4 Class8-10 Illumination and Shading

1. Global vs. Local Illumination
   Global: Models indirect illumination and occlusions Local: Only models direct illumination
2. Irradiance
   $$E = \int_{\Omega} I(x,\omega) cos\theta dx$$
   Note: $I(x,\omega)$ is the light intensity arriving from all directions and entering the hemisphere Ω over unit serface area.
   Also we only care the vertical(normal) part of the light, we dismiss all lights parallel to the surface by using $\cos\theta$
3. Simplified lighting:
   Assume all lights are distant-point light.
   - Source have **uniform** intensity distribution
   - Neglect distance fallout
   - Direction to source is constant within scene
   - Using 2 parameters to define a light:
     $direction(x, y, z)$ vector from surface to light source
     $intensity(r, g, b)$ of the light
4. specular reflection and diffuse reflection
   - Color shift by attenuation of RPG components for all reflection
   - Specular Reflection Model(View-Dependent):
     $$L_j(V) = L_e \cdot K_s \cdot (V \cdot R)^{spec}$$
     $$R = 2(N \cdot L)N - L$$
     Note: V and R should be normalized.
     Direction: Reflection occurs mainly in the "mirror" R direction, but there is some spread in similar directions V.
     $spec$ controls the distribution of intensity about R. Higher value of $spec$ make the surface smoother
     K_s controls the color attenuation of Surface.
   - Diffuse Reflection Models(View-Independent):
     $$L_j = L_e \cdot K_d(L \cdot N)$$
     Direction: All **output** directions are the same. But we only care vertical **input** light.
     L and N should be normalized.
     K_d is the surtface attenuation component.
   - Ambient Light
     $$L_j = L_a \cdot K_a$$
     Direction: All input and output directions are the same.
     Only one ambient light is needed and allowed.
   - Complete Shading Equation:
     $$Color = (K_s \sum L_e \cdot (V \cdot R)^{spec}) + (K_d \sum L_e \cdot (L \cdot N)) + (L_a \cdot K_a)$$
5. Detail about HW4(Lighting Implementation)

- $\vec{L}$ denotes the direction to a infinity-far point-light source
- $\vec{E}$ denotes the camera direction. If camera is far away, $\vec{E}$ is constant(In HW4.)
- $\vec{N}$ is specified at triangle vertices.
- $\vec{R}$ must be computed for each lighting calculation (at **a point**).
  Calculation of $\vec{R}$:
  $$\vec{R} = 2(\vec{N} \cdot \vec{L})\vec{N} - \vec{L}$$
  Avoiding sqrt-root in this calculation
- Choosing a Shading Space: Wee need all $\vec{L}, \vec{E}, \vec{N}, \vec{R}$ in some affine(pre-perspective) space
  Suggest use **Image Space** for HW4.
  Model space is also a reasonable choice since Normal vectors are already in that space. This is most **efficient**!
- **Image Space Lighting (ISL)**
  Create a Transformation stack from model space to image space.
  Need to **normalized the Scale and delete translation** for each matrix, **only maintain the rotation**, before push into this stack!
- **Check** the sign of $\vec{N} \cdot \vec{E}$ and $\vec{N} \cdot \vec{L}$:
  Both positive: Compute lighting model.
  Both negative: **Flip normal($\vec{N}$)** and compute lighting model.
  Different sign: Skip it.
- **Check** the sign of $\vec{R} \cdot \vec{E}$: If negative, set to 0.
- **Check** color overflow(> 1.0): Set to 1.
- **Compute Color** at all pixels:
  Per Face - flat shading
  Per Vertex - interpolate vertex colors, Gouraud Shading(specular highlights are undersampled, aliased).
  Per Pixel - interpolate normals, Phong Shading (Expensive computation, but better sampling)
  Set **Shading Modes Parameter** for different lighting calculation.
- **Pitfall** in **Phong** Interpolation:
  Need to **normalize** the interpolation normal vector.

4.1 Class10: Something More About Shading

1. Non-Uniform Scaling:
   A non-uniform scaling alters the relationship between the surface orientation and the Normal Vector.
   So we **cannot** use the same matrix M for transformation of the Normals and the vertex coordinates.
   We can fix this by using a different transformation $Q = f(M)$ for transforming the Normals.
2. How to create a matrix for Normals:
   In HW4, We create a matrix **dismiss all** scale matrix.
   For Detail:
   As the definition of Normals:
   $$\vec{N}^T \cdot \vec{P} = 0$$
   After include the transform matrix:
   $$(Q\vec{N})^T \cdot (M\vec{P}) = 0$$

By Definition of Matrix Multiplyer:

$$\vec{N}^T \cdot Q^T \cdot M \cdot \vec{P} = 0$$

Since we already know $\vec{N}^T \cdot \vec{P} = 0$ we only need the inner part equal to identity matrix:

$$Q^T \cdot M = I, Q = (M^{-1})^T$$

Note that: If we only used uniform scaling: $S = I$ after normalization.

If we compute Q for each M pushed on the X_{im} transform stack, the resulting X_n stack has Q and therefore allows non-uniform scaling.

3. Model Space Lighting(MSL):

   Only need to transform Global lighting parameters once per models.

   Also need to transform Eye/camera direction into model space.

5 Class 11-13: Texture Mapping

5.1 Screen-Space Parameter Interpolation

1. In Z-buffer interpolation, we know that linear interpolation for z is **wrong in image space**, we need to interpolate in **perspective space**.
2. Accurate interpolation of RGB color or Normal vectors should also take perspective into account.
   But we can ignore the color and normal interpolation error.
3. Interpolation for **Texture Function**: checkerboard Example: Using Linear Interpolation for $u\&v$ is also wrong!
4. How to compute perspective-correct interpolation of u, v at each pixel.
   - For each parameter P, we used P^s to denote the value in perspective space.
   - Note that: For Z interpolation $V_z^s = \frac{V_z}{\frac{V_z}{d}+1} = \frac{V_z \cdot d}{V_z + d}$
   - Rescale V_z^s to $V_z^s \in [0, Z_{max}]$

$$V_z^s = \frac{V_z \cdot d}{V_z + d} \cdot (\frac{Z_{max}}{d}) = \frac{V_z \cdot Z_{max}}{V_z + d}$$

   - We can also get the invert equation:

$$V_z = \frac{V_z^s \cdot d}{Z_{max} - V_z^s}$$

   - For parameter from image space to perspective space:

$$P^s = \frac{P}{\frac{V_z}{d} + 1} = \frac{Pd}{V_z + d}$$

   - Also we can get inver equation:

$$P = \frac{P^s(V_z + d)}{d}$$

   - We don't have V_z but we already calculated V_z^s in HW2, so we can used that:

$$P^s = \frac{P}{(\frac{V_z^s}{Z_{max} - V_z^s} + 1)}$$

$$P = P^s \cdot \frac{V_z^s}{Z_{max} - V_z^s} + 1$$

   - Note that we only have V_z^s and Z_{max} in this equation that we already know the value, we don't need to care d and some other parameter.
   - We used $V_z' = \frac{V_z^s}{Z_{max} - V_z^s}$ to simplify the equation:

$$P^s = \frac{P}{V_z' + 1}$$

$$P = P^s \cdot (V_z' + 1)$$

5. The Step for Parameter interpolation:
   Get V_z^p for each vertex.
   Transform P to perspective space P^s for each vertex.
   Interpolate V_z^p for each pixel.
   Interpolate P^s for each pixel.
   Transform P^z back to P by using V_z^p for each pixel.

5.2 Texture

1. Scale u, v to Texture Image Size:
   (u, v) coords range over $[0, 1]$
   2D Image is a pixel array of $xs - 1, ys - 1$
   But $u * (xs - 1)$ might not be Integer so we need to interpolate the color for non-Integer (u, v) coordinate from nearest 4 Integer point.

$$Color(p) = (1 - s)(1 - t)A + s(1 - t)B + stC + (1 - s)tD$$

2. For Phong Shading, using texture function $f(u, v)$ to replace k_d and k_a
3. For Gouraud Shading, using $f(u, v)$ to replace all k_s, k_d and k_a
4. Procedural Texture
5. Bump Texture: Alter normals at each pixel to create bump.
6. Noise Texture: 3D Noise Volume, Tubulence
7. Evironment(Reflection) Mapping: Cube Map
   Problems: Sampling problems and high curvature problems

5.3 Implementation Of Texutre(HW5)

1. Step1: Texture coordinates: surface point $\rightarrow (u, v)$
   Input: vertex in image space
   Output: (u, v)
2. Step2: $(u, v) \rightarrow$ RGB color
   Input: (u, v) Output: RGB color from image LUT
3. Interpolation of (u, v) need to be in perspective space.
4. Interpolation of 4-corner for non-Integer (u, v) is needed.

6 Class14-15 Antialiasing

6.1 The Source of Aliasing

1. Quantization error arise from insufficient accuracy of sample
2. Aliasing error arise from insufficient samples
3. Nyquist Theorem: Sample at least twice the rate of highest frequency present in the signal.
   f(t) filtered for cutoff freq ω_F(Remove high frequencies before sampling)
   Sample Rate $\frac{1}{T_0}$ is greater than $2\omega_F$
   Reconsturct(interpolate) with *sinc* function
4. Solution: Band-limit the input signal before sampling.

6.2 Implement Antialiasing(HW6)

1. Antialiasing by jitter supersampling
2. Sample a pixel several with different center and weight