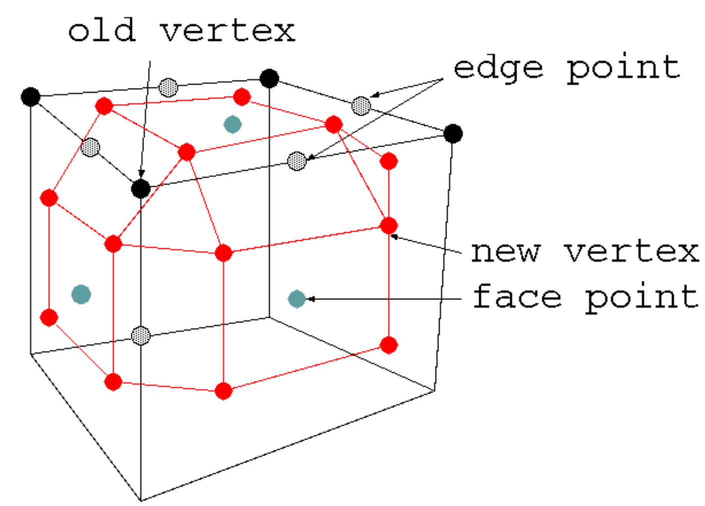


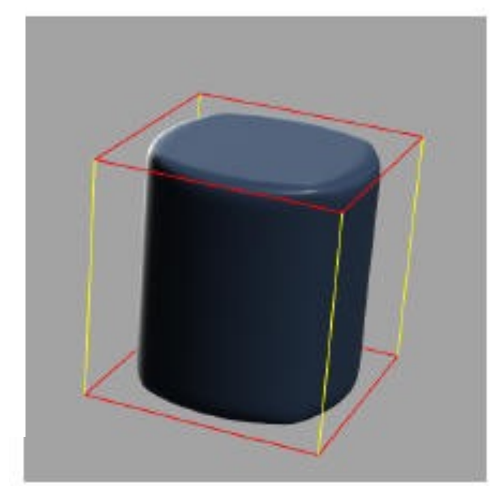
Subdivision Surfaces











(d)

What next?

- Practical low level implementation details: Real Time Rendering book http://www.realtimerendering.com

- Building demos - ideas:

  - Pixel shaders (https://open.gl/)

  - Spherical harmonic lighting (http://www.cs.columbia.edu/~cs4162/slides/spherical-harmonic-lighting.pdf)

  - Real-time radiosity (progressive refinement)

  - Photon mapping (http://graphics.ucsd.edu/~henrik/)

  - Real-time ray casting/tracing