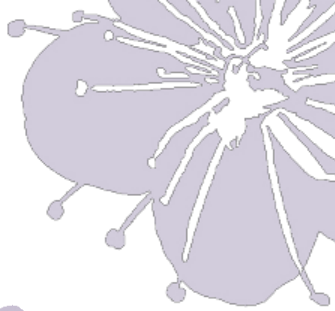
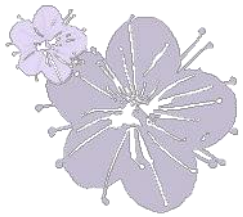
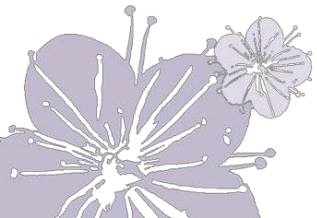



OpenGL Homework 3

CS 550000 Computer Graphics
April 27, 2016
Department of Computer Science
National Tsing Hua University

Outline

- Goal
- Grading
- Control
- Submission
- Reminders
- Appendix

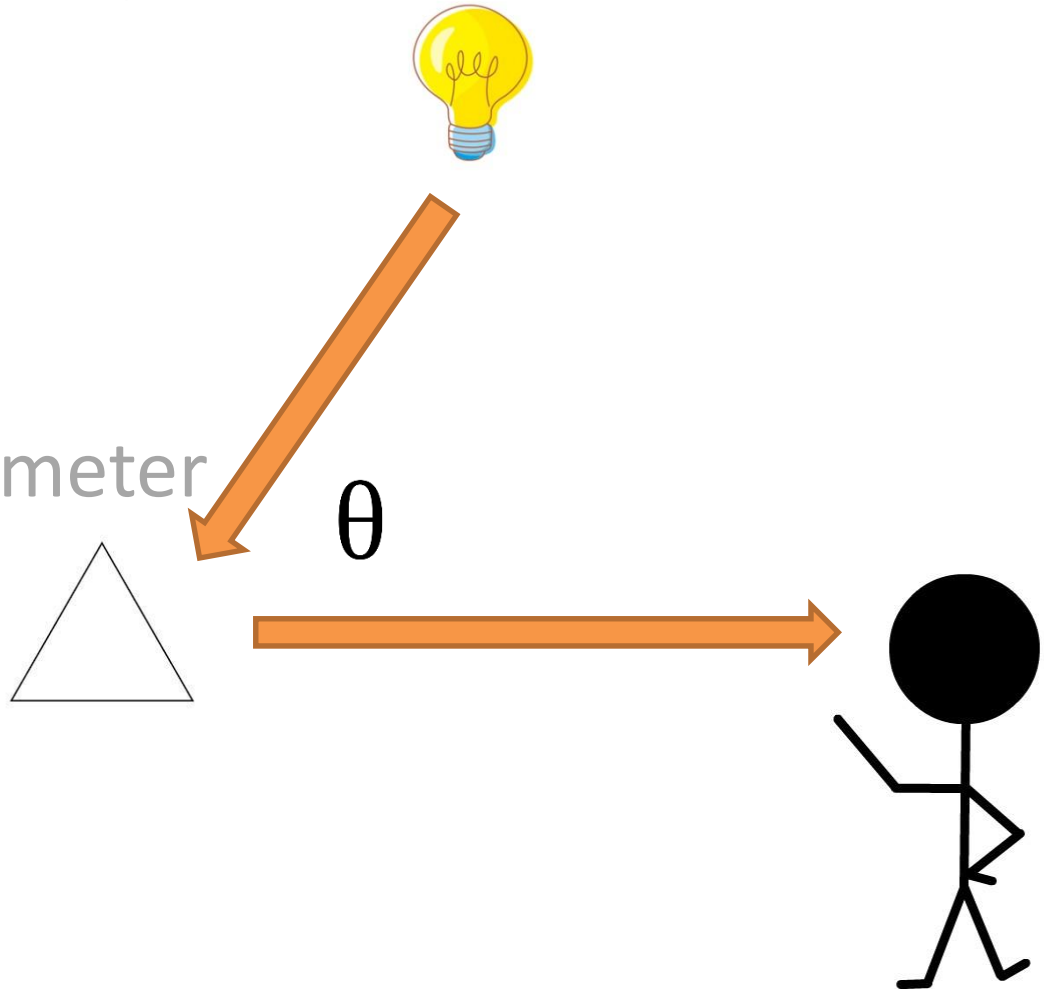
Outline

- Goal
- Grading
- Control
- Submission
- Reminders
- Appendix

Goal

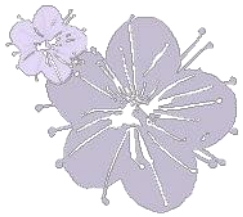
Source type and parameter

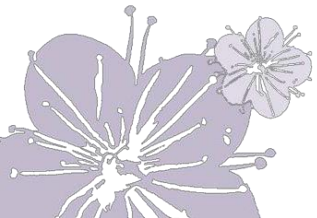
Material Parameter



Goal

- This homework will focus on how to write a lighting effect in vertex shader (Gouraud shading).





Goal

	Directional	Point	Spot
Ambient			
Diffuse			
Specular			
Cut-off angle			
Exp			

Goal

(Material Para)*(Source Para)*(Source Effect)*(Color)

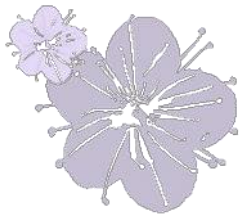
Para = Ambient, Diffuse, Specular… etc.
Read Material Para from file,
Set Source Para by yourself

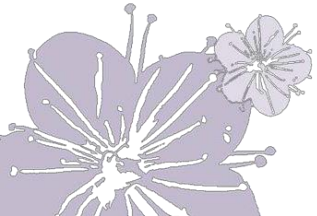
Source Effect depend on the equation and setting

Goal

Sum the lighting result of each light-source

You can think of “Ambient” as a parameter,
Or you can implement it as a kind of light source.





Notice

- Change color models to normal models.
- Traverse normal value instead of color value.
- The model is composed of different groups.
- A group is composed of a lot of triangles.
- Each group shares the same material.

Outline

- Goal
- Grading
- Control
- Submission
- Reminders
- Appendix

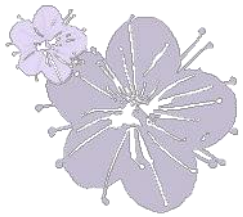
Grading – Basic(Gouraud shading) (100%)

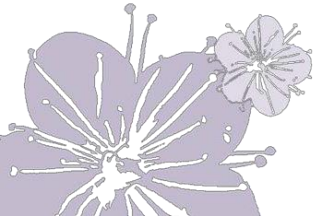
- Lighting (80%)
  - Directional Light (20 %)
    - Ambient (4 %)
    - Diffuse (8 %)
    - Specular (8 %)
  - Point Light (30%)
    - Ambient (10%)
    - Diffuse (10%)
    - Specular (10%)
  - Spot Light (30%)
    - Ambient (6%)
    - Diffuse (6%)
    - Specular (6%)
    - SpotExp (6%)
    - SpotConsCutoff (6%)

- Basic & Control (10%)
  - Normalization, Perspective mode
  - Press R/r to auto rotate the Model
  - Change each light source position
  - Change spot light parameter
  - Turn on/off the lights
  - Help menu
- Report (10%)
  - Express your work.
    - How to operate your program
    - Implementation and problems you met
    - Other efforts you have done
    - Screenshots

Grading – Bouns(Phong shading)(15%)

- Lighting (10%)
  - Implement phong shading
- Keyboard (5%)
  - Change the shading mode



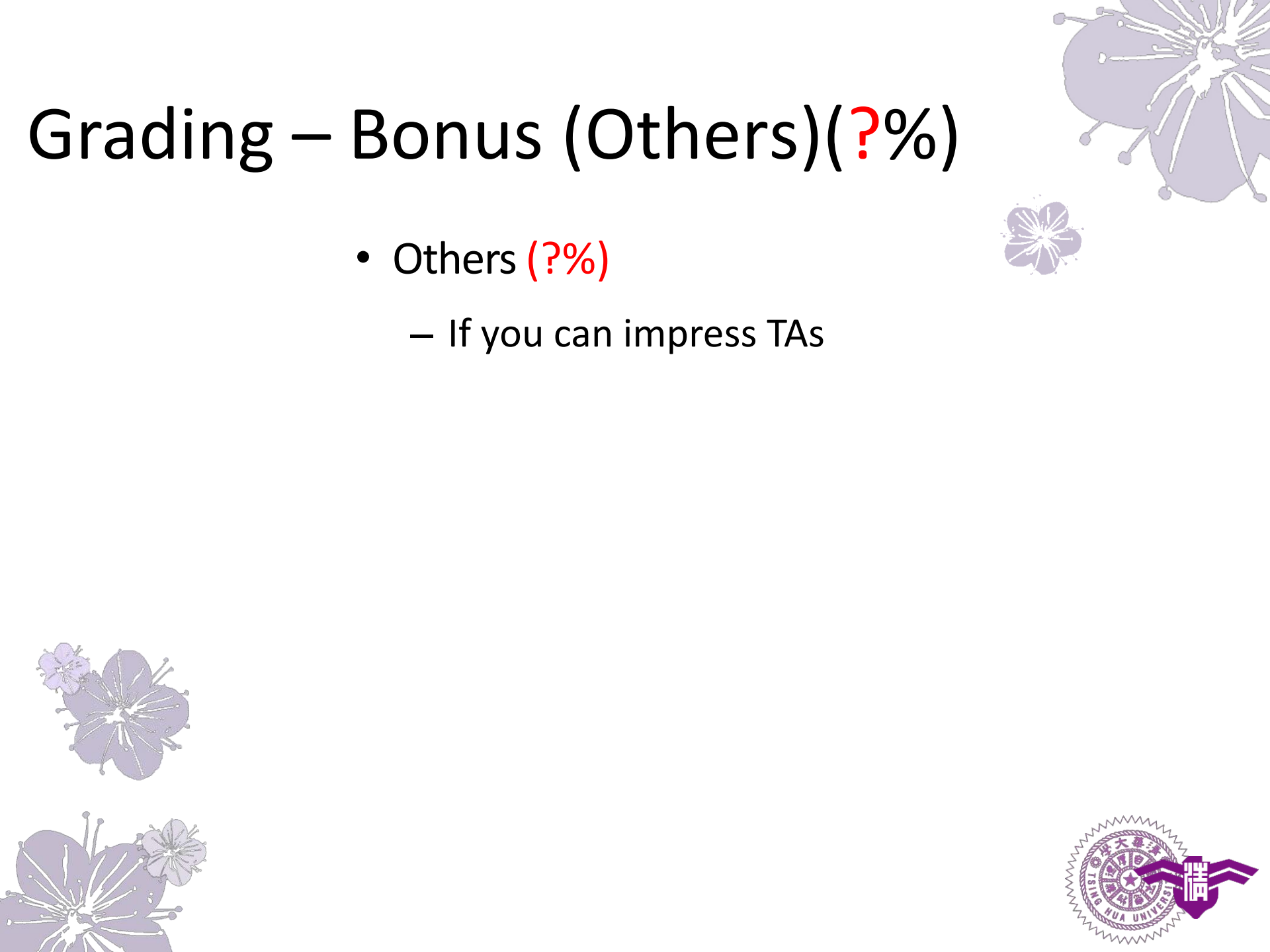


Grading – Bouns(change source)(5%)

- Correct equation (5%)
  - Change spot light to other kind of source

Grading – Bonus (Others)(?%)

- Others (?%)
  - If you can impress TAs



Outline

- Goal
- Grading
- Control
- Submission
- Reminders
- Appendix

Control (Toggle)

Directional (3 different) ***Point*** ***Spot*** ***Auto Rotation***

Ambient ***Diffuse*** ***Specular*** ***Per-pixel***

Control (Point light)

Initially locate at (0,-1,0)
Move the point light on y-plane

Control (Spot light)

Click to tune the EXP

Scroll to tune the cut-off angle

Hover to move the position on z-plane (z = 1)

Control (Spot light)

Spot direction are fixed to (0,0,-1)

glutPassiveMotionFunc(onMouseMotion);

Control (Others)

- z/x : change model
- o/p : perspective
- h : help menu
- c : change the spot light to another kind

Outline

- Goal
- Grading
- Control
- Submission
- Reminders
- Appendix

Submission

- Due date: You should know
- Submit your project to iLMS.
- Filename: HW3_XXXXXXXXX.zip
- Zip your file base on the requirement

*** Delete What you should delete***

Outline

- Goal
- Grading
- Control
- Submission
- Reminders
- Appendix

Reminders

- Late submission is accepted (-10 points/week), DO NOT give up!
- Ask and share information through iLMS

課程資訊 [報表]

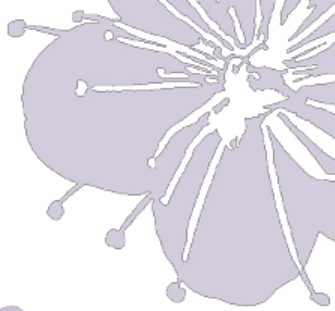
訪客: 3495
文章: 147
討論: 147
容量: 剩餘 **1.5 GB** (2.9 GB)
老師: 李潤容
助教: 羅逸翔, 阮維廷, 蔡繪琦
閱讀權限: 不開放旁聽 (僅成員可以閱讀)

課程功能 [管理]

課程活動(公告)
上課教材 (6)
課堂整理
課程說明
課程行事曆
討論區 (147)
小組專區

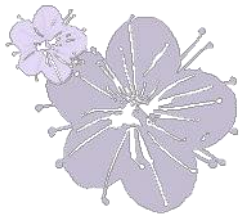
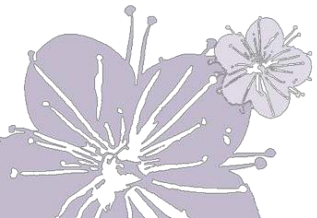
Outline

- Goal
- Grading
- Submission
- Reminders
- Control
- Appendix




Appendix

CS 550000 Computer Graphics
April 27, 2016
Department of Computer Science
National Tsing Hua University


How to traverse

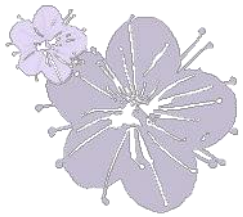
- texturedknot11KC.obj
- maxplanck20KN.obj
- lucy25KN.obj
- lion12KN.obj
- igea17KN.obj
- hippo23KN.obj
- happy10KN.obj
- elephant16KN.obj
- dragon10KN.obj
- Dino20KN.obj
- brain18KN.obj
- armadillo12KN.obj
- texturedknot11KC.obj.mtl
- maxplanck20KN.obj.mtl
- lucy25KN.obj.mtl
- lion12KN.obj.mtl
- igea17KN.obj.mtl
- hippo23KN.obj.mtl
- happy10KN.obj.mtl
- elephant16KN.obj.mtl
- dragon10KN.obj.mtl
- Dino20KN.obj.mtl
- brain18KN.obj.mtl
- armadillo12KN.obj.mtl

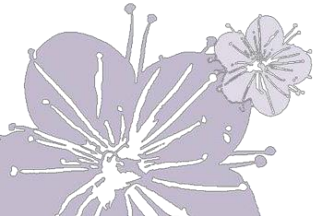
How to traverse

```
GLMmodel* OBJ;
GLMgroup* group = OBJ->groups;
while (group)
{
        //Get material data here
        for (i = 0; i<(int)group->numtriangles; i++)
        {
                //Get OBJ data here
        }
        group = group->next;
}
```

Get material data

- OBJ->materials[group->material].ambient
- OBJ->materials[group->material].diffuse
- OBJ->materials[group->material].specular





Get triangle data

- ***Same method as Assigment # 01***

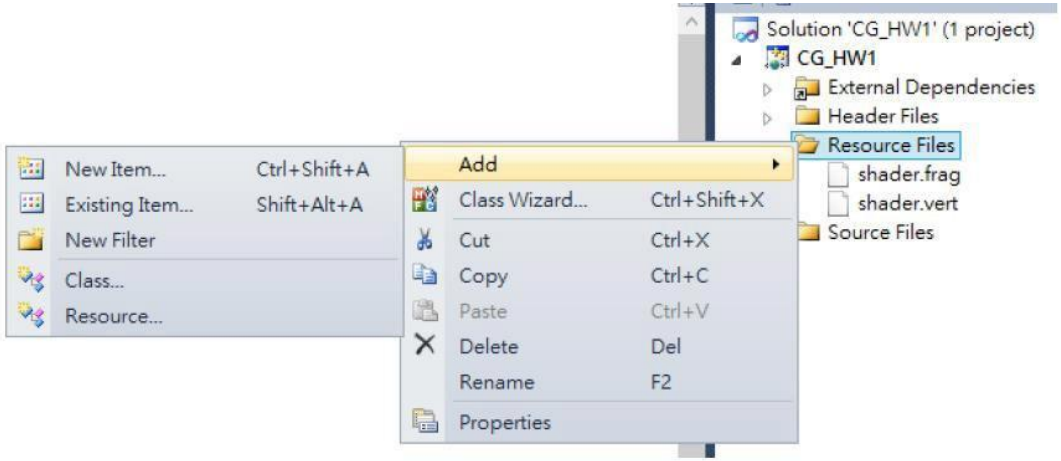
```
//Triangle index
int triangleID = group->triangles[i];
//the index of each vertices
int indv1 = OBJ->triangles[triangleID].vindices[0];
int indv2 = OBJ->triangles[triangleID].vindices[1];
int indv3 = OBJ->triangles[triangleID].vindices[2];
```

Use vertex normal for lighting

```
//the index of each normals
int indn1 = OBJ->triangles[triangleID].nindices[0];
int indn2 = OBJ->triangles[triangleID].nindices[1];
int indn3 = OBJ->triangles[triangleID].nindices[2];
// the vertex normal
OBJ->normals[indn1*3];
OBJ->normals[indn1*3+1];
OBJ->normals[indn1*3+2];
OBJ->normals[indn2*3];
OBJ->normals[indn2*3+1];
OBJ->normals[indn2*3+2];
OBJ->normals[indn3*3];
OBJ->normals[indn3*3+1];
OBJ->normals[indn3*3+2];
```

How to change the shader file

- add *sample.vert* and *sample.frag* to *Resource Files*



- change the strings in *textFileRead* to *sample.vert* and sample.*frag in setShaders()*

```
vs = textFileRead("sample.vert");
fs = textFileRead("sample.frag");
```

Shader code

- Notice that the shader codes provided by TAs are ambient only. You need to add diffuse , specular, and so on in your light equation.

- In fact, there are so many lighting shader code in google. You can reference them, too.

- Lighthouse3d
  - http://www.lighthouse3d.com/tutorials/glsl-tutorial/lighting/

Shader code

- You should use the method referred in the slide
- Don’t do tricks
  - E.g. the method of spot light
- So you are supposed to write the detail in report.

(p.s. TA has hidden the shader code)

Add GL API in main.cpp

- add the following **global variables**

```
GLint iLocNormal;
GLint iLocMDiffuse;
GLfloat *normals;
```

It will store the *location* of shader variables

We don't need **color* this time because we have material attributes. But we need **normal* this time for lighting. BTW, don't forget to *malloc() *normal*.

- add the following sentences in *setShaders()* after *glLinkProgram(p)*

Notice that this argument is "variable name" in your shader code.

```
glEnableVertexAttribArray(iLocNormal);

iLocNormal = glGetAttribLocation(p, "av3normal");
iLocMDiffuse = glGetUniformLocation(p, "uv4matDiffuse");
```

- Add *glUniform4v(…)* in the traversing part to assign value to shader.

```
GLMgroup* group = OBJ->groups;
while (group)
{
  glUniform4fv(iLocMDiffuse, 1, OBJ->materials[group->material].diffuse);
```

- Change *glVertexAttribPointer* and *glDrawArrays* to the correct position. (Hint: there are the same material attributes in a group. Then, think by yourself, please)
- add the following sentence after *glVertexAttribPointer(…, vertices)*

```
glVertexAttribPointer(iLocNormal, 3, GL_FLOAT, GL_FALSE, 0, normals);
```

Add GL API in main.cpp

- Similar as previous page, you need to add other some global variable and *glGetUniformLocation()* to connect with shader.
- Following are some variables that you might use **or not**.

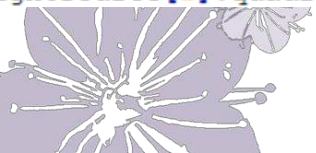
```
iLocMDiffuse = glGetUniformLocation(p, "Material.diffuse");
iLocMAmbient = glGetUniformLocation(p, "Material.ambient");
iLocMSpecular = glGetUniformLocation(p, "Material.specular");
iLocMShininess = glGetUniformLocation(p, "Material.shininess");

iLocLDAmbient = glGetUniformLocation(p, "LightSource[0].ambient");
iLocLDPosition = glGetUniformLocation(p, "LightSource[0].position");
```

```
attribute vec4 av4position;
attribute vec3 av3normal;

varying vec4 vv4color;

uniform mat4 mvp;

struct LightSourceParameters {
    vec4 ambient;
    vec4 diffuse;
    vec4 specular;
    vec4 position;
    vec4 halfVector;
    vec3 spotDirection;
    float spotExponent;
    float spotCutoff; // (range: [0.0,90.0], 180.0)
    float spotCosCutoff; // (range: [1.0,0.0],-1.0)
    float constantAttenuation;
    float linearAttenuation;
    float quadraticAttenuation;
};

struct MaterialParameters {
    vec4 ambient;
    vec4 diffuse;
    vec4 specular;
    float shininess;
};

uniform MaterialParameters Material;
uniform LightSourceParameters LightSource[3];

void main() {

    vec4 vv4ambient_D = Material.ambient * LightSource[0].ambient;
    vv4color = vv4ambient_D;
    gl_Position = mvp * av4position;
}
```

Add GL API in main.cpp

- 3 light source, each has their own parameter.
- Material parameter, Control variable
- MVP matrices
- A bunch of variable to pass…lol
- Well manage your variable before you go.
- Trace the framework to get more ideas

```
/Ambient
ightsource[0].position[0] = 0;
ightsource[0].position[1] = 0;
ightsource[0].position[2] = -1;
ightsource[0].position[3] = 1;
ightsource[0].ambient[0] = 0.75;
ightsource[0].ambient[1] = 0.75;
ightsource[0].ambient[2] = 0.75;
ightsource[0].ambient[3] = 1;

/directional light
ightsource[1].position[0] = 0;
ightsource[1].position[1] = 1;
ightsource[1].position[2] = 0;
ightsource[1].position[3] = 1;
ightsource[1].ambient[0] = 0;
ightsource[1].ambient[1] = 0;
ightsource[1].ambient[2] = 0;
ightsource[1].ambient[3] = 1;
ightsource[1].diffuse[0] = 1;
ightsource[1].diffuse[1] = 1;
ightsource[1].diffuse[2] = 1;
ightsource[1].diffuse[3] = 1;
ightsource[1].specular[0] = 1;
ightsource[1].specular[1] = 1;
ightsource[1].specular[2] = 1;
ightsource[1].specular[3] = 1;
ightsource[1].constantAttenuation = 1;
ightsource[1].linearAttenuation = 4.5 / PLRange;
ightsource[1].quadraticAttenuation = 75 / (PLRange*
```



```
//point light
lightsource[2].position[0] = 1;
lightsource[2].position[1] = 0;
lightsource[2].position[2] = -1;
lightsource[2].position[3] = 1;
lightsource[2].ambient[0] = 0
lightsource[2].ambient[1] = 0;
lightsource[2].ambient[2] = 0;
lightsource[2].ambient[3] = 1;
lightsource[2].diffuse[0] = 1;
lightsource[2].diffuse[1] = 1;
lightsource[2].diffuse[2] = 1;
lightsource[2].diffuse[3] = 1;
lightsource[2].specular[0] = 1;
lightsource[2].specular[1] = 1;
lightsource[2].specular[2] = 1;
lightsource[2].specular[3] = 1;
lightsource[2].constantAttenuation = 1;
lightsource[2].linearAttenuation = 4.5 / PLRange;
lightsource[2].quadraticAttenuation = 75 / (PLRange*PLRange);

//spot light
lightsource[3].position[0] = 0;
lightsource[3].position[1] = 0;
lightsource[3].position[2] = 2;
lightsource[3].position[3] = 1;
lightsource[3].ambient[0] = 0;
lightsource[3].ambient[1] = 0;
lightsource[3].ambient[2] = 0;
lightsource[3].ambient[3] = 1;
lightsource[3].diffuse[0] = 1;
lightsource[3].diffuse[1] = 1;
lightsource[3].diffuse[2] = 1;
lightsource[3].diffuse[3] = 1;
lightsource[3].specular[0] = 1;
lightsource[3].specular[1] = 1;
lightsource[3].specular[2] = 1;
lightsource[3].specular[3] = 1;
lightsource[3].spotDirection[0] = 0;
lightsource[3].spotDirection[1] = 0;
lightsource[3].spotDirection[2] = -2;
lightsource[3].spotDirection[3] = 0;
lightsource[3].spotExponent = 0.1;
lightsource[3].spotCutoff = 45;
lightsource[3].spotCosCutoff = 0.96592582628; //1/12 pi
```

Shader compile status

- Because shader is compiled in runtime, you can't see error messages in compile time.
- So, you may add some code after *glCompileShader(v)* and *glCompileShader(f)* to ensure your shader is compliable and print the log information after compile.

```
// compile vertex shader
glCompileShader(v);
GLint vShaderCompiled;
showShaderCompileStatus(v, &vShaderCompiled);
if(!vShaderCompiled) system("pause"), exit(123);

// compile fragment shader
glCompileShader(f);
GLint fShaderCompiled;
showShaderCompileStatus(f, &fShaderCompiled);
if(!fShaderCompiled) system("pause"), exit(456);
```

Quotes

And God said, Let there be light: and there was light.

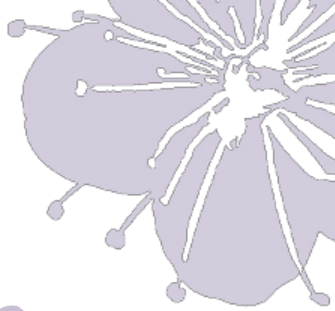
- Generis 1:3

神說：「要有光。」就有了光。

- 創世紀 1:3

Shader says, Let there be light: and where is light?


Q
&
A

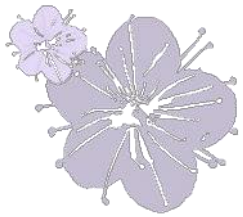
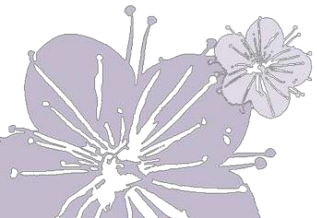



OpenGL Homework 3 Hint

CS 550000 Computer Graphics

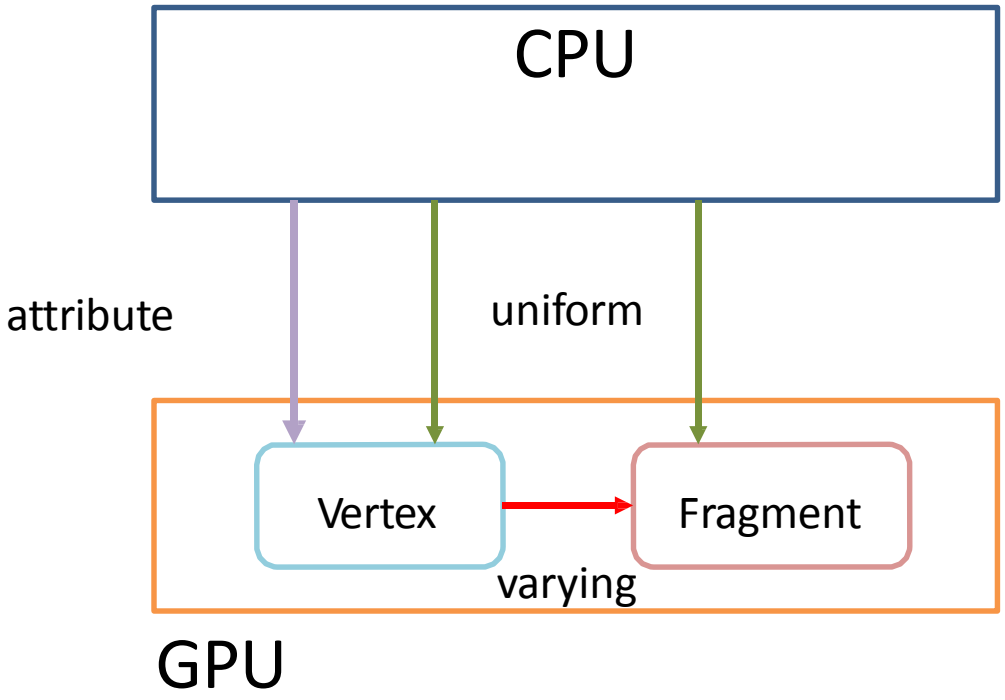
April, 2017

Department of Computer Science
National Tsing Hua University

Shader

- Vertex shader
  - Deal with vertex
- Fragment shader
  - Deal with pixel



What should we pass to shader?

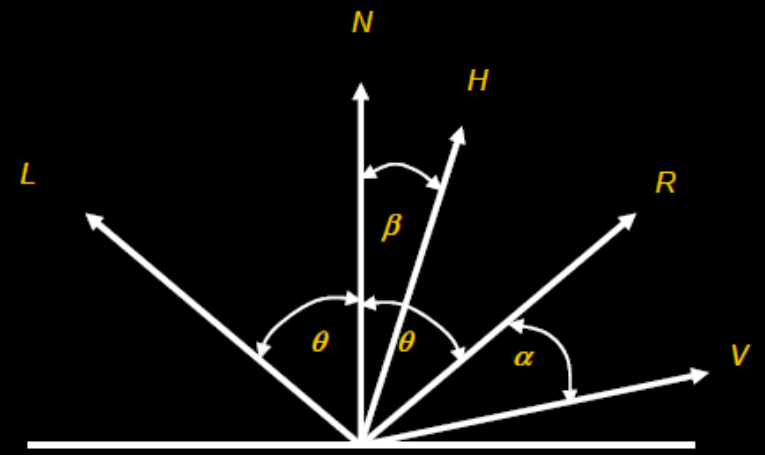
- Transformation
  - Vertices position (av4position)
  - mvp matrix
- Lighting
  - Normal vector(av3normal)
  - Model Transform matrix
  - Rotate matrix with transpose after inverse
  - Lighting parameters
  - Material parameters
  - Eye position (in world space)

$$I = I_a k_a + \sum_{p=1}^{m} f_p I_p (k_d (N \cdot L_p) + k_s (N \cdot H_p)^{n'})$$

$$H = \frac{L+V}{|L+V|}$$

$$\theta + \beta = \theta - \beta + \alpha$$

$$\beta = \frac{1}{2}\alpha$$



- L : vector of vertex position to light source
- V : vector of vertex position to eye position
- vertex position : M * N * Vertices position
  (M = Model Transform matrix, N = Normalization matrix)
- N : R * Normal vector
  (R = Rotate matrix with transpose after inverse)


Q
&
A