CS 580 – Discussion
HW 4
Week 7

Yijing Li

Slides modified from: Matthias Hernandez

Regrading for HW 1-2-3

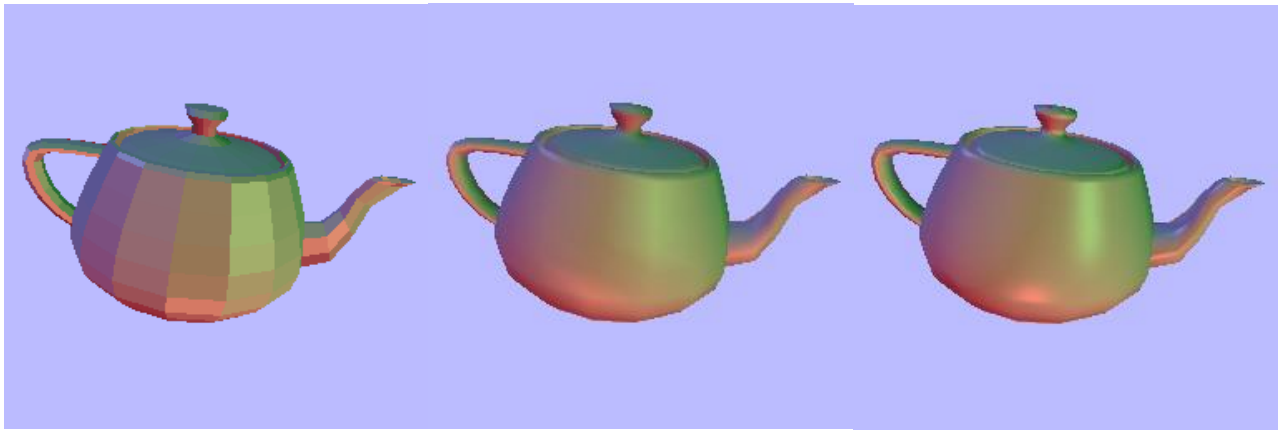
- In your HW 4 submission
  - Write a README.txt file stating which homeworks you want regraded.

- Do not re-submit HW 1, 2 or 3
- Please do not ask for regrading if you got 10/10

HW 4 goal: add shading

HW 3

HW 4



Flat shading

Gouraud shading

Phong shading

Application4

New parameters:

- lighting direction
- shading mode
- lighting properties
- normal at each vertex

Lighting

```
nameListLights[0] = GZ_DIRECTIONAL_LIGHT;
valueListLights[0] = (GzPointer)&light1;
nameListLights[1] = GZ_DIRECTIONAL_LIGHT;
valueListLights[1] = (GzPointer)&light2;
nameListLights[2] = GZ_DIRECTIONAL_LIGHT;
valueListLights[2] = (GzPointer)&light3;
status |= GzPutAttribute(m_pRender, 3, nameListLights, valueListLights);

nameListLights[0] = GZ_AMBIENT_LIGHT;
valueListLights[0] = (GzPointer)&ambientlight;
status |= GzPutAttribute(m_pRender, 1, nameListLights, valueListLights);
```

Shading Mode

```
/*
* Select either GZ_COLOR or GZ_NORMALS as interpolation mode
*/
    nameListShader[1]  = GZ_INTERPOLATE;
#if 0
    interpStyle = GZ_COLOR;         /* Gouraud shading */
#else
    interpStyle = GZ_NORMALS;       /* Phong shading */
#endif
```

Material

```
    valueListShader[1] = (GzPointer)&interpStyle;
    nameListShader[2]  = GZ_AMBIENT_COEFFICIENT;
    valueListShader[2] = (GzPointer)ambientCoefficient;
    nameListShader[3]  = GZ_SPECULAR_COEFFICIENT;
    valueListShader[3] = (GzPointer)specularCoefficient;
    nameListShader[4]  = GZ_DISTRIBUTION_COEFFICIENT;
    specpower = 32;
    valueListShader[4] = (GzPointer)&specpower;

status |= GzPutAttribute(m_pRender, 5, nameListShader, valueListShader);
```

Normal

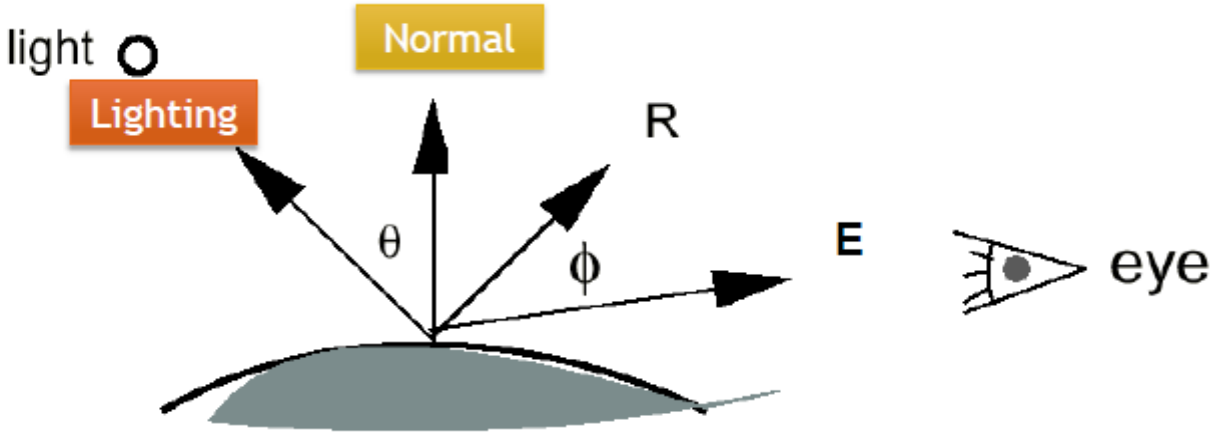
```
/*
* Set the value pointers to the first vertex of the
* triangle, then feed it to the renderer
* NOTE: this sequence matches the nameList token sequence
*/
   valueListTriangle[0] = (GzPointer)vertexList;
   valueListTriangle[1] = (GzPointer)normalList;
   GzPutTriangle(m_pRender, 2, nameListTriangle, valueListTriangle);
```

Passing the parameters to the renderer

- status |= GzPutAttribute(m_pRender, 1, nameListLights, valueListLights);
- status |= GzPutAttribute(m_pRender, 5, nameListShader, valueListShader);

```
#ifndef GZRENDER
#define GZRENDER
typedef struct {            /* define a renderer */
  GzDisplay     *display;
  GzCamera      camera;
  short         matlevel;           /* top of stack - current xform */
  GzMatrix      Ximage[MATLEVELS];  /* stack of xforms (Xsm) */
  GzMatrix      Xnorm[MATLEVELS];   /* xforms for norms (Xim) */
  GzMatrix      Xsp;                /* NDC to screen (pers-to-screen) */
  GzColor       flatcolor;          /* color state for flat shaded triangles */
  int           interp_mode;
  int           numlights;
  GzLight       lights[MAX_LIGHTS];
  GzLight       ambientlight;
  GzColor       Ka, Kd, Ks;
  float         spec;       /* specular power */
  GzTexture     tex_fun;    /* tex_fun(float u, float v, GzColor color) */
}  GzRender;
#endif
```

Lighting equation



Material

$$C = (Ks \sum_L [Ie\ (R \bullet E)^s]) + (Kd \sum_L [Ie\ (N \bullet L)]) + (Ka\ Ia)$$

Shading Mode

GZ_FLAT : flat shading
GZ_COLOR : Gouraud Shading
GZ_NORMALS : Phong shading

Material

	n_s	k_d	k_s
Metal	large	Small, color of metal	Large, color of metal
Plastic	medium	Medium, color of plastic	Medium, white
Planet	0	varying	0

```
GzColor diffuseCoefficient = {0.2f, 0.2f, 0.2f};
GzColor specularCoefficient = { 0.8f, 0.8f, 0.8f };
specpower = 10;
```

```
GzColor diffuseCoefficient = {0.7f, 0.7f, 0.7f};
GzColor specularCoefficient = { 0.2f, 0.2f, 0.2f };
specpower = 5;
```

```
GzColor diffuseCoefficient = {0.4f, 0.4f, 0.4f};
GzColor specularCoefficient = { 0.0f, 0.0f, 0.0f };
specpower = 0;
```

Lighting representation

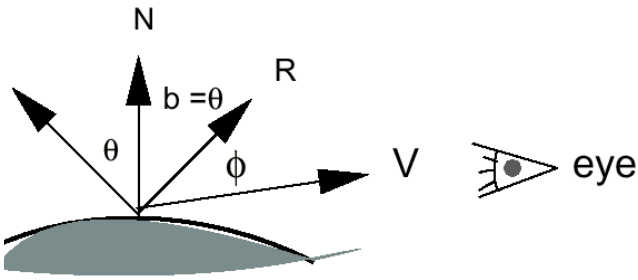
- Specify lights in image space (attached to camera)
  - Direction light (LSI)
  - Ambient light

Direction Color

```
/* Light */
GzLight light1 = { {-0.7071, 0.7071, 0}, {0.5, 0.5, 0.9} };
GzLight light2 = { {0, -0.7071, -0.7071}, {0.9, 0.2, 0.3} };
GzLight light3 = { {0.7071, 0.0, -0.7071}, {0.2, 0.7, 0.3} };
GzLight ambientlight = { {0, 0, 0}, {0.3, 0.3, 0.3} };
```

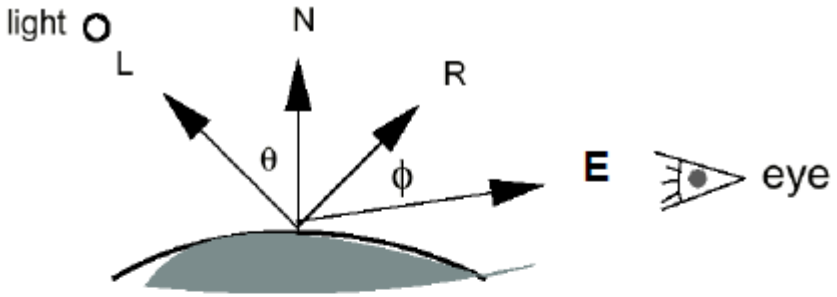
- LSI is the direction to an infinitely-far point-light source
  - from the scene to the light
  - constant in whole scene and specified by the application

```
#ifndef GZLIGHT
#define GZLIGHT
typedef struct  GzLight
{
    GzCoord   direction;
    GzColor      color;
} GzLight;
#endif
```



Normals

- Normals are provided per vertex **in model space**
  - Need to convert **to image space** with Xwm – Xiw
    - Keep a **2nd stack** for normals (Xnorm)
    - Xsp and Xpi are Identity
    - ONLY rotations: no translation/scaling



$$C = (K_s \sum_L [I_e (R \bullet E)^s]) + (K_d \sum_L [I_e (N \bullet L)]) + (K_a I_a)$$

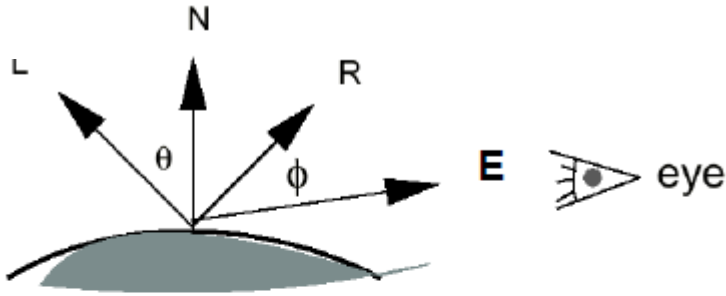
All terms of shading eq are known (and fixed) for the entire triangle, except for *Normals*

Computing color: orientations

- R • E calculations must be clamped to zero to maintain [0, 1] bounded range
- Sign of N • L and N • E
  - Both positive : compute lighting model
  - Both negative : flip normal and compute lighting model on backside of surface

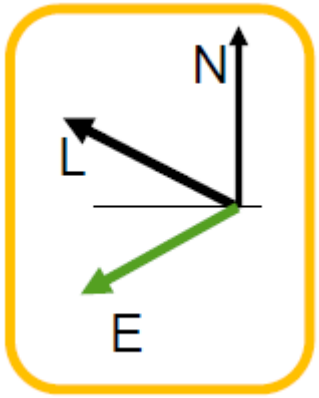
Flipped N

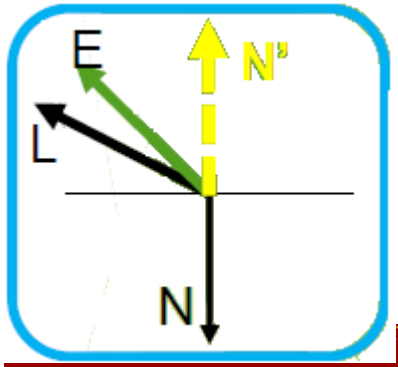
  - Each has different sign : light and eye are on opposite sides of the surface so the light contributes zero

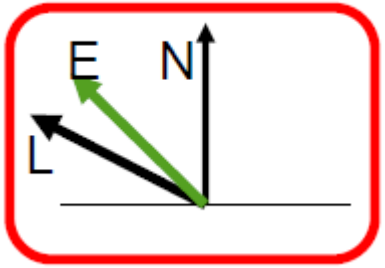


$$C = (K_s \sum_L [I_e (R \bullet E)^s]) + (K_d \sum_L [I_e (N \bullet L)]) + (K_a I_a)$$

Skip it







Shading types

- App sets GZ_INTERPOLATE flag
  - GZ_FLAT : flat shading (default -- so HW3 should still work)

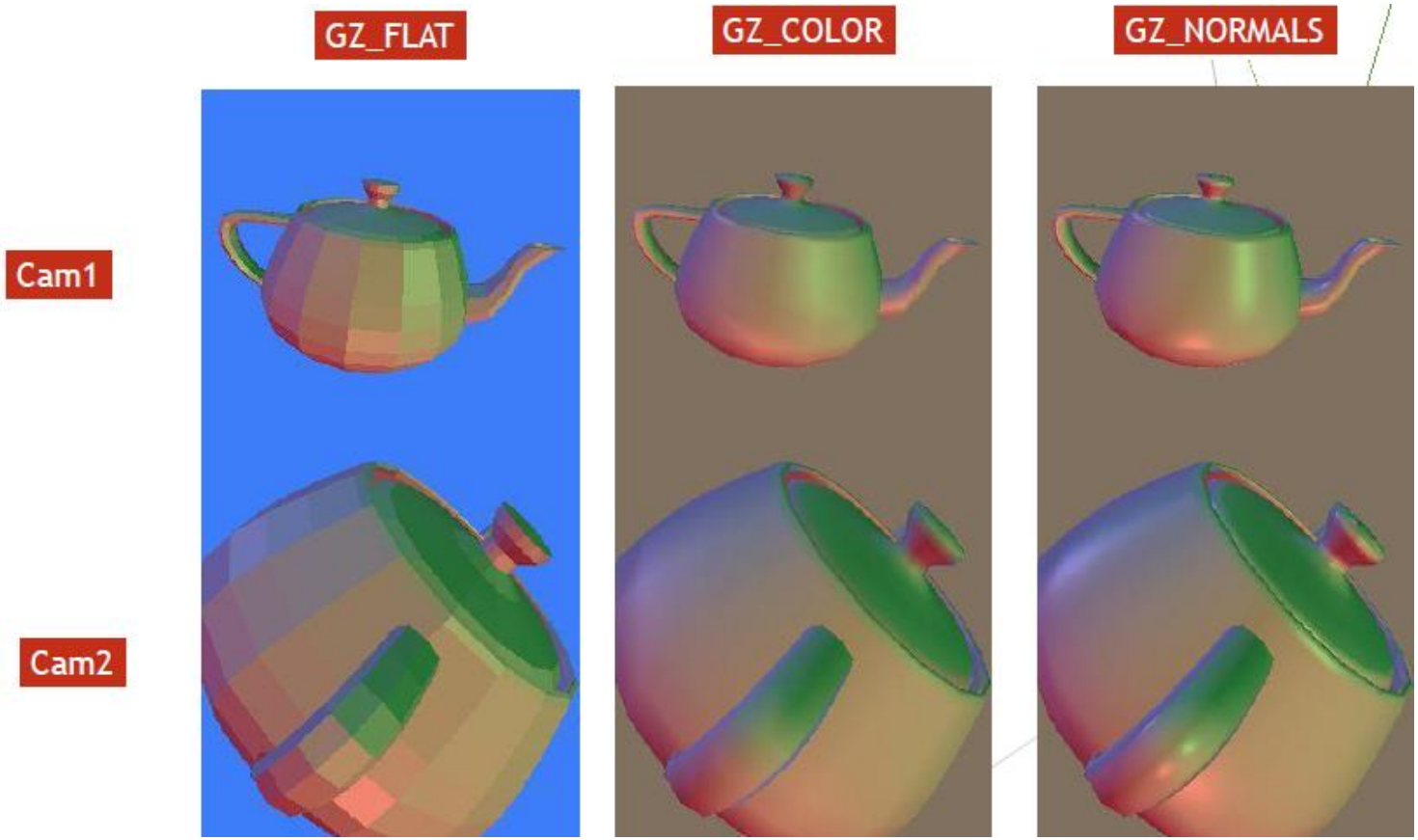
  - GZ_COLOR : vertex shading and color interpolation (Gouraud)
  - GZ_NORMALS : normal interpolation and pixel shading (Phong)

- Paint the whole triangle with the color from the first vertex (using the normal of the triangle)

- Compute color at three vertices
- **Interpolate color** on the triangle

- **Interpolate normal** on the triangle
- Compute color at every pixel
- Note: Normalize the length of normal!!

Shading types: results



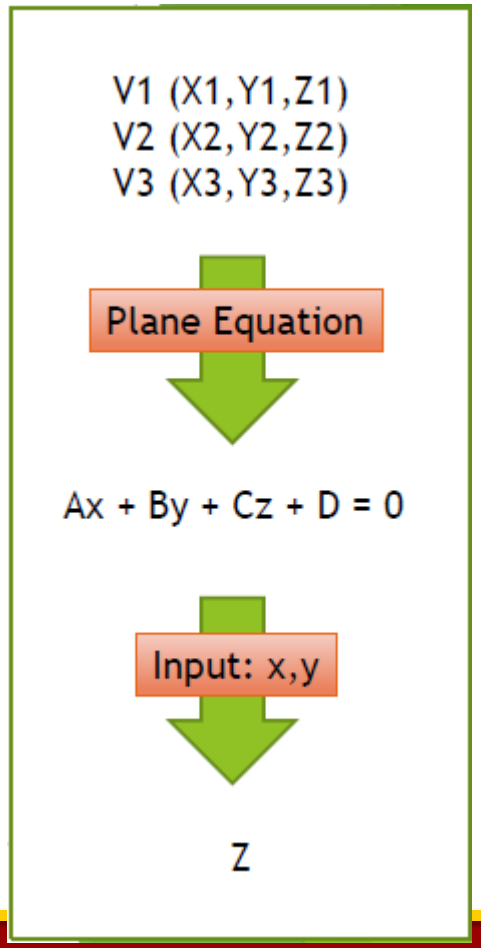
Interpolating

In HW2: interpolating Z

- A general 3D plane equation has four terms:

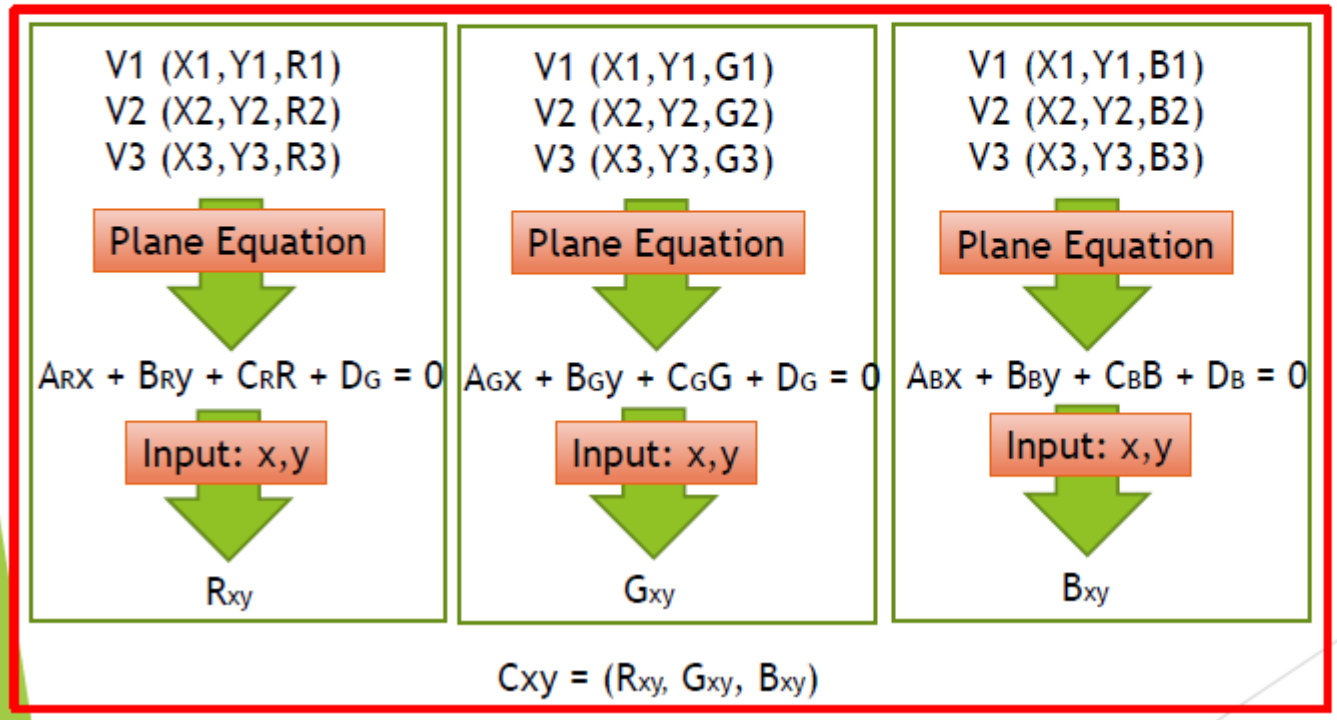
  Ax + By + Cz + D = 0 [1] where vector (A,B,C) is normal to the plane

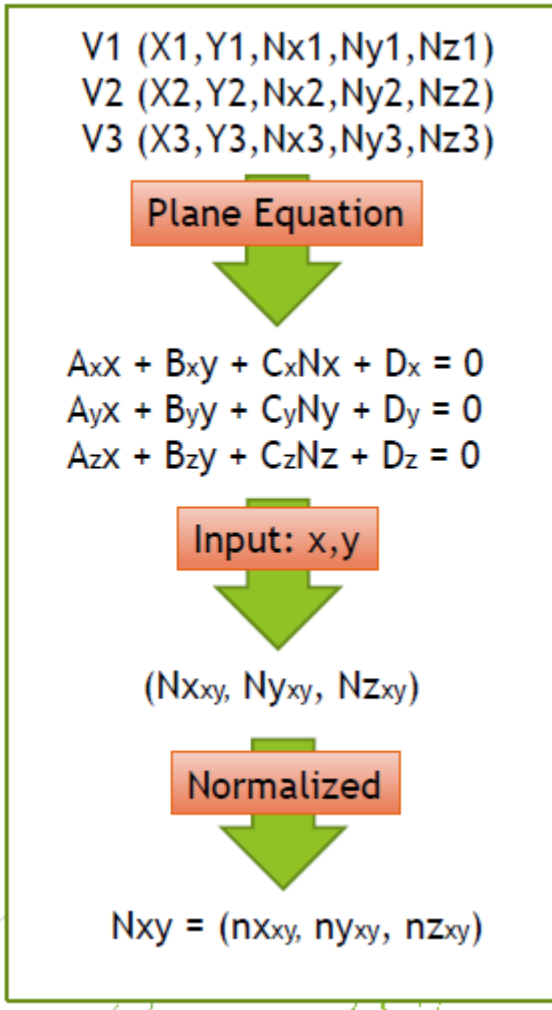
- Given (A,B,C,D) for a given tri, evaluate [1] at any pixel (x,y) and solve for z at that pixel



Interpolating with colors/normals

- We can substitute other parameters for Z, and compute (A,B,C,D) plane coefficients to interpolate these parameters to pixels





Q&A