PV227 GPU programming

Marek Vinkler

Department of Computer Graphics and Design

Points and vectors

- points (before projection) are quadruples: $(x, y, z, 1.0)$,
  - can be transformed with a 4×4 matrix,
- vectors are also quadruples: $(x, y, z, 0.0)$,
  - can be transformed with a 4×4 or 3×3 matrix.

Normal transformation

- points are transformed to eye space with modelview matrix,
- vectors constructed from points (e.g. $P_2 - P_1$) are also transformed with this matrix,
- normals are not!

Normal transformation

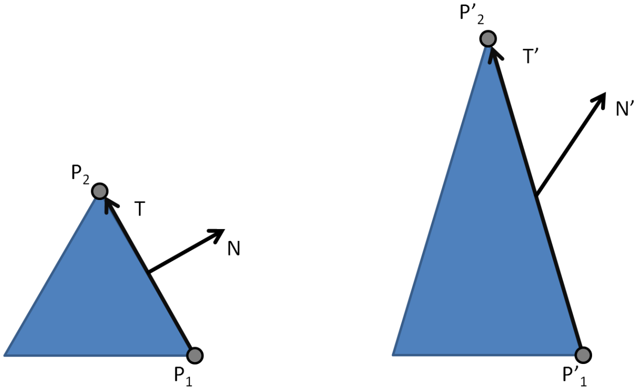


Figure: Taken from lighthouse3d.com

Normal transformation

- caused by non-uniform scale,
- we need another matrix (N) for transforming normal $\vec{n}$,
- M is the modelview matrix, $\vec{t}$ is tangent vector $P_2 - P_1$ and I is identity.

$$(M \times \vec{t}) \cdot (N \times \vec{n}) = 0$$
$$(M \times \vec{t})^T \times (N \times \vec{n}) = 0$$
$$\vec{t}^T \times M^T \times N \times \vec{n} = 0$$

Normal transformation

$$\vec{t}^T \times M^T \times N \times \vec{n} = 0$$

$$\vec{t} \cdot \vec{n} = 0 \Rightarrow \vec{t}^T \times \vec{n} = 0 \Rightarrow M^T \times N = I$$

$$\begin{aligned} M^T \times N &= I \\ (M^T)^{-1} \times M^T \times N &= (M^T)^{-1} \\ N &= (M^T)^{-1} \end{aligned}$$

Normal transformation

- N is inverse transpose of M (3×3 submatrix of M),
- for orthogonal matrices $A^T = A^{-1}$, rotation is orthogonal,
- M is orthogonal $M = (M^T)^{-1} \Rightarrow N = M$.

Renormalization

- normals must be of unit length,
- can be destroyed by normal transformation → must be normalized in vertex shader,
- interpolation can also destroy vector length → must be normalized in fragment shader.

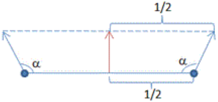


Figure: Taken from lighthouse3d.com

Lighting

- computation of light's interaction with surfaces,
- huge cheat,
- ambient, diffuse and specular light,
- flat, gouraud and phong lighting,
- directional, point and spot light,
- no shadow, no bouncing of light.

Ambient lighting

- approximates lighting after infinite number of bounces,
- homogeneous,
- prevents black areas that look unnatural,
- usually chosen as fraction of the diffuse (material) color,
- $I = K_a$.

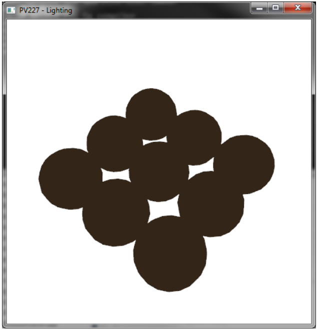


Figure: Ambient spheres

Directional light

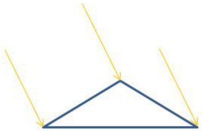


Figure: Taken from lighthouse3d.com

- far away light,
- defined by a direction vector (position is irelevant),
- can represent e.g. the sun.

Gouraud shading

- per vertex lighting,
- interpolation of vertex colors,
- unable to capture lighting details inside polygons.

Diffuse lighting

- simulate light's interaction with perfectly diffuse material,
- angle dependent,
- significant color component,
- $I = \cos(\alpha) \times K_d$.

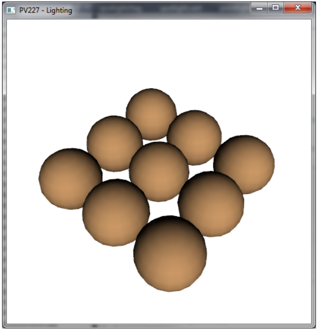


Figure: Diffuse spheres

Diffuse lighting

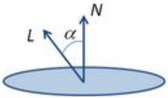


Figure: Taken from lighthouse3d.com

- amount of incoming light diminishes with increasing angle,
- $\cos(\alpha) = \frac{\vec{L} \cdot \vec{N}}{|\vec{L}| \times |\vec{N}|}$,
- normalized vectors: $I = (\vec{L} \cdot \vec{N}) \times K_d$,
- all vectors must be in same space, usually defined in world space, computation in camera space.

Flat shading

- per vertex lighting,
- no interpolation,
- unable to capture smooth changes in light intensity.

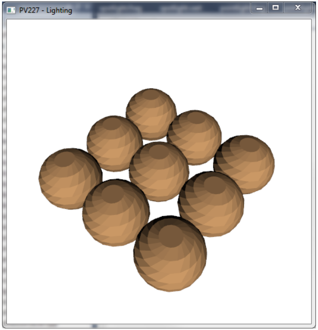


Figure: Flat shading

Combined lighting

- light from various sources can be combined (added),
- combination of ambient and diffuse prevents black areas,
- $I = K_a + \cos(\alpha) \times K_d$,
- value should not be outside the $[0.0, 1.0]$ range.

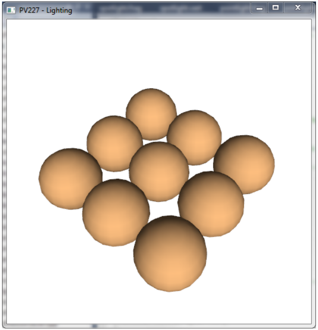


Figure: Ambient + Diffuse spheres

Specular lighting

- simulate light's interaction with reflective material,
- angle dependent,
- highlight of the light's color, not material color,
- $I = \cos(\beta)^s \times K_s$, s controls size of the highlight.

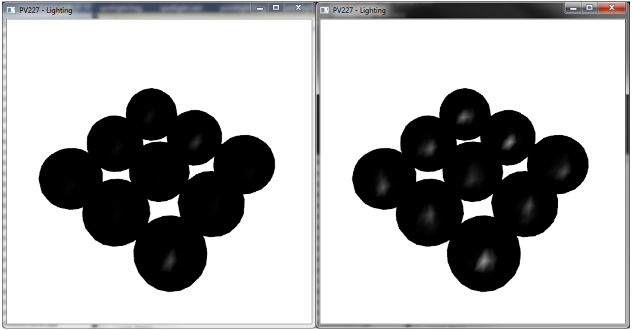


Figure: Specular spheres (Phong *vs* Blinn-Phong)

Phong lighting

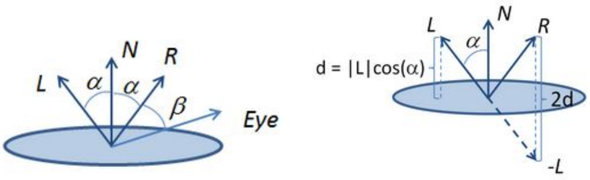


Figure: Taken from lighthouse3d.com

- amount of reflected light diminishes with increasing angle,
- $\vec{R} = -\vec{L} + 2(\vec{N} \cdot \vec{L}) \times \vec{N}$,
- $\cos(\beta) = \vec{R} \cdot \vec{Eye}$,
- all vectors must be in same space, normalized.

Blinn-Phong lighting

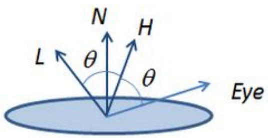


Figure: Taken from lighthouse3d.com

- amount of reflected light diminishes with increasing angle,
- $\vec{H} = \vec{L} + \vec{Eye}$,
- $\cos(\beta) = \vec{H} \cdot \vec{N}$,
- all vectors must be in same space, normalized.

Combined lighting

- light from various sources can be combined (added),
- ambient, diffuse and specular form the baseline lighting,
- $I = K_a + \cos(\alpha) \times K_d + \cos(\beta)^s \times K_s$,
- value should not be outside the $[0.0, 1.0]$ range.

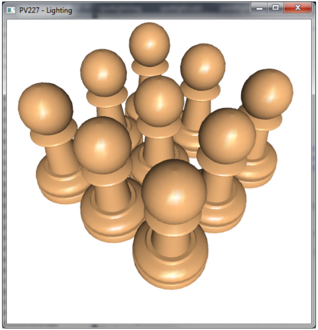


Figure: Ambient + Diffuse + Specular pawns

Phong shading

- per pixel lighting,
- smooth lighting including details,
- interpolation of vertex attributes (normal, eye, light).

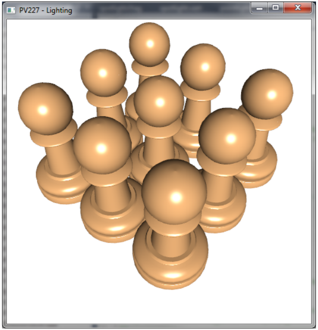
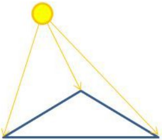


Figure: Per pixel lighting pawns

Point light

- light source inside the scene,
- defined by a position vector (all directions),
- can represent e.g. a lightbulb.



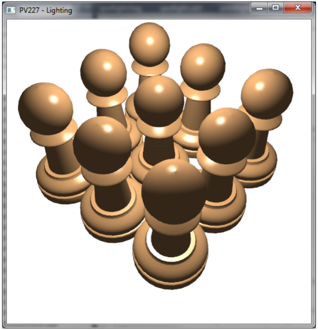
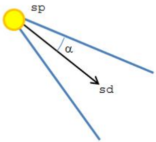


Figure: Point light pawns, Taken from lighthouse3d.com

Spot light

- light source inside the scene,
- only a directed cone is illuminated,
- defined by a position vector, direction vector and angle,
- can represent e.g. a flashlight.



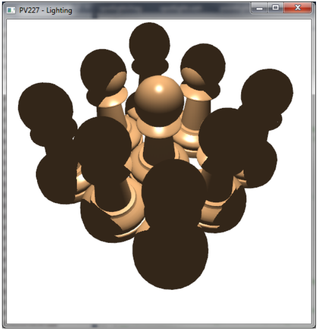


Figure: Spot light pawns