
財團法人國家實驗研究院
國家高速網路與計算中心
National Center for High-Performance Computing
OpenGL
OpenGL 程式設計
林敬堯@NCHC
chingyao@nchc.org.tw
DATE:8/29/07

課程大綱

- Introduction/概說
- First example/第一個範例程式
- Basic primitives/基本圖畫單元
- Transformation/座標轉換
- More about GLUT
- Light/光
- Pick & Select/如何選擇物件
- Performance issues/效能

Introduction/概說 (1)

- **OpenGL** is a standard specification defining a cross-language cross-platform API for writing applications that produce 2D and 3D computer graphics. (*from wiki*)
  - Window system independent
  - Hardware independent
- OpenGL Library: A software interface to graphics hardware

Introduction/概說 (2)

What does OpenGL do?

- Draw points, lines, and polygons.
- Model/View Transformation
- Depth test (Hidden Surface Removal)
- Light/Shading (Gouraud)
- Texture mapping
- Pixels/fragment operation

Introduction/概說 (3)

- OpenGL is a state machine
- Call OpenGL functions to set OpenGL state.

```
glColor3f(0.f, 0.f, 1.f);
…
glEnable(GL_DEPTH_TEST);
…
glDisable(GL_DEPTH_TEST);
```

Introduction/概說 (4)

- OpenGL Extensions
  - Introduce new functions and new constants, and relax or remove restrictions on existing OpenGL functions
  - Each vendor has an alphabetic abbreviation
  - Standard Extension (OpenGL Extension Registry)
    - ARB – officially approved by the OpenGL Architecture Review Board (OpenGL ARB)
    - EXT – agreed upon by multiple OpenGL vendors.

Introduction/概說 (5)

- Extensions
  - NVIDIA
    - Use the abbreviation “NV”
      - Example: glCombinerParameterfvNV()
  - How to use extensions?
    - GLEW (OpenGL Extension Wrangler Library)
    - GLEE (OpenGL Easy Extension library)

Introduction/概說 (6)

- What **can't** you do with OpenGL?
  - Create Window,
  - Input (ex: 滑鼠, 鍵盤)
  - User Interface

Introduction/概說 (7)

- OpenGL Utility Toolkit (GLUT) (*http://www.xmission.com/~nate/glut.html*)
  - A platform-independent library for managing windows, input.
  - Useful for simple applications
  - How to use GLUT?
    - Copy glut32.dll to window \system
    - Copy glut32.lib & glut.h to right place
    - Include “GL/glut.h” instead of “GL/gl.h”
- Other UI: FLTK, wxWidget, QT

First Example/第一個範例程式

Example: Triangle.c

```
void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f(1.f, 0.f, 0.f);

    // draw a triangle
    glBegin(GL_TRIANGLES);
    glVertex3f (0.f, 1.f, 0.0);
    glVertex3f (0.866f, -0.5f, 0.0);
    glVertex3f (-.866f, -0.5f, 0.0);
    glEnd();

    glFlush ();
}
```

First Example -Using VC 2005 express

- Download and install VC 2005 express (search google)
- Download and install **Microsoft Platform SDK.**
  - http://msdn2.microsoft.com/en-us/vstudio/aa700755.aspx
  - Need to follow the instructions on that pages!
- Download GLUT, copy

```
- glut32.dll to %WinDir%\System,
- glut32.lib to $(MSDevDir)\..\..\VC98\lib,
- glut.h to $(MSDevDir)\..\..\VC98\include\GL.
```

First Example - On Linux (c/c++)

- Link:
  `-lglut -GLU -GL -lXt -lX11 -lXext -lm`

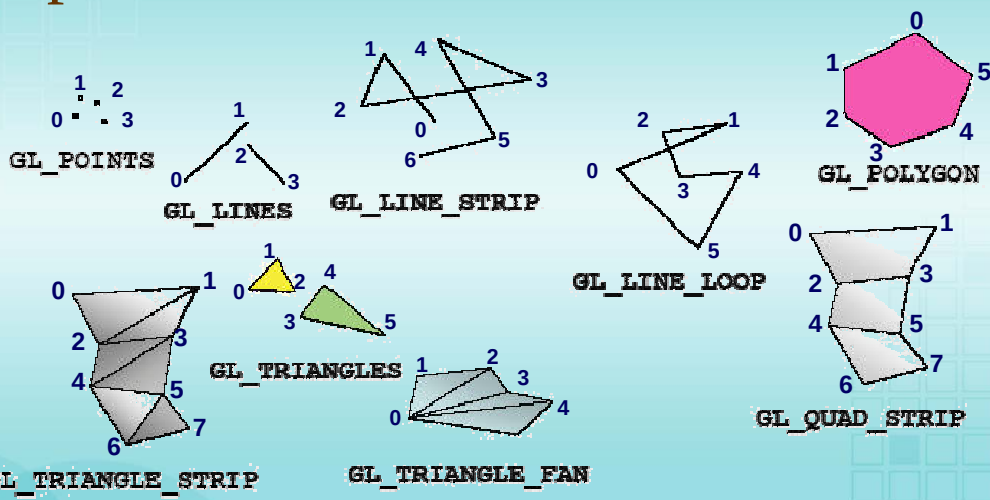
- FLTK
  - http://www.fltk.org/
  - Download and install (linux or windows)
  - It will have example codes of how to set up an OpenGL window

Remote OpenGL

- Linux to Windows
  - VirtualGL
  - http://www.virtualgl.org/
  - install turbovnc server on Linux
  - compile/install virtualGL on Linux
  - on clients, run turboVNC viewer
  - on the server, use ‘vglrun’ to run OpenGL application

Basic Primitives (1)

- OpenGL Primitives



Basic Primitives (2)

- Drawing Primitives
  - `glBegin(GLenum mode)`
  - `glEnd()`

```
glColor3f(1.f, 0.f, 0.f);
glBegin(GL_TRIANGLES);
glVertex3f(0.f, 0.f, 0.f);
glVertex3f(1.f, 0.f, 0.f);
glVertex3f(0.f, 1.f, 0.f);
glEnd();
```

Basic Primitives (3)

- Modify the example

```
void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f(1.f, 0.f, 0.f);

    // draw a triangle
    glBegin(GL_TRIANGLES);
    glVertex3f (0.f, 1.f, 0.0);
    glVertex3f (0.866f, -0.5f, 0.0);
    glVertex3f (-.866f, -0.5f, 0.0);
    glEnd();

    glFlush ();
}
```

```
glColor3d(…)
glColor3i(…)
glColor3ub(…)
…
glColor4d(…)
glColor4f(…)
glColor4i(…)
glColor4ub(…)
```

Basic Primitives (3)

- Modify the example

```
void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);
    glColor3f(1.f, 0.f, 0.f);

    // draw a triangle
    glBegin(GL_TRIANGLES);
    glVertex3f (0.f, 1.f, 0.0);
    glVertex3f (0.866f, -0.5f, 0.0);
    glVertex3f (-.866f, -0.5f, 0.0);
    glEnd();

    glFlush ();
}
```

glVertex + (2/3/4)+ (d/f/i/s) + <v>

Ex: void glVertex3iv(const GLint *v);

Basic Primitives (4)

- Transparent effects
- `glColor4f(r, g, b, alpha);`

```
glEnable(GL_BLEND)
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

Basic Primitives (5)

- Display Lists
  - A display list stores a sequence of OpenGL commands for later execution.
  - It is a simple way of enhancing your OpenGL application making it run faster.

Basic Primitives (6)

- Create display list

```
GLuint dl;
dl = glGenLists(1);
```

- Store the commands

```
GL_COMPILE &
GL_COMPILE_AND_EXECUTE
```

```
glNewList(dl, GL_COMPILE);
glBegin(GL_TRIANGLES);
glVertex3f (0.f, 1.f, 0.0);
glVertex3f (0.866f, -0.5f, 0.0);
glVertex3f (-.866f, -0.5f, 0.0);
glEnd();
glEndList();
```

Basic Primitives (7)

- Execute the command in a display list

```
glCallList(dl);
```

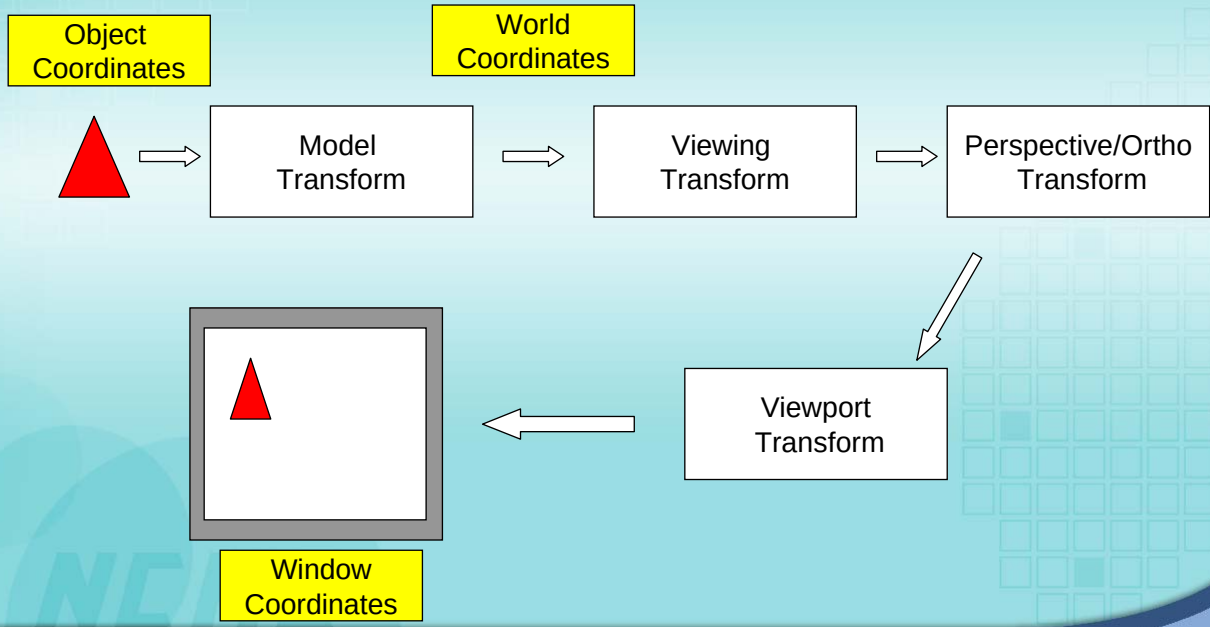
Demo: tool_amView

Transformation

- Scene setup
  - Model transformation
    - Arrange objects within viewing volume
  - View transformation
    - Camera position
    - Viewing direction
    - View-up vector
  - Projection
    - Choose shape of viewing volume
  - Viewport transformation
    - Project to display coordinates

Transformation

- A series of operations that



Model Transformation (1)

- Model Transform
  - `glMatrixMode(GL_MODELVIEW);`
  - `glRotate*(`***`theta`***`,x,y,z);` `*: d(ouble), f(loat)`
    - Rotate ***theta*** degrees degree around vector (x, y, z);
  - `glTranslate* (x,y,z);`
    - Translate (x, y, z);
  - `glScale*(x,y,z);`
    - Scale (X, Y, Z);

Model Transformation (2)

Example 1: Transform.c

- `glTranslatef(x, y, z);`
- `glRotatef(angle, x, y, z);`

$$v = T_1 \times T_2 \times v_1$$

```
glRotatef(…);     // T1
glTranslatef(…);  // T2
drawSquare();     // v1
```

Model Transformation (3)

- The matrix in OpenGL is – **"Column-Major"**

$$\begin{bmatrix} a_0 & a_4 & a_8 & a_{12} \\ a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \end{bmatrix}$$

- Ex. An translation matrix

$$\begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x-2 \\ y+3 \\ z+5 \\ 1 \end{pmatrix}$$

Model Transformation (4)

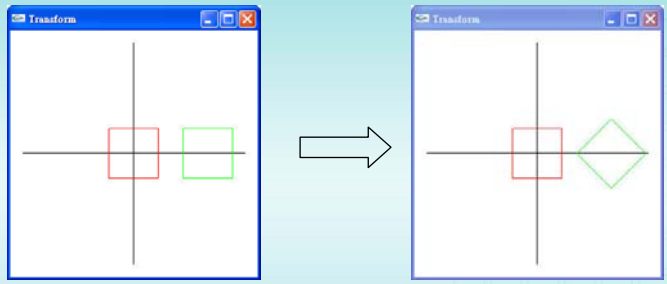
- Matrix manipulation
  - `glLoadIdentity();`
  - `glLoadMatrix *(const TYPE *m);`
    - Replace the top of the active matrix with `m`
  - `glMultMatrix* (const TYPE *m);`
    - Multiple the current matrix with `m`

```
*: d(ouble), f(loat)
```

Model Transformation (5)

- Code test: translate and rotate

Example 2: Transform.c



- `glLoadIdentity();`

Test: change the order of
`glRotatef(…)` and `glTranslatef(…)`

Model Transformation (6)

- `glPushMatrix():`
  - Save the current transformation matrix into a stack.
- `glPopMatrix():`
  - Restore a transformation matrix from the stack.

Example 3: Transform_2.c

Model Transformation (7)

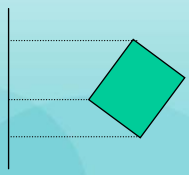
- Viewing transform:
  rotate the object or move the camera?

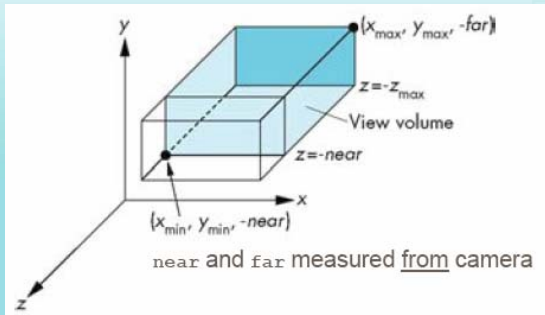
```
void gluLookAt(
   GLdouble eyeX,
   GLdouble eyeY,
   GLdouble eyeZ,
   GLdouble centerX,
   GLdouble centerY,
   GLdouble centerZ,

   GLdouble upX,
   GLdouble upY,
   GLdouble upZ );
```

View Transformation (1)

- `glMatrixMode(GL_PROJECTION);`
- `glOrtho(…)`: 2D map, chart

```
void glOrtho(
   GLdouble left, GLdouble right,
   GLdouble bottom, GLdouble top,
   GLdouble zNear, GLdouble zFar )
```

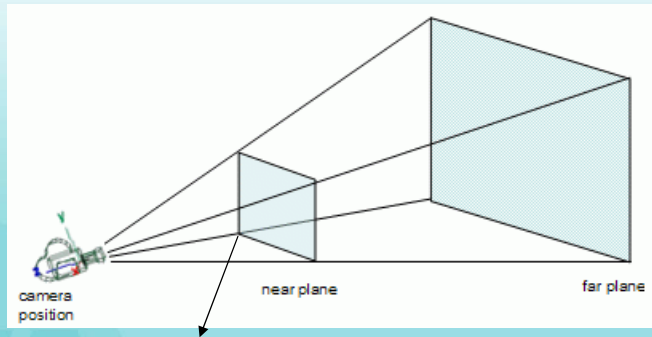




View Transformation (2)

- glFrumtum(…)

```
void glFrustum(
   GLdouble left, GLdouble right,
   GLdouble bottom, GLdouble top,
   GLdouble zNear, GLdouble zFar )
```



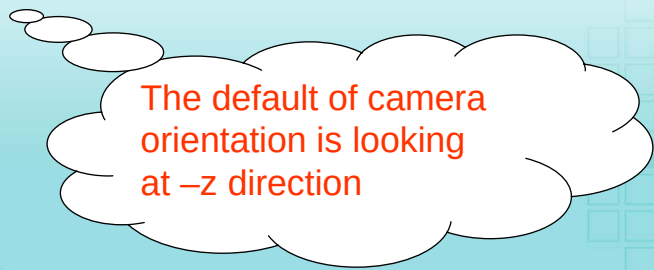
(R, T, F)

(L, B, N)

View Transformation (3)

- `gluPerspective(…)`

```
void gluPerspective(
   GLdouble fovy,   // vertical field of view
   GLdouble aspect, // match window aspect ratio
   GLdouble zNear, GLdouble zFar )
```



More about GLUT (1)

- Initialize window

```
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize (300, 300);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("Triangle");
    …
    glutMainLoop();
}
```

More about GLUT (2)

- Initialize window

```
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize (300, 300);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("Triangle");
    …
    glutMainLoop();
}
```

Initialize GLUT library

More about GLUT (3)

- Initialize window

```
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize (300, 300);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("Triangle");
    …
    glutMainLoop();
}
```

Set the display

More about GLUT (4)

- Initialize window

```
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize (300, 300);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("Triangle");
    …
    glutMainLoop();
}
```

Set the size and location

More about GLUT (5)

- Initialize window

```
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize (300, 300);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("Triangle");
    …
    glutMainLoop();
}
```

Window name

More about GLUT (6)

- Initialize window

```
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize (300, 300);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("Triangle");
    …
    glutMainLoop();
}
```

Enter the GLUT even processing loop

More about GLUT (7)

- Display Mode

```
glutInitDisplayMode(…):
- GLUT_SINGLE/GLUT_DOUBLE
- GLUT_RGB/GLUT_RGBA /(GLUT_INDEX,…)
- GLUT_DEPTH: for hidden surface removal
- GLUT_ALPHA: with alpha components
```

More about GLUT (8)

- Events handler
- What is a callback function?
  - Window-system application are even-driven.
    - Programs create a window and enter an event loop.
    - As each event arrives, the program examines the event and processes it.
  - Even can be pressing a key, moving mouse, etc.
  - When an event happens, a function corresponding to that event is called.
    - The function is called “callback”.
    - We assign a callback function to a specific event.

More about GLUT (9)

- GLUT callback

```
int main(int argc, char **argv)
{
    …

    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutSpecialFunc(…);
    glutMouseFunc(…);
    glutMotionFunc(…);
    glutIdleFunc(…);
    glutReshapeFunc(…);

    glutMainLoop();
}
```

We need to declare callback functions as "**static function**" if we want to put the callback functions in a class

```
void display()
{
    …
    glutSwapBuffers();
}
```

More about GLUT (10)

- GLUT callback – idle function

```
int main(int argc, char **argv)
{
    …

    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutMouseFunc(…);
    glutMotionFunc(…);
    glutIdleFunc(…);
    glutReshapeFunc(…);
    glutSpecialFunc(…);

    glutMainLoop();
}
```

1. Can achieve animation
2 GLUT_DOUBLE

```
void idle()
{
    …
    glutPostRedisplay();
}
```

More about GLUT (10)

- Keyboard

```
int main(int argc, char **argv)
{
    …
    glutKeyboardFunc(keyboard);
    glutSpecialFunc(…);
    glutIdleFunc(…);
    ..
    glutMainLoop();
}
```

```
void keyboard(unsigned char key, int x, int y)
{
    switch (key)
    {
        …
    }
}
```

```
case ‘q’:
    break;
case ‘Q’:
    break;
case 27:  //ESC
    break;
case 32:  //Enter
    break;
```

More about GLUT (11)

- Special Keyboard (F1, F2,…, UP...)

```
int main(int argc, char **argv)
{
    …
    glutKeyboardFunc(keyboard);
    glutSpecialFunc(specialKey);
    glutIdleFunc(…);
    ..
    glutMainLoop();
}
```

```
void specialKey(int key, int x, int y)
{
    switch (key)
    {
        …
    }
}
```

More about GLUT (12)

- Special Key

```
GLUT_KEY_F1,
GLUT_KEY_F2,
…
GLUT_KEY_F12,
GLUT_KEY_UP,
GLUT_KEY_DOWN,
GLUT_KEY_LEFT,
GLUT_KEY_RIGHT,
GLUT_KEY_HOME,
GLUT_KEY_END,
GLUT_KEY_INSERT,
GLUT_KEY_PAGE_UP,
GLUT_KEY_PAGE_DOWN
```

To get the status of ALT, CTRL, SHIFT key, use :

```
glutGetModifiers();
```

Example 3: Transform_2.c

More about GLUT (13)

- Mouse events

```
int main(int argc, char **argv)
{
    …
    glutMouseFunc(mouse);
    glutMotionFunc(motion);
    ..
    glutMainLoop();
}
```

button:
GLUT_LEFT_BUTTON,
GLUT_MIDDLE_BUTTON,
GLUT_RIGHT_BUTTON

state:
GLUT_UP,
GLUT_DOWN

```
void mouse(int button, int state,
int x, int y)
{
}
```

More about GLUT (14)

- Mouse moving

```
int main(int argc, char **argv)
{
    …
    glutMouseFunc(mouse);
    glutMotionFunc(motion);
    ..
    glutMainLoop();
}
```

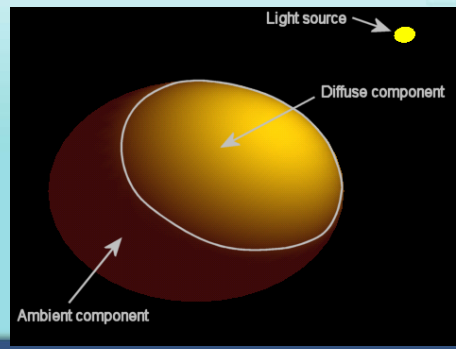
```
void motion(int x, int y)
{
}
```

Example 4: Events_light

- Perspective & Ortho projection (p)
- Trackball class
  - Drag left button: rotate
  - Drag middle button: translation

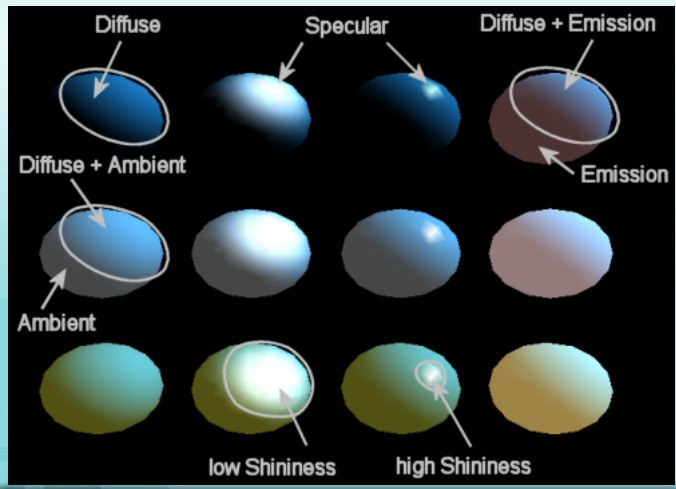
Light (1)

- Make things real
- Different light:
  - Ambient: indirect lighting
  - Diffuse: based on its orientation to a light source.



Light (2)

- Specular: reflection of light source on a shinny surface. It came from a particular direction.



Light (3)

- Enable the light
  `glEnable(GL_LIGHTING);`
- Disable the light
  `glDisable(GL_LIGHTING);`

Light (4)

- To see the light effects, we need to provide surface normal
  `glNormal3*()`

```
*:b, d, f, i, s
```

Light (5)

- Light parameters
- OpenGL support 8 or more light sources.
- Enable and disable individual light sources:

```
glEnable(GL_LIGHTi);
glDisable(GL_LIGHTi);
```

Light (6)

- Simple light effects

```
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0); // default light
```

- Set up different light

```
GLfloat paleYellow[4] = {1.f, 1.f, .75f, 1.f};
GLfloat white[4] = {1.f, 1.f, 1.f, 1.f};
glLightfv(GL_LIGHT1, GL_DIFFUSE, paleYellow);
glLightfv(GL_LIGHT1, GL_SPECULAR, white);
```

Light (5)

- When the light is enabled, it **doesn't** use current color (by `glColor*(…)`). Instead, it use **material colors** set by calls to `glMaterial*(…)`

```
*: i(nteger), f(loat)
```

Light (6)

- Setup the material

```
GLfloat ambient[] = {0.3, 0.3, 0.3, 1.0};
GLfloat diffuse[] = {1.0, 1.0, 1.0, 1.0};
GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};
GLfloat position[] = {1.0, 2.0, 5.0, 0.0};

glMaterialfv(GL_FRONT, GL_AMBIENT, ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, specular);
glMaterialf(GL_FRONT, GL_SHININESS, shininess);
```

Example 4: Events_light

- Modified the surface normal in the surface code
- Different materials (1, 2, 3, 4)
- Blend function, ‘b’

Light - Different materials

Material	GL_AMBIENT	GL_DIFFUSE	GL_SPECULAR	GL_SHININESS
Brass	0.329412 0.223529 0.027451 1.0	0.780392 0.568627 0.113725 1.0	0.992157 0.941176 0.807843 1.0	27.8974
Bronze	0.2125 0.1275 0.054 1.0	0.714 0.4284 0.18144 1.0	0.393548 0.271906 0.166721 1.0	25.6
Polished Bronze	0.25 0.148 0.06475 1.0	0.4 0.2368 0.1036 1.0	0.774597 0.458561 0.200621 1.0	76.8
Chrome	0.25 0.25 0.25 1.0	0.4 0.4 0.4 1.0	0.774597 0.774597 0.774597 1.0	76.8
Copper	0.19125 0.0735 0.0225 1.0	0.7038 0.27048 0.0828 1.0	0.256777 0.137622 0.086014 1.0	12.8
Polished Copper	0.2295 0.08825 0.0275 1.0	0.5508 0.2118 0.066 1.0	0.580594 0.223257 0.0695701 1.0	51.2
Gold	0.24725 0.1995 0.0745 1.0	0.75164 0.60648 0.22648 1.0	0.628281 0.555802 0.366065 1.0	51.2
Polished Gold	0.24725 0.2245 0.0645 1.0	0.34615 0.3143 0.0903 1.0	0.797357 0.723991 0.208006 1.0	83.2
Pewter	0.105882 0.058824 0.113725 1.0	0.427451 0.470588 0.541176 1.0	0.333333 0.333333 0.521569 1.0	9.84615

Material	GL_AMBIENT	GL_DIFFUSE	GL_SPECULAR	GL_SHININESS
Silver	0.19225 0.19225 0.19225 1.0	0.50754 0.50754 0.50754 1.0	0.508273 0.508273 0.508273 1.0	51.2
Polished Silver	0.23125 0.23125 0.23125 1.0	0.2775 0.2775 0.2775 1.0	0.773911 0.773911 0.773911 1.0	89.6
Emerald	0.0215 0.1745 0.0215 0.55	0.07568 0.61424 0.07568 0.55	0.633 0.727811 0.633 0.55	76.8
Jade	0.135 0.2225 0.1575 0.95	0.54 0.89 0.63 0.95	0.316228 0.316228 0.316228 0.95	12.8
Obsidian	0.05375 0.05 0.06625 0.82	0.18275 0.17 0.22525 0.82	0.332741 0.328634 0.346435 0.82	38.4
Pearl	0.25 0.20725 0.20725 0.922	1.0 0.829 0.829 0.922	0.296648 0.296648 0.296648 0.922	11.264
Ruby	0.1745 0.01175 0.01175 0.55	0.61424 0.04136 0.04136 0.55	0.727811 0.626959 0.626959 0.55	76.8
Turquoise	0.1 0.18725 0.1745 0.8	0.396 0.74151 0.69102 0.8	0.297254 0.30829 0.306678 0.8	12.8
Black Plastic	0.0 0.0 0.0 1.0	0.01 0.01 0.01 1.0	0.50 0.50 0.50 1.0	32
Black Rubber	0.02 0.02 0.02 1.0	0.01 0.01 0.01 1.0	0.4 0.4 0.4 1.0	10

Pick & Select

- gluProject & gluUnProject

```
GLint gluProject(
   GLdouble objX,
   GLdouble objY,
   GLdouble objZ,
   const GLdouble *model,
   const GLdouble *proj,
   const GLint *view,
   GLdouble* winX,
   GLdouble* winY,
   GLdouble* winZ )
```

```
GLint gluUnProject(
   GLdouble winX,
   GLdouble winY,
   GLdouble winZ,
   const GLdouble *model,
   const GLdouble *proj,
   const GLint *view,
   GLdouble* objX,
   GLdouble* objY,
   GLdouble* objZ )
```

Pick & Select (2)

- gluUnProject(…):
  map window coordinates to object coordinates
- gluProject(…):
  map object coordinates to window coordinates

Pick & Select (3)

- glSelection Buffer

Example 5: Pick_Select

Demo: tetviewer

Performance Issues

- Bottleneck: the slowest part of an application

Performance Issues

- OpenGL applications
  - *Application* bottleneck
    - Application may not pass data fast enough to the OpenGL pipeline
  - *Transform-limited* bottleneck
    - OpenGL may not be able to process vertex transformations fast enough
  - *Fill-limited* bottleneck
    - OpenGL may not be able to rasterize primitives fast enough

Useful links & Resources

- OpenGL home
  http://www.opengl.org
- Nehe opengl lessions
  http://nehe.gamedev.net/lesson.asp?index=01
- GLUT examples
  http://www.xmission.com/~nate/glut.html
- Nate Robin' page
  http://www.xmission.com/~nate/opengl.html
- NVIDIA
  http://www.nvidia.com

- Reuse your & others' codes