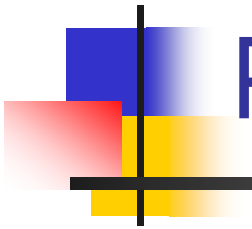
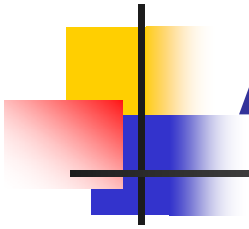


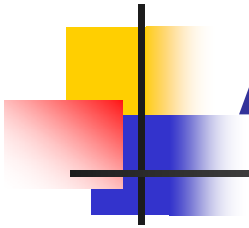
Pendahuluan OpenGL

Dr. Mohammad Iqbal



Apa itu OpenGL ?

- OpenGL is a low-level software interface to graphics hardware
  - No commands for performing windowing tasks or obtaining user input are included.
  - No high-level commands for describing models of 3D objects are provided.
- Why low level
  - You can access graphics hardware directly
  - Independent to window system, cross platform
- High level enough, so it's independent to graphics hardware
- Industry standard

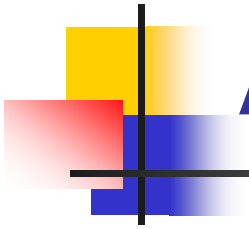


Apa itu OpenGL ?

- Interactive CG system that allows programmers to access graphics hardware
  - > Easy to use
  - > Programs run efficiently
  - > Hardware-independent

- Graphics API (*A*pplication *P*rogrammer's *I*nterface)
  - > A library of functions
  - > Others: DirectX (Microsoft), Java3D
  - > OGL evolved from GL (SGI)

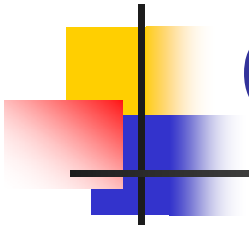
Apa itu OpenGL ?



- Contains over 200 functions
- Portable
  Implementations available for nearly all hardware and operating systems
- Portability → input or windowing are *not* included

  > Options for Windows: GLUT or MFC

  > GLUT = OGL Utility Toolkit
  Implementations of GLUT exist for most environments
  → GLUT somewhat portable

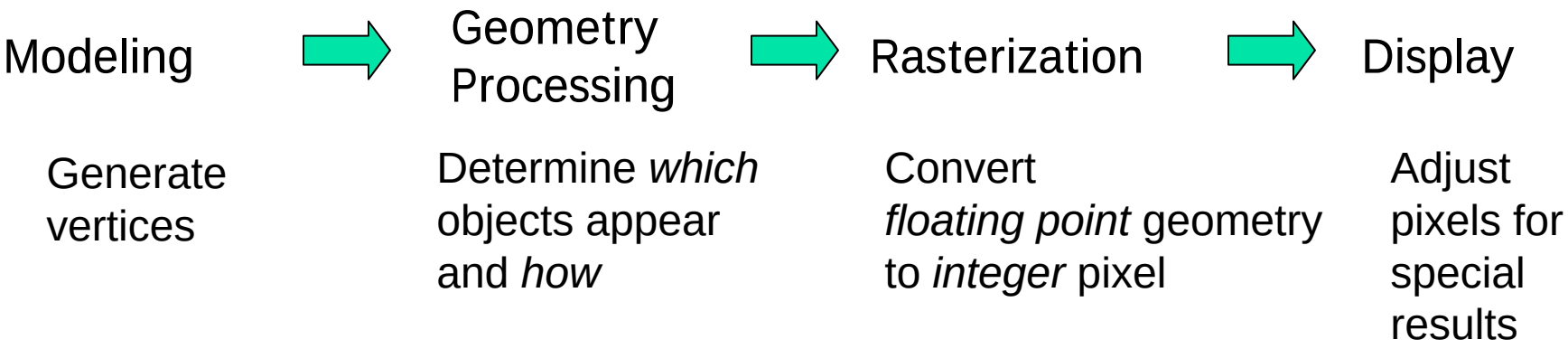


OGL Org

- Controlled by the OGL Architectural Review Board
  SGI, IBM, NVIDIA, ... -- some major players in CG
- Current: version 4.3
- very stable, upward compatible
- C and Fortran versions & unofficial Java
- www.opengl.org

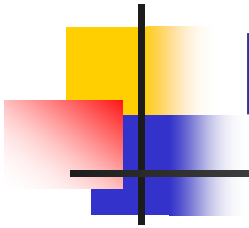
Element Utama App CG

Recall the *Viewing Pipeline* …



Your focus: Modeling and Geometric Processing

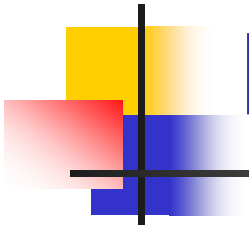
rasterization & display operations are mostly done for you or allow for special effects



Element Utama App CG

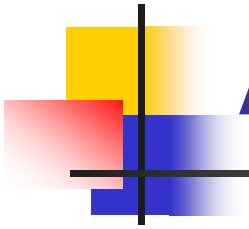
Flow of your basic CG apps will be

- Initializing functions (os and windowing) – glut
- Input, interactive funtions – glut
- Specify a set of objects to render – ogl
- Describe properties of these objects – ogl
- Define how these objects should be viewed – ogl
- Termination (os, windowing) -- glut



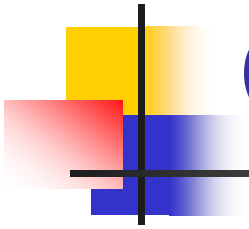
Element Utama App CG

- Objects: geometric & image
  - > Geometric: defined by points, lines, polygons
    - -- called *geometric primitives*
    - -- note: smooth curves and surfaces rendered in a discretized form
  - > Geometric objects are the focus in this course
- Definition of geometric objects is separate from definition of appearance
  - > Ex: color and style of a line call separate from geometry
  - > Ex: define object as you wish, and then define its positioning and how we will view it



Apa yang OpenGL hasilkan ?

- Draw with points, lines, and polygons.
- Matrix Operations (Transformations)
- Hidden Surface Removal (Z-Buffer)
- Lighting (Phong model)
- Gouraud Shading
- Texture mapping
- Pixels operations

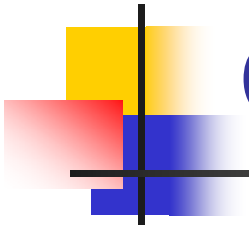


OpenGL sebagai *State Machine*

Two types of function calls in OGL:

1. input: specify objects
2. set state: determines how objects processed

Ex: set the current color to red;
after that, all objects drawn will be red

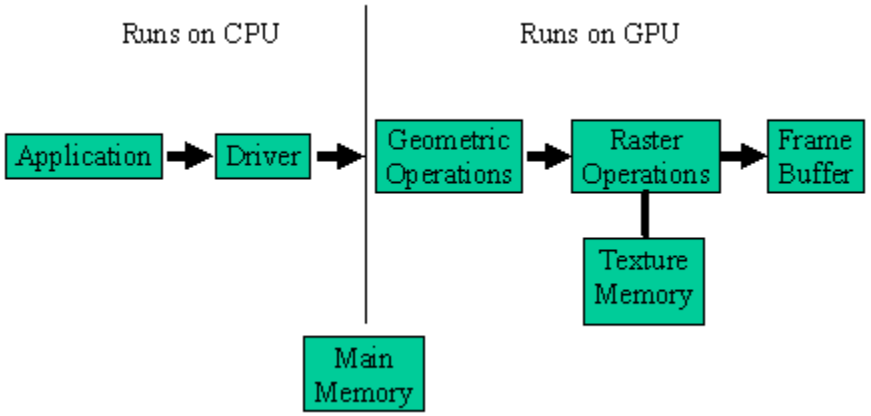


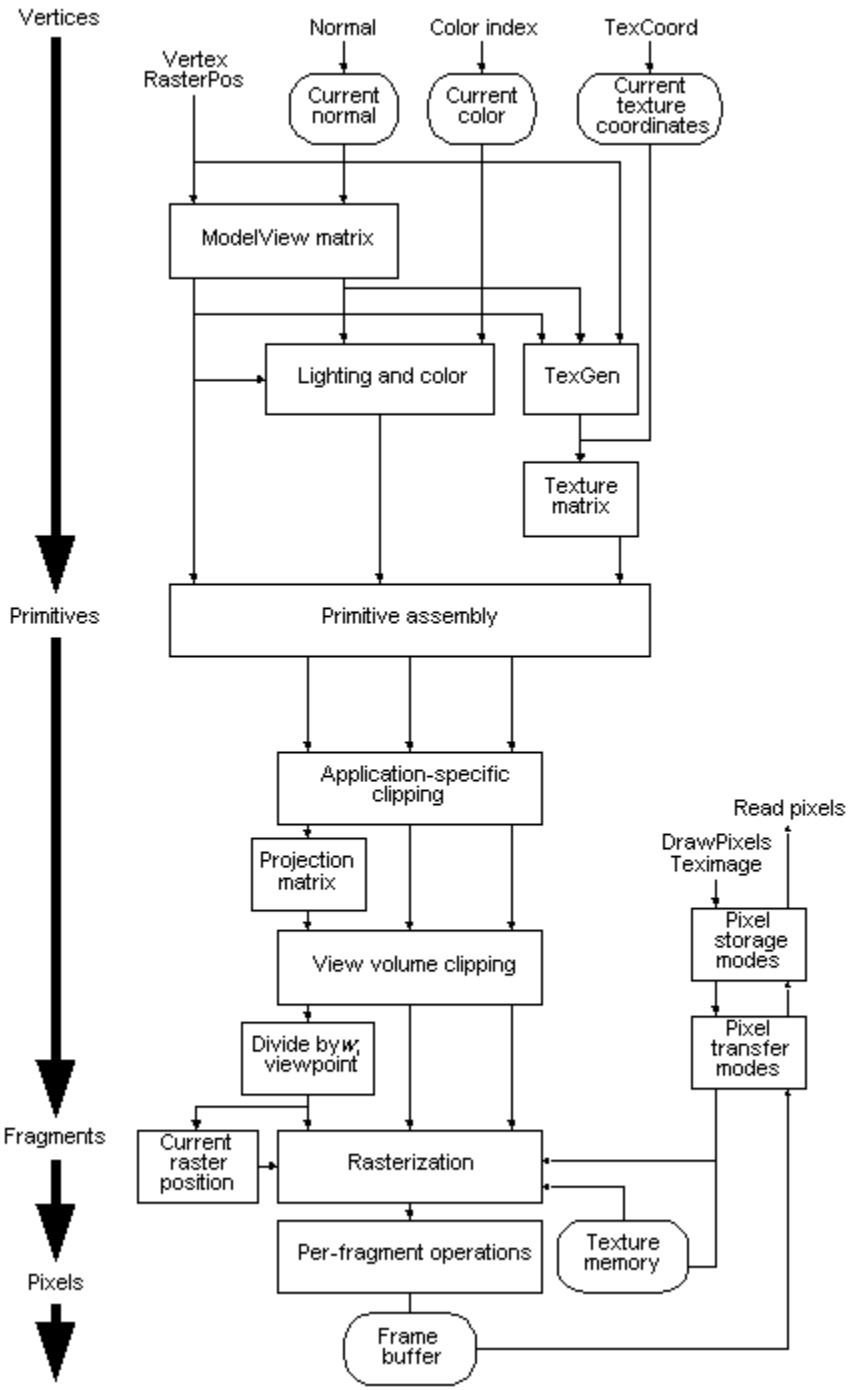
OGL & Penyajian secara Pipeline

The pipeline architecture has a great influence on the `spirit' of OGL

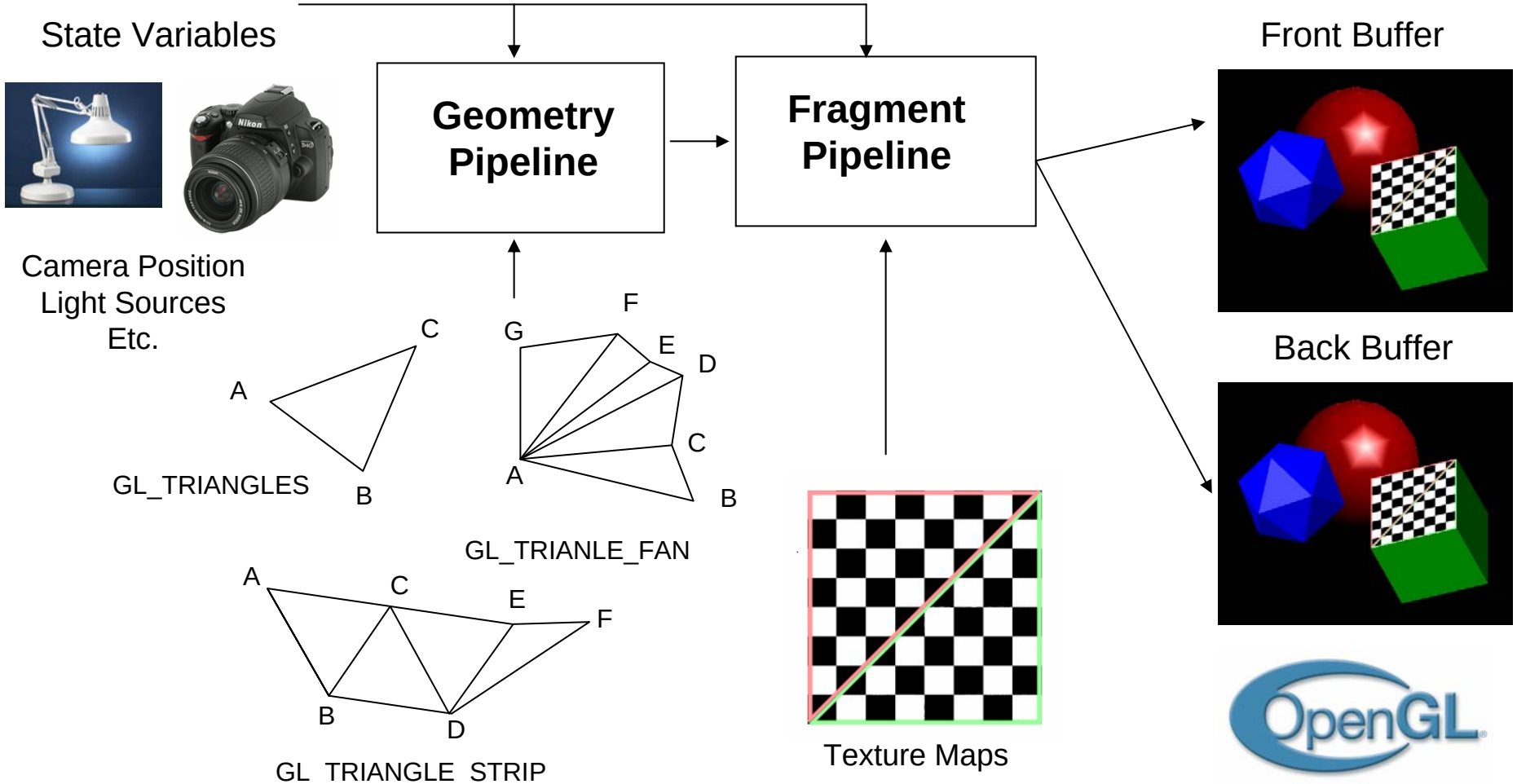
- Designed for speed!
- Special hardware for each step
- Pipeline model means that primitives processed independently → no global effects (e.g. shadows)

OpenGL Graphics Pipeline

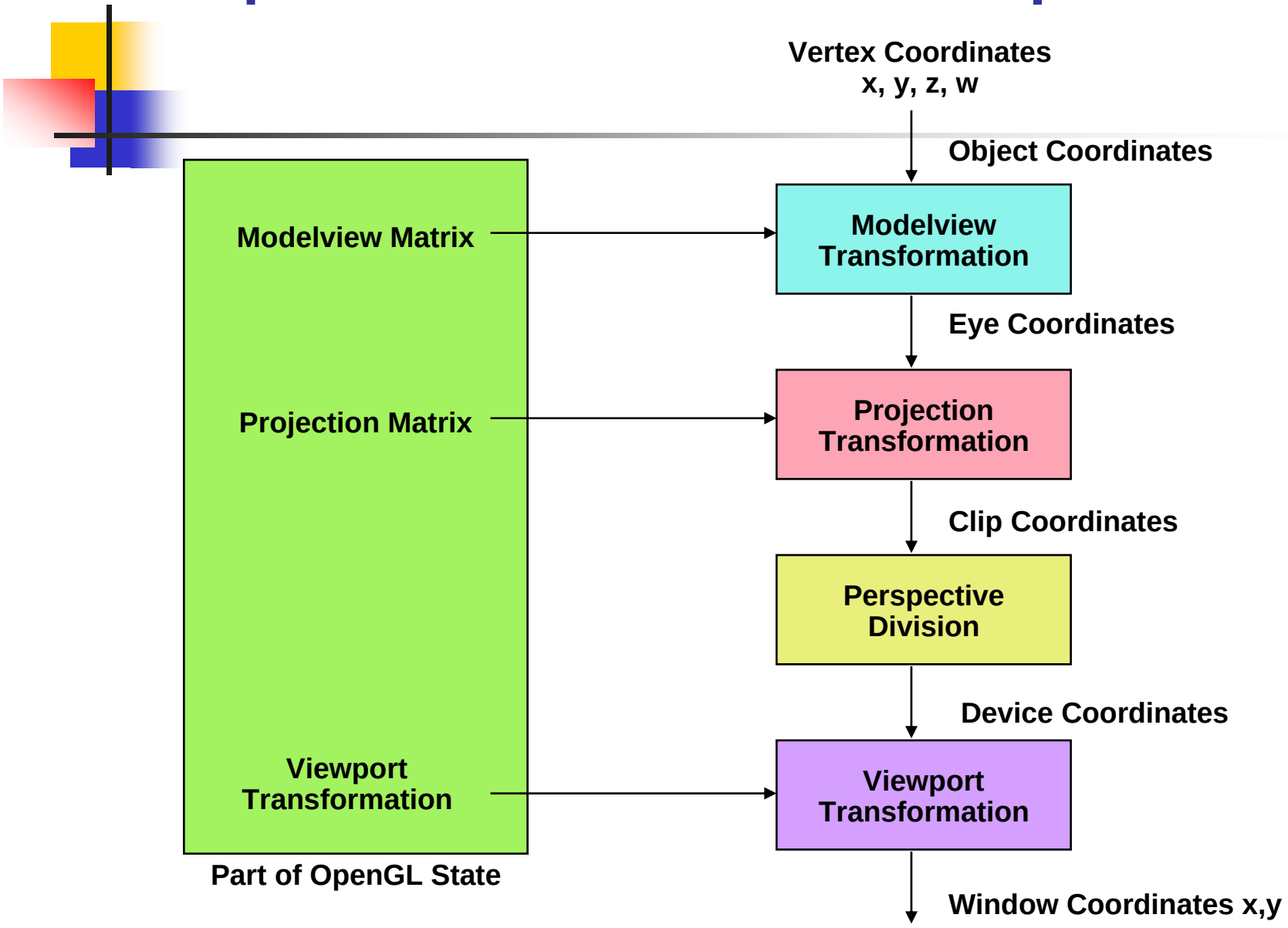




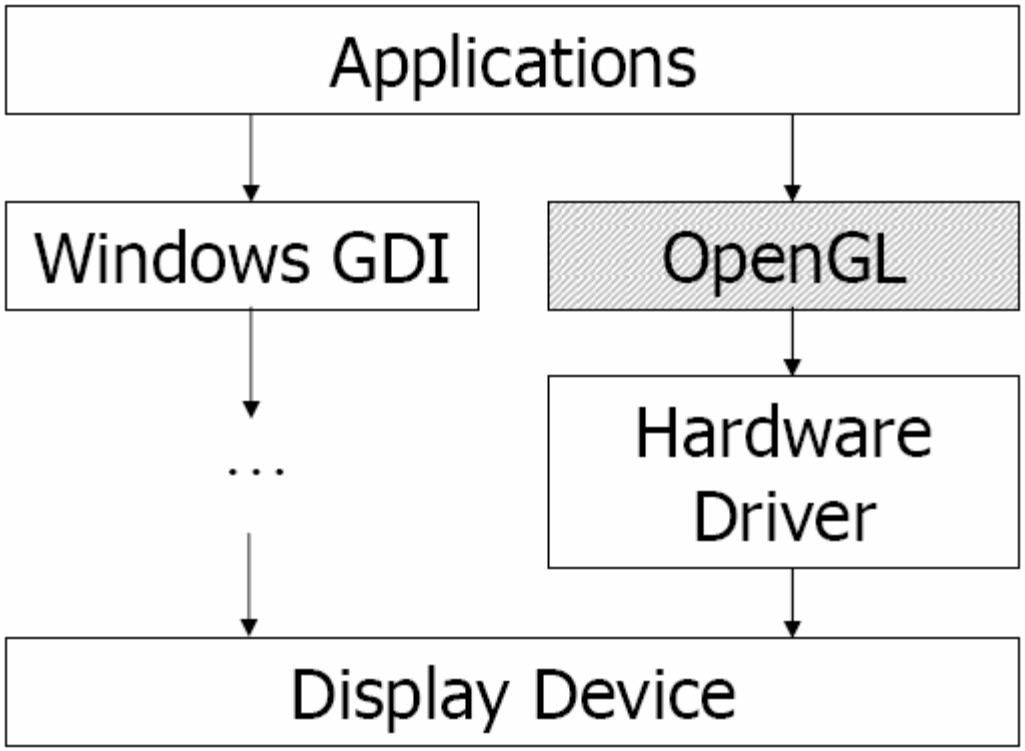
OpenGL Graphics Pipeline

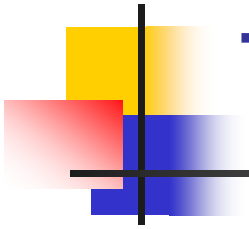


OpenGL Geometric Pipeline



Hirarki API


Applications
Windows GDI
OpenGL
...
Hardware Driver
Display Device

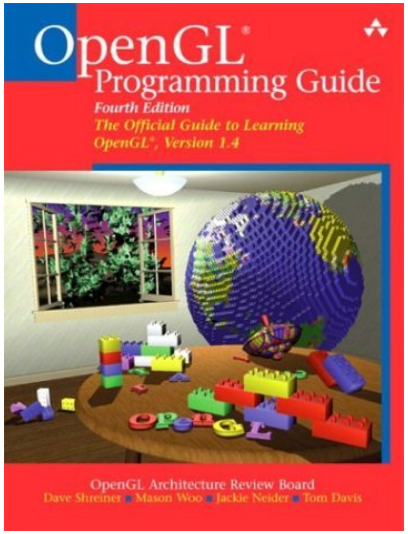


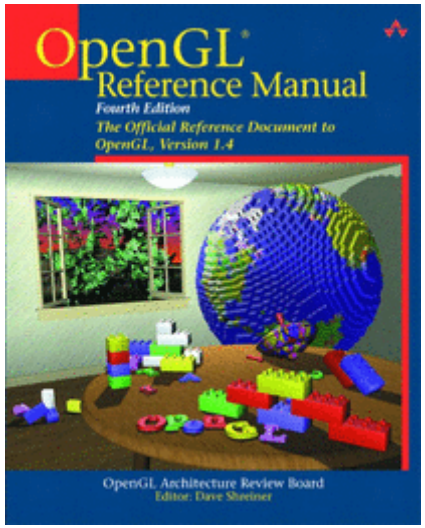
Tentang *Buffer*

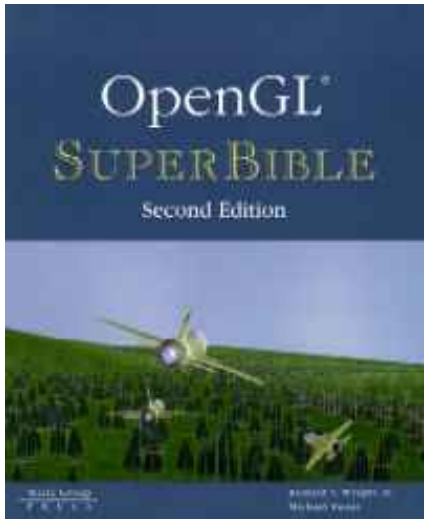
- A buffer is a memory area in the graphics hardware for some special purposes.
- An OpenGL system can manipulate the four buffers:
  - Color buffers
  - Depth buffer (Z-Buffer)
  - Stencil buffer
  - Accumulation buffer
    - – Will be described in later chapter.

Buku dan Web Link

- **Books**
  - **OpenGL Programming Guide (Red-book)**
  - **OpenGL Reference Manual (Blue-book)**
  - **OpenGL Super Bible**

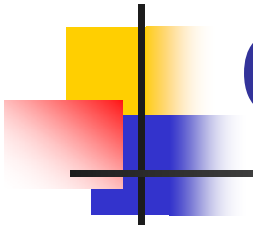






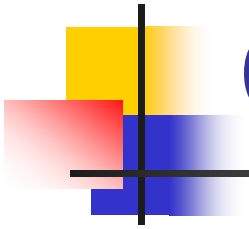
Web

- http://www.opengl.org/documentation/specs/glut/spec3/spec3.html
- http://www.ncrg.aston.ac.uk/~cornfosd/graphics/opengl/openglman.html
- NeHe's OpenGL Lessons
  http://users.polytech.unice.fr/~buffa/cours/synthese_image/DOCS/Tutor



Getting Started

- You'll be using VC++ environment
- System folders should already have OGL dlls:
  opengl32.dll and glu32.dll
- Corresponding lib files in ..\VC\lib
  opengl32.lib, glu32.lib, glaux.lib
- Include files in ..\VC\include\GL
  gl.h, glu.h, glaux.h
- GLUT files from the web (www.xmission.com/~nate/glut.html)
  or the class resources page
  glut.h, glut32.lib, glut32.dll
  On your own system: files go in the same place as corresponding ogl files



Getting Started

To start your own program in VC++ 6.0 do the following.

0) Start VC++
1) File->New->Open a console application
2) Select an "empty project" and pick a name and directory
3) File->New->C++ source (pick a name and directory)
4) You are ready to go!

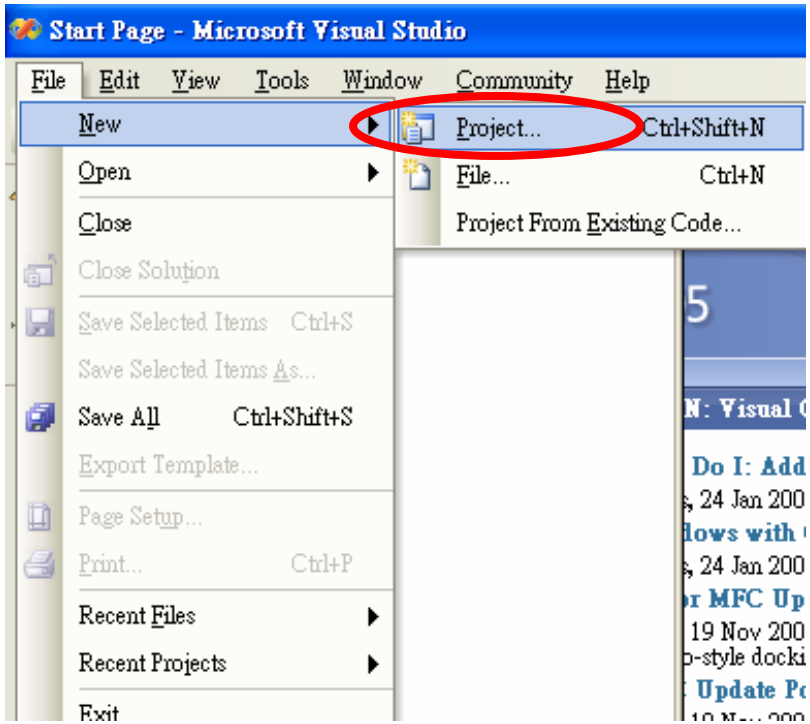
See class resources page for details for .NET

To compile, build and execute, see the “Build” menu (or toolbar)

See the Lighthouse3D website (top page) for instructions for getting rid of the extra console window.

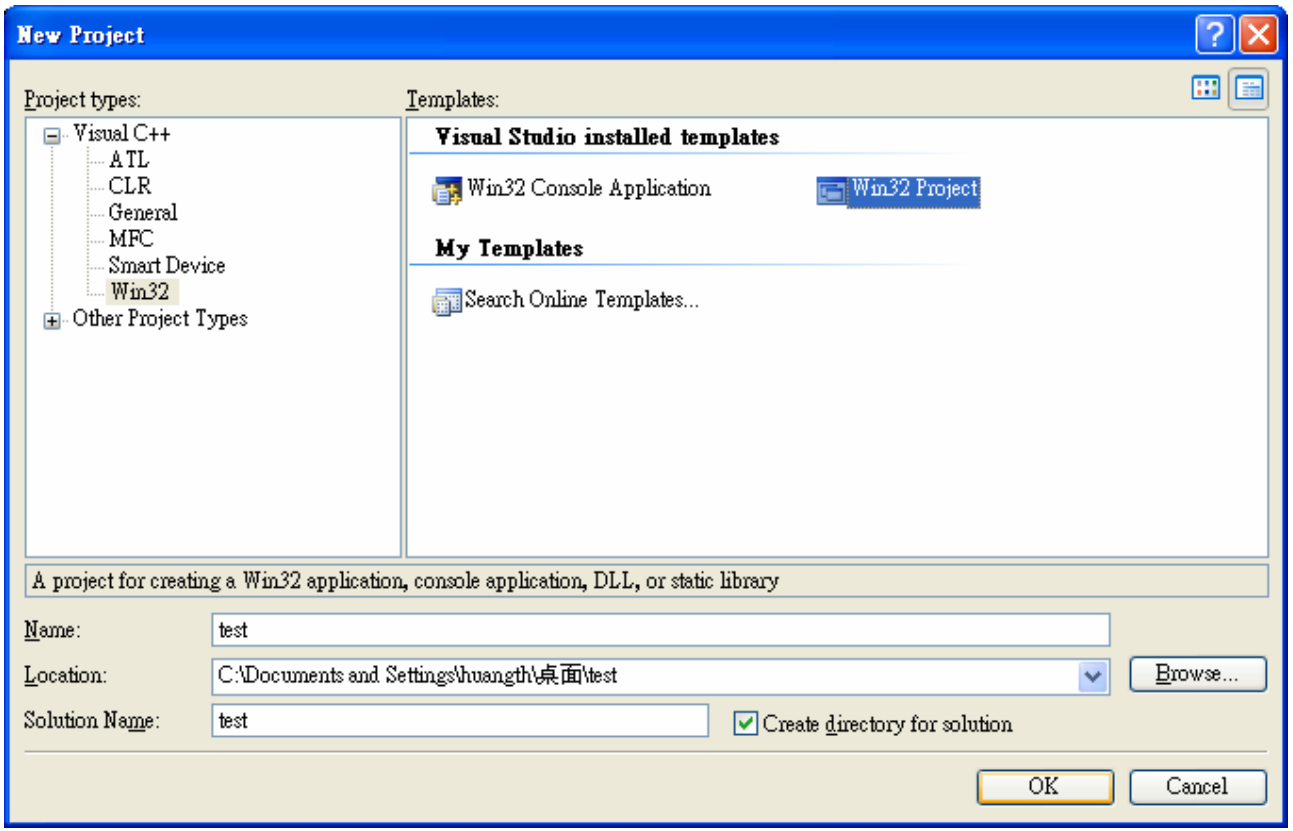
Setup for a OpenGL Program(VC8.0) 1/8

- Microsoft Visual C++ 8.0
  - New Blank Project



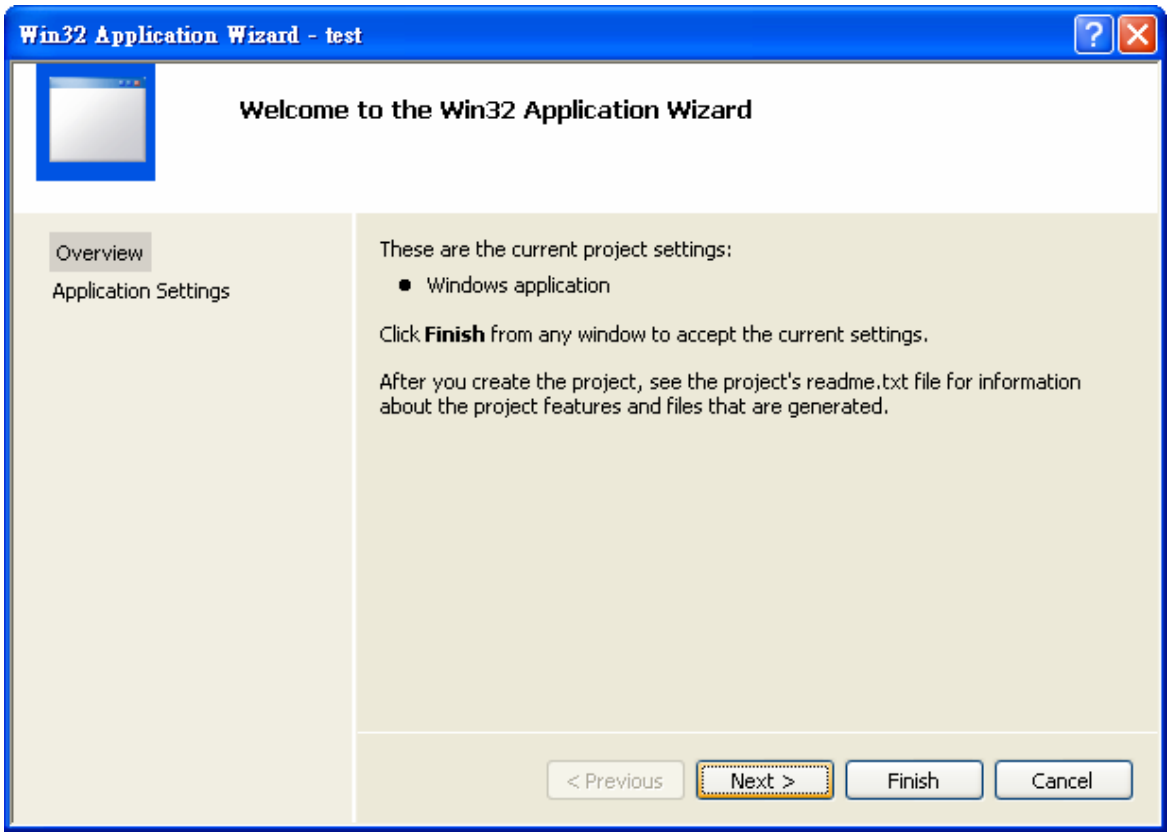
Setup for a OpenGL Program(VC8.0)

- New Win32 Project



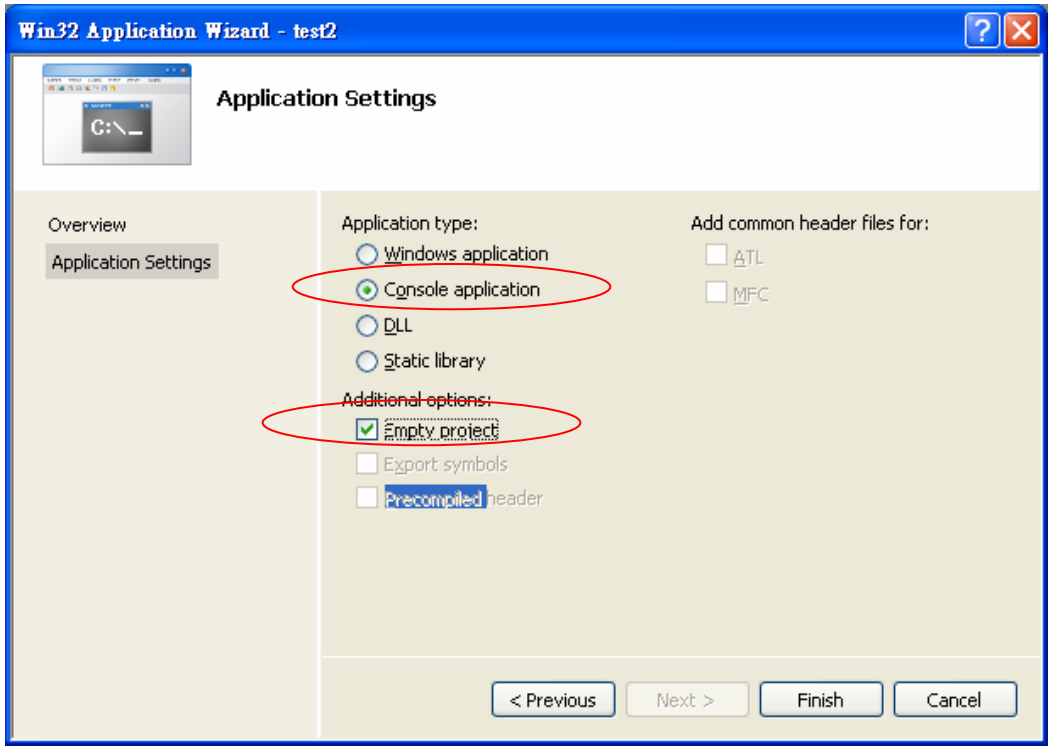
Setup for a OpenGL Program(VC8.0)

- Next



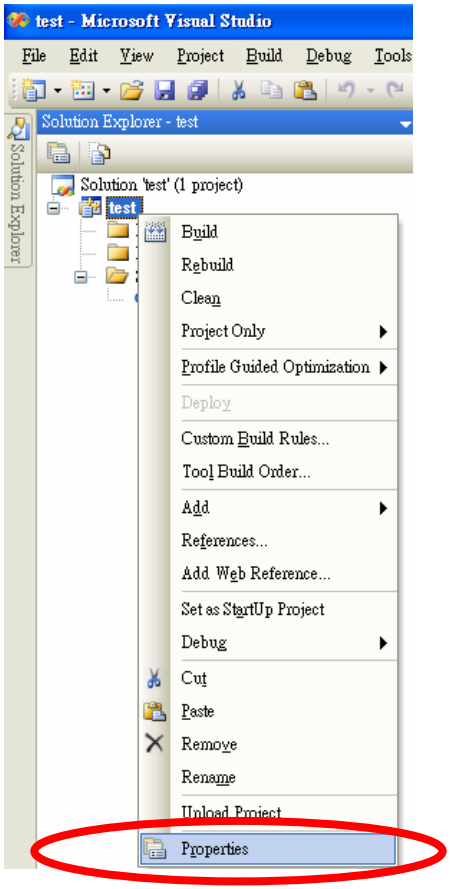
Setup for a OpenGL Program(VC8.0)

- Use Console application Empty , Finish



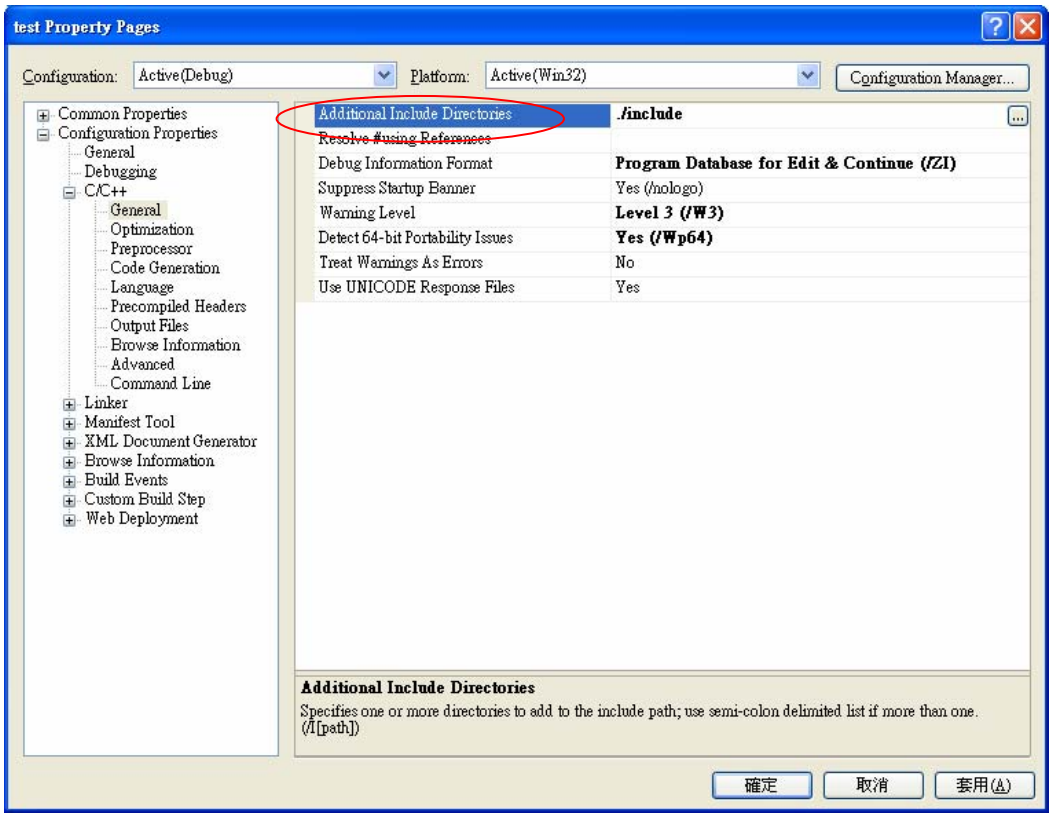
Setup for a OpenGL Program(VC8.0)

- Set properties



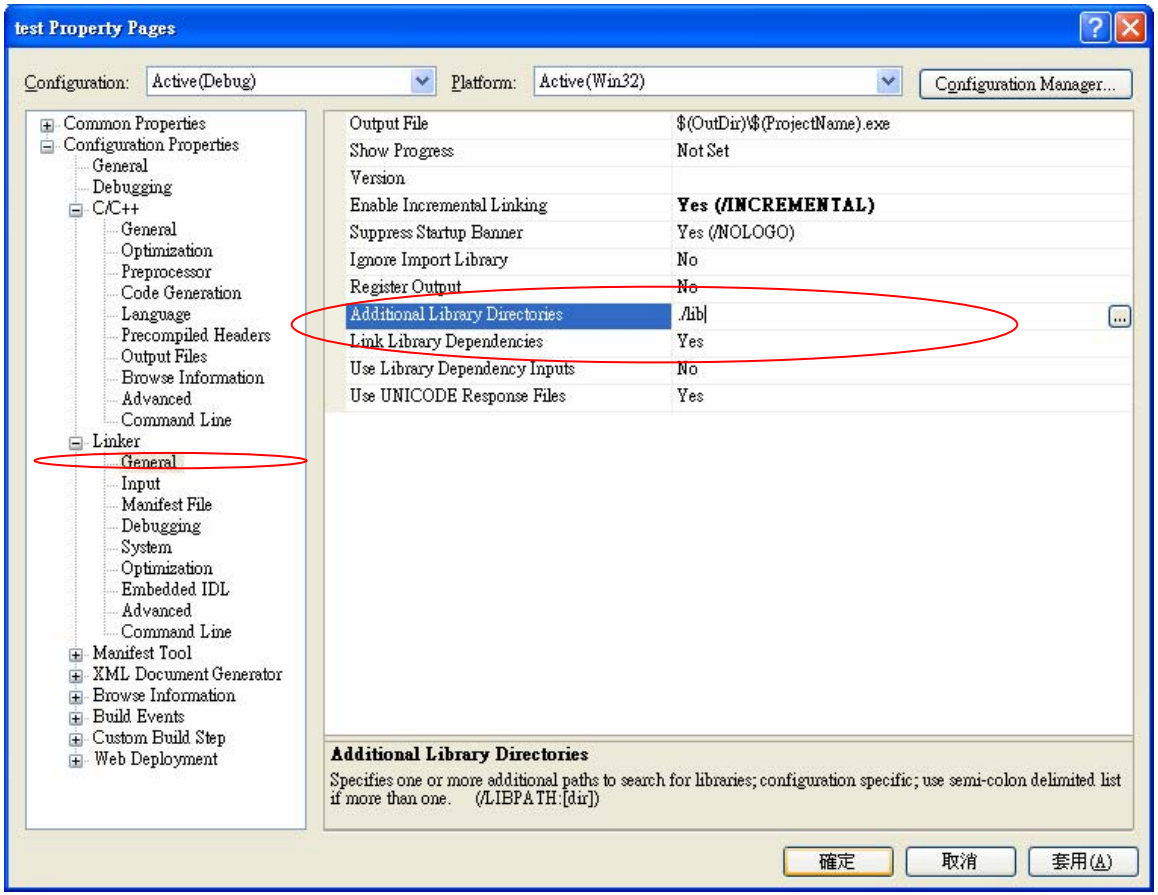
Setup for a OpenGL Program(VC8.0) 6/8

- Set include to the location of glut.h



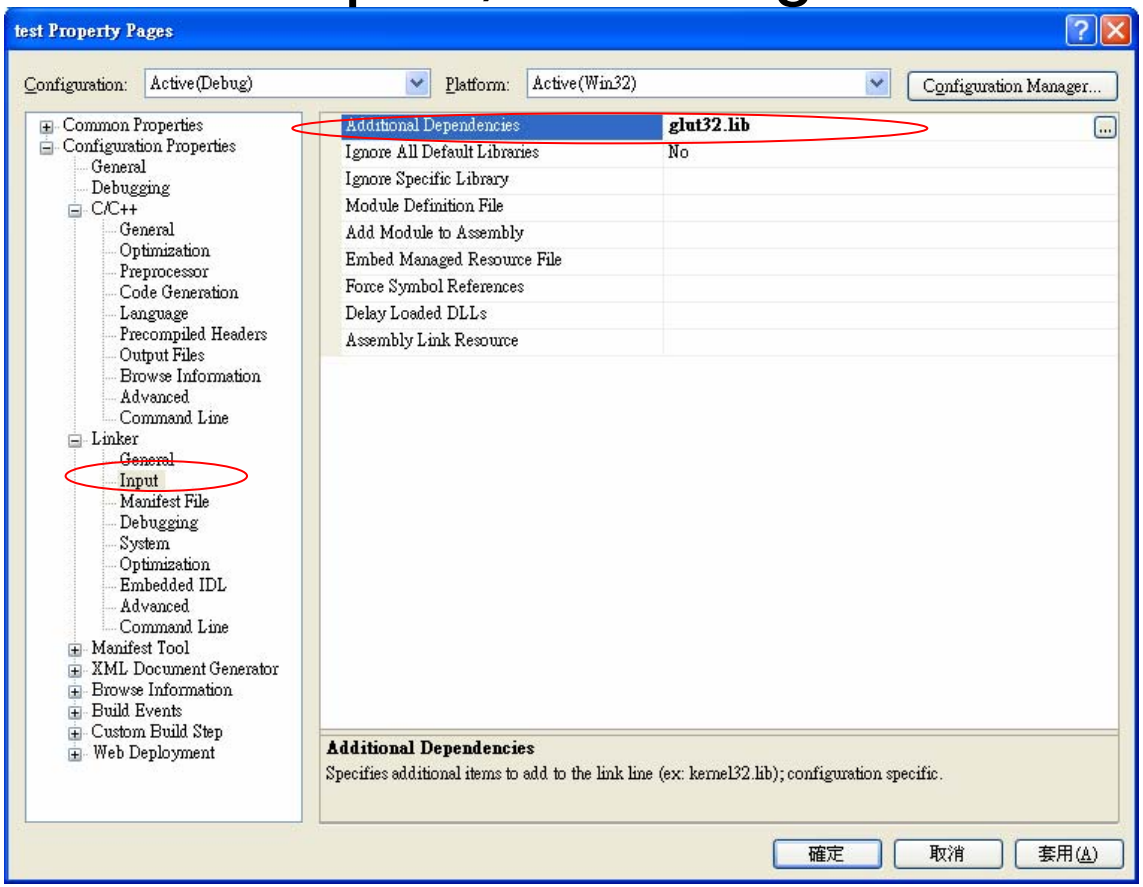
Setup for a OpenGL Program(VC8.0) 7/8

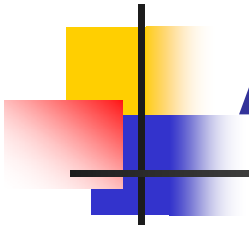
- Set link to the location of the lib file



Setup for a OpenGL Program(VC8.0) 8/8

- Set link input ,add the glut32.lib

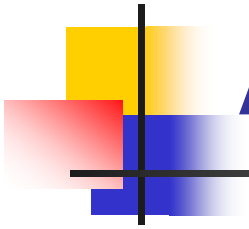




Aturan Dasar program OGL

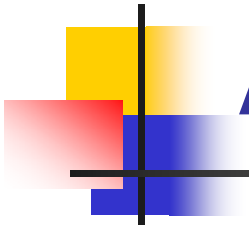
basics.c → Concepts illustrated:

- creating a window (glutInit .. glutCreateWIndow)
- event loops and callbacks (glutDisplayFunc)
- basic coordinate transformations (gluOrtho2D)
- aspect ratio
- pipeline elements
- setting attributes (glColor3f)
- flow of a program
- avoid the “silent program”
  – what you see can be deceiving!



Aturan dalam Programming

- GL library for OGL
  glVertex3f()
- GLU library for extra functions, built on top of GL
  gluOrtho2D()
- GLUT library for interaction functions
  glutInit()



Aturan dalam Programming

OGL uses standard C data types

floats, ints, doubles, ...

But for portability: better to use OGL data types

GLfloat, GLdouble, ...

void glVertex3f(GLfloat x, GLfloat y, GLfloat z)

Aturan dalam Programming

Flexibility of calling functions

several ways to call the same functionality

glVertex{234}{sidf}(TYPE coords, …);

glVertex{234}{sidf}v(TYPE *coords);

common practice: use the “star” notation

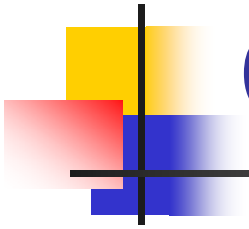
Ex: glVertex*

refers to all routines that define a vertex

Examples

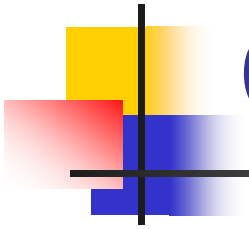
```
Glint ix, iy
GLfloat x, y, z;
GLfloat point[3];
…. (assign values)

glVertex2i(ix, iy);
glVertex2f(x, y);
glVertex3f(x, y, z);
glVertex3fv(point);
```



Contoh Program OpenGL 1/4

```
#include “glut.h”
void display();
void reshape(GLsizei, GLsizei);
void main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutCreateWindow("sample");
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
}
```



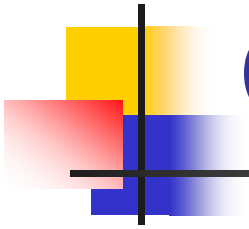
Contoh Program OpenGL 2/4

```
void display(){
   glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(1.0f, 1.0f, 1.0f);
   glutSolidTeapot(1.0);
   glFlush();
}
```

Contoh Program OpenGL 3/4

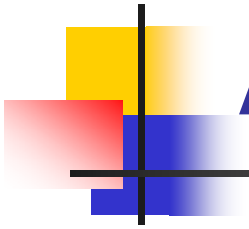
```
void reshape(GLsizei w, GLsizei h){
   glViewport(0, 0, w, h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glFrustum(-0.5, 0.5, -0.5, 0.5, 1.0, 20.0);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0,
   0.0);
}
```

Contoh Program OpenGL 4/4



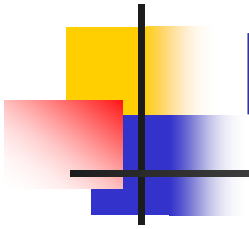
OpenGL Utility Toolkit (GLUT)

- A window system-independent toolkit to hide the complexity of differing window system APIs.
- Providing following operations:
  - Initializing and creating window
  - Handling window and input events
  - Drawing basic 3D objects
  - Running the program



A Word on GLUT

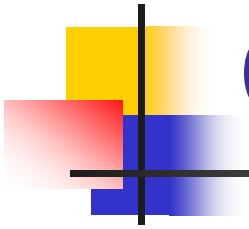
- About 50 function calls
- Sits on top of wgl
  (windows windowing system control)
  (for other platforms: glX, agl)
- A nice tutorial:
  http://www.lighthouse3d.com/opengl/glut/



Klasifikasi OGL Functions

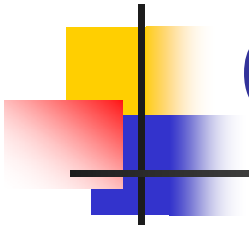
- Primitive Functions
  - > define the things you draw
  - > geometric and images
- Attribute Functions
  - > appearance of primitives
  - > colors, line types, material properties, lights, textures
- Viewing Functions
  - > properties of the “virtual camera”
- Control Functions
  - > turn OGL functionality on/off and query state
- Windowing Functions
  - > GLUT – not OGL – window, mouse, and keyboard control

All but the primitive functions are state changing



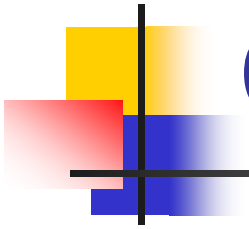
GLUT Functions 1/7

- void glutInit(int *argcp, char **argv);
  - Initializing the GLUT library
  - Should be called before any other GLUT funcitons
  - http://www.opengl.org/resources/libraries/glut/spec3/node10.html
- void glutInitDisplayMode(unsigned int mode);
  - Specify a display mode for windows created.
  - GLUT_RGB / GLUT_RGBA / GLUT_INDEX
  - GLUT_SINGLE / GLUT_DOUBLE
  - GLUT_DEPTH / GLUT_STENCIL / GLUT_ACCUM
  - http://www.opengl.org/resources/libraries/glut/spec3/node12.html



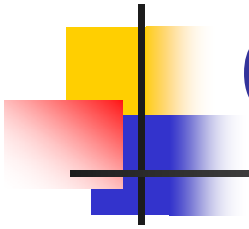
GLUT Functions 2/7

- void glutInitWindowSize(int width, int height);
- void glutInitWindowPosition(int x, int y);
  - Initializing the window position and size.
  - http://www.opengl.org/resources/libraries/glut/spec3/node11.html
- int glutCreateWindow(char *name);
  - Open a window with previous settings.
  - http://www.opengl.org/resources/libraries/glut/spec3/node16.html#383



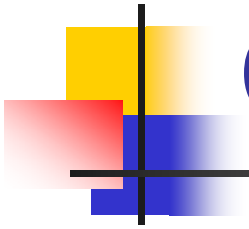
GLUT Functions 3/7

- void glutDisplayFunc(void (*func)(void));
  - Called whenever the contents of the windows need to be redrawn.
  - Put whatever you wish to draw on screen here.
  - Use glutPostRedisplay() to manually ask GLUT to recall this display function
  - http://www.opengl.org/resources/libraries/glut/spec3/node46.html



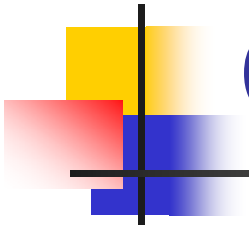
GLUT Functions 4/7

- void glutReshapeFunc(void (*func)(int width, int height));
  - Called whenever the window is resized or moved.
  - You should always call glViewport() here to resize your viewport.
  - http://www.opengl.org/resources/libraries/glut/spec3/node48.html



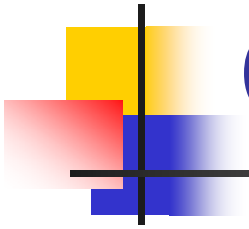
GLUT Functions 5/7

- void glutKeyboardFunc(void (*func)(unsigned char key, int x, int y));
  - Sets the keyboard callback for the current window.
  - http://www.opengl.org/resources/libraries/glut/spec3/node49.html
- void glutIdleFunc(void (*func)(void));
  - Sets the global idle callback.
  - http://www.opengl.org/resources/libraries/glut/spec3/node63.html



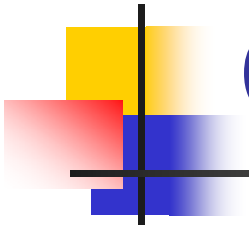
GLUT Functions 6/7

- void glutMouseFunc(void (*func)(int button, int state, int x, int y));
  - sets the mouse callback for the current window.
  - http://www.opengl.org/resources/libraries/glut/spec3/node50.html
- void glutMotionFunc(void (*func)(int x, int y));
  - set the motion callbacks respectively for the current window.
  - http://www.opengl.org/resources/libraries/glut/spec3/node51.html



GLUT Functions 7/7

- void glutMainLoop(void);
  - Enter the GLUT processing loop and never returns.
  - http://www.opengl.org/resources/libraries/glut/spec3/node14.html#376
- void glutPostRedisplay(void);
  - marks the current window as needing to be redisplayed.
  - http://www.opengl.org/resources/libraries/glut/spec3/node20.html#465



GLUT Objects

- GLUT provides the follow objects:
  - Sphere, cube, torus, icosahedron, ocrtahedron, tetrahedron, teapot, dodecahedron, cone.
  - Both wireframe and solid
  - Ex: glutSolidSphere(1.0, 24, 24)
  - Ex: glutWireCube(1.0)
  - http://www.opengl.org/resources/libraries/glut/spec3/node80.html#SECTION00012000000000000000

Appendix: Example of Keyboard Callback Function

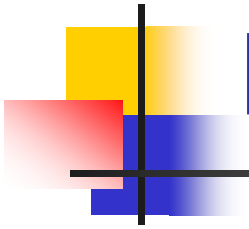
```
void keyboard(unsigned char key, int x, int y)
{
   printf("you press the key %c \n", key);
   printf("the mouse is on %d %d \n", x, y);
}
```

Appendix: Example of Mouse Callback Function

```
int startX, startY;
void mouse(int button, int state, int x, int y){
    if (state == GLUT_DOWN){
        startX = x;
        startY = y;
    }
    else if(state == GLUT_UP){
        printf("the mouse moves %d %d \n", x - startX,
    y - startY);
    }
}
```

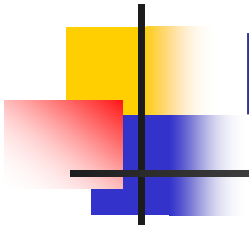
Appendix: Example of Motion Callback Function

```
void motion(int x, int y)
{
  printf("the mouse is moving to %d %d", x, y);
}
```



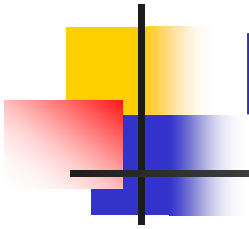
Example 1/4

```
#include “glut.h”
void display();
void reshape(GLsizei w, GLsizei h);
void main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(250, 250);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Drawing sample");
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMainLoop();
}
```



Example 2/4

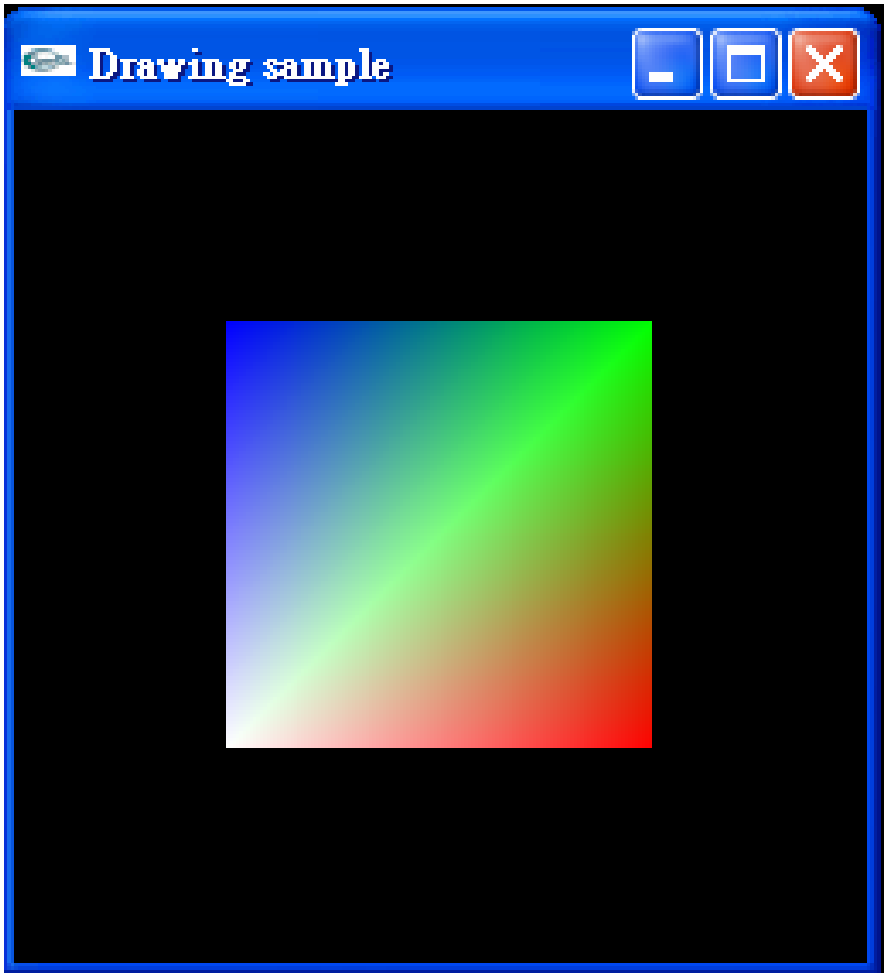
```
void reshape(GLsizei w, GLsizei h){
   glViewport(0, 0, w, h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(-2.0f, 2.0f, -2.0f, 2.0f, -2.0f, 2.0f);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}
```

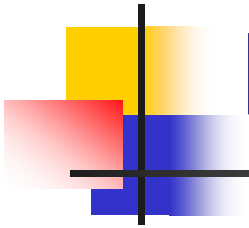


Example 3/4

```
void display(){
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_POLYGON);
            glColor3d(1.0f, 1.0f, 1.0f);
            glVertex3f(-1.0f, -1.0f, 0.0f);
            glColor3d(1.0f, 0.0f, 0.0f);
            glVertex3f(1.0f, -1.0f, 0.0f);
            glColor3d(0.0f, 1.0f, 0.0f);
            glVertex3f(1.0f, 1.0f, 0.0f);
            glColor3d(0.0f, 0.0f, 1.0f);
            glVertex3f(-1.0f, 1.0f, 0.0f);
    glEnd();
    glFlush();
}
```

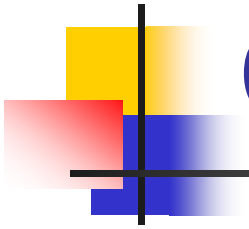
Example 4/4





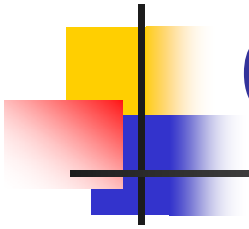
Data Type

OpenGL Type	Internal representation	C-Language Type	Suffix
GLbyte	8-bit integer	signed char	b
GLshort	16-bit integer	short	s
GLint, GLsizei	32-bit integer	int or long	i
GLfloat	32-bit floating	float	f
GLflampf	pointer		
GLfouble	64-bit floating	double	d
GLclampd	pointer		
Glubyte	8-bit unsigned integer	unsigned char	ub
GLuboolean	8-bit unsigned integer	unsigned char	ub
GLushort	16-bit unsigned integer	unsigned short	us
GLuint, GLenum	32-bit unsigned integer	unsigned long	ui
GLbitfield	32-bit unsigned integer		



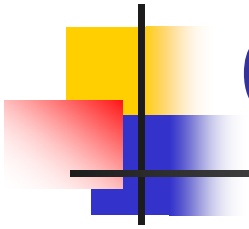
Clear the buffers 1/2

- void glClearColor(*GLclampf, GLclampf, GLclampf, GLclampf*)
  - Set the current values for use in cleaning color buffers in RGBA mode.
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc01_0rhu.asp
- void glClearDepth(*GLclampd*)
  - Set the current values for use in cleaning depth buffer.
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc01_4j1k.asp



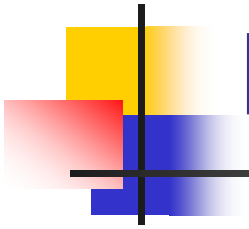
Clear the buffers 2/2

- void glClear(*GLbitfield*)
  - Clear the specified buffers to their current clearing values.
    - GL_COLOR_BUFFER_BIT
    - GL_DEPTH_BUFFER_BIT
    - GL_ACCUM_BUFFER_BIT
    - GL_STENCIL_BUFFER_BIT
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc01_8koi.asp



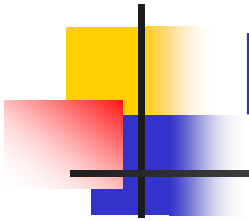
Color representation

- RGBA: red, green, blue, alpha
- Each channel has intensity from 0.0~1.0
  - Values outside this interval will be clamp to 0.0 or 1.0
  - Alpha is used in blending and transparency
- Specify a color:
  - glColor{34}{sifd}[v](…)
    - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc01_62b6.asp



Points, Lines and Polygons 1/4

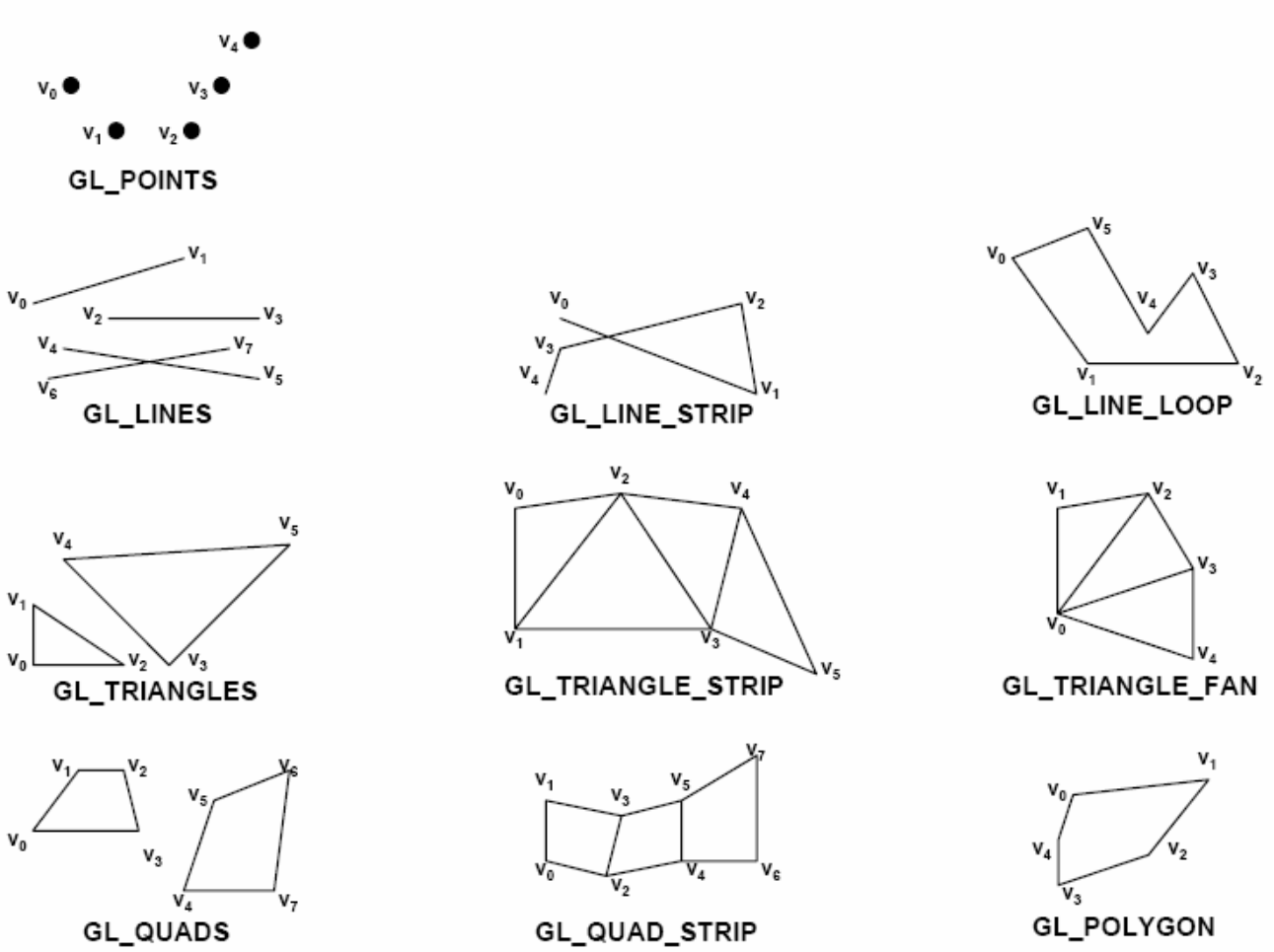
- Describe points, lines, polygons
  - void glBegin(*GLenum*)
    - Marks the beginning of a vertex-data list
    - The mode can be any of the values in next page
  - void glEnd()
    - Marks the end of a vertex-data list
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc01_9u3y.asp

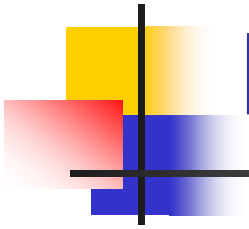


Points, Lines and Polygons 2/4

Value	Meaning
GL_POINTS	individual points
GL_LINES	pairs of vertices interpreted as individual line segments
GL_LINE_STRIP	serious of connected line segments
GL_LINE_LOOP	same as above, with a segment added between last and first vertices
GL_TRIANGLES	triples of vertices interpreted as triangles
GL_TRIANGLE_STRIP	linked strip of triangles
GL_TRIANGLE_FAN	linked fan of triangles
GL_QUADS	quadruples of vertices interpreted as four-sided polygons
GL_QUAD_STRIP	linked strip of quadrilaterals
GL_POLYGON	boundary of a simple, convex polygon

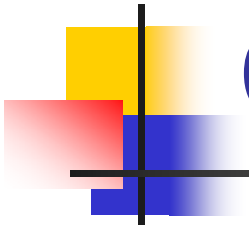
Points, Lines and Polygons 3/4





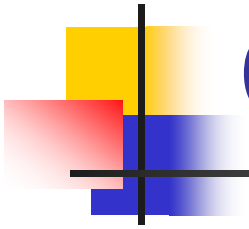
Points, Lines and Polygons 4/4

- Specifying the vertices
  - glVertex{234}{sifd}[v](…)
    - Specifies a vertex for use in describing a geometric object
    - Can only effective between a **glBegin()** and **glEnd()** pair
    - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc03_2kag.asp



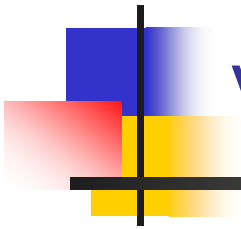
Completion of drawing 1/2

- void glFlush()
  - Forces previously issued OpenGL commands to begin execution.
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc02_8sa0.asp
- void glFinish()
  - Forces previous issued OpenGL commands to complete.
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc02_3aqw.asp



Completion of drawing 2/2

- void glutSwapBuffers()
  - Swap front and back buffers.
  - http://www.opengl.org/resources/libraries/glut/spec3/node21.html



Viewing and Transformation

OpenGL program -3

```
#include “glut.h”

static GLfloat year=0.0f, day=0.0f;
void display();
void reshape(GLsizei , GLsizei );
void idle();
void keyboard(unsigned char , int, int);
```

OpenGL program -3

```
void display()
{   // clear the buffer
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW); //model view
    glColor3f(1.0, 1.0, 1.0);
    glutWireSphere(1.0, 20, 16);      // the Sun
    glPushMatrix();
            glRotatef(year, 0.0, 1.0, 0.0);
            glTranslatef(3.0, 0.0, 0.0);
            glRotatef(day, 0.0, 1.0, 0.0);
            glutWireSphere(0.5, 10, 8);            // the Planet
    glPopMatrix();
    // swap the front and back buffers
    glutSwapBuffers();
}
```

OpenGL program -3

```
void reshape(GLsizei w, GLsizei h)
{   // viewport transformation
    glViewport(0, 0, w, h);
    // projection transformation
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, (GLfloat)w/(GLfloat)h, 1.0, 20.0);
    // viewing and modeling transformation
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(0.0, 3.0, 5.0,        // eye
              0.0, 0.0, 0.0,        // center
              0.0, 1.0, 0.0);       // up
}
```

OpenGL program -3

```
// GLUT idle function
void idle()
{
        day += 10.0;
        if(day > 360.0)
                day -= 360.0;
        year += 1.0;
        if(year > 360.0)
                year -= 360.0;
        // recall GL_display() function
        glutPostRedisplay();
}
```

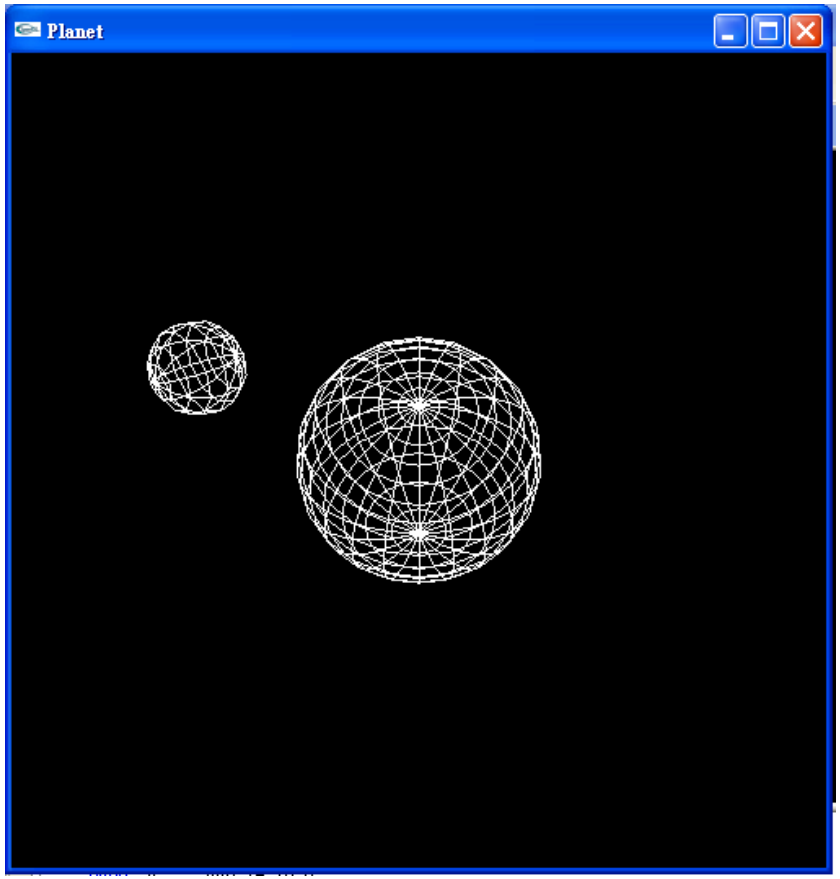
OpenGL program -3

```
// GLUT keyboard function
void keyboard(unsigned char key, int x, int y)
{   switch(key)
    {       case 'd': day += 10.0;
                      if(day > 360.0)              day -= 360.0;
                      glutPostRedisplay();         break;
            case 'y': year += 1.0;
                      if(year > 360.0)             year -= 360.0;
                      glutPostRedisplay();         break;
            case 'a': // assign idle function
                      glutIdleFunc(idle);          break;
            case 'A': glutIdleFunc(0);             break;
            case 27: exit(0);
    }
}
```

OpenGL program -3

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutCreateWindow("Planet");
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
    return 0;
}
```

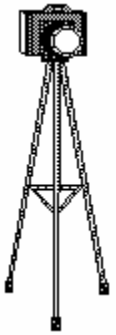
OpenGL program -3



The camera analogy 1/2

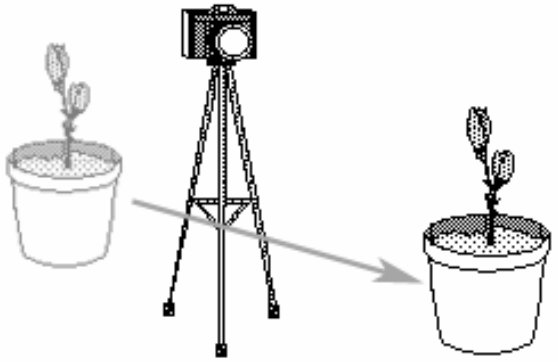
- Set up tripod and pointing the camera at the scene

  (viewing transformation)



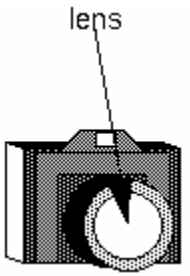
- Arrange the scene to be photographed into the desired composition

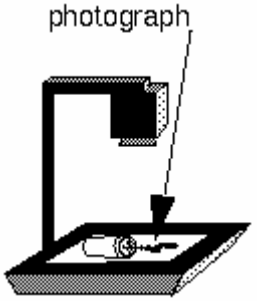
  (modeling transformation)



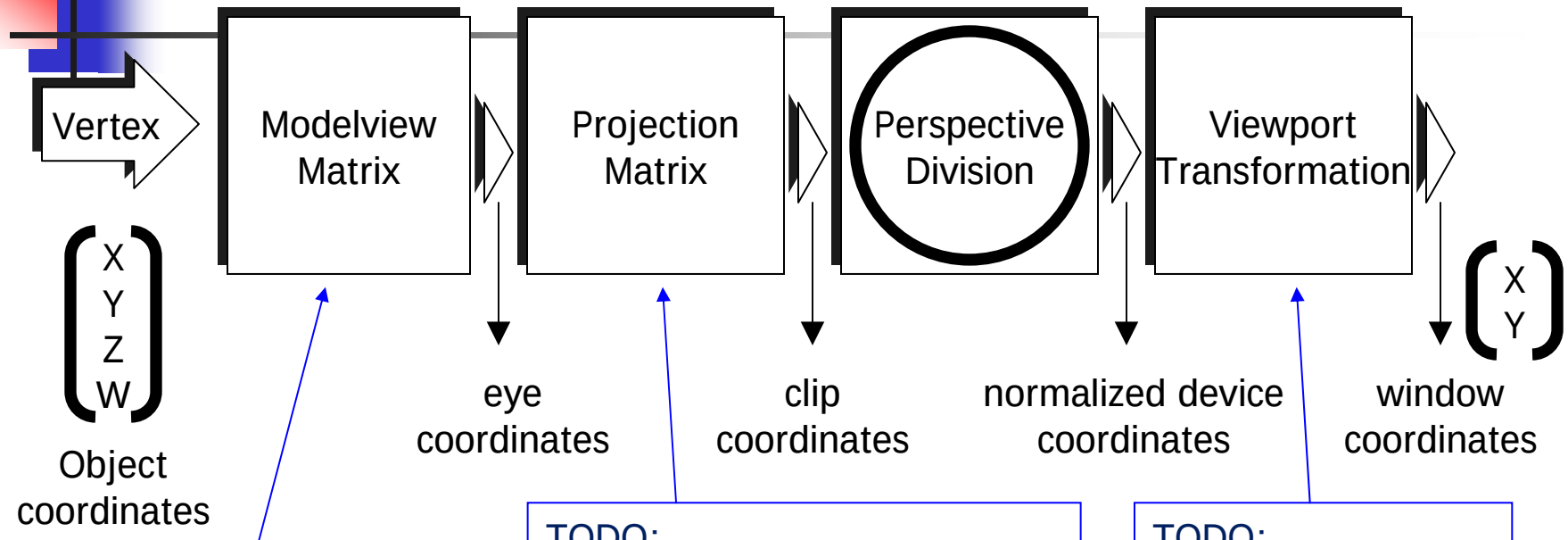
The camera analogy 2/2

- Choose a camera lens or adjust the zoom
  (projection transformation).
- Determine how large you want the final photograph to be
  (viewport transformation).





Overview



TODO:

1. Switch matrix mode to GL_MODELVIEW and call glLoadIdentity().
2. Call gluLookAt().
3. Your own modeling transformations.

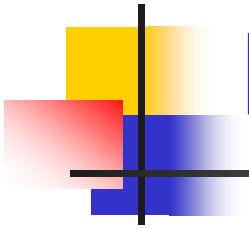
TODO:

1. Switch matrix mode to GL_PROJECTION and call glLoadIdentity().
2. Call gluPerspective() if you want perspective projection.
3. Call gluOrtho2D() if you want orthogonal projection.

TODO:

1. Call glViewport().

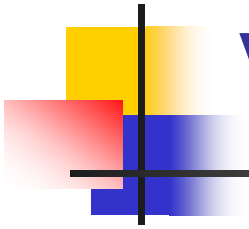
Matrix in OpenGL



- Consider a transformation ($T^1T^2…T^n$). To build the transformation matrix, we shall multiply identity matrix by T^1 then T^2…until T^n,
- The order of issuing commands shall be inversed.
  - Ex: rotate then translate

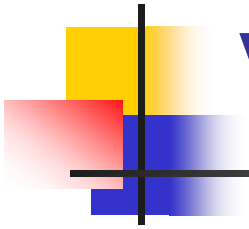
    glTranslatef(1,0,0);

    glRotatef(45.0, 0, 0, 1);



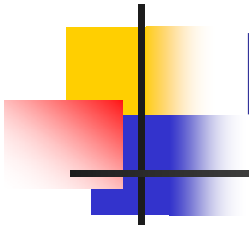
Viewing transformations 1/2

- gluLookAt(*GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdoubpe upz*);
  - eyex, eyey, eyez is where the camera is positioned.
  - centerx, centery, centerz is where the camera looks at.
  - Upx, upy, upz is the up-vector of the camera.
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glufnc01_8wtw.asp



Viewing transformations 2/2

- Use **gluLookAt()** to indicate where the camera is placed and aimed.
- If not called, the camera has a default position at the origin, points down the negative Z-axis, and an up-vector of positive Y-axis.

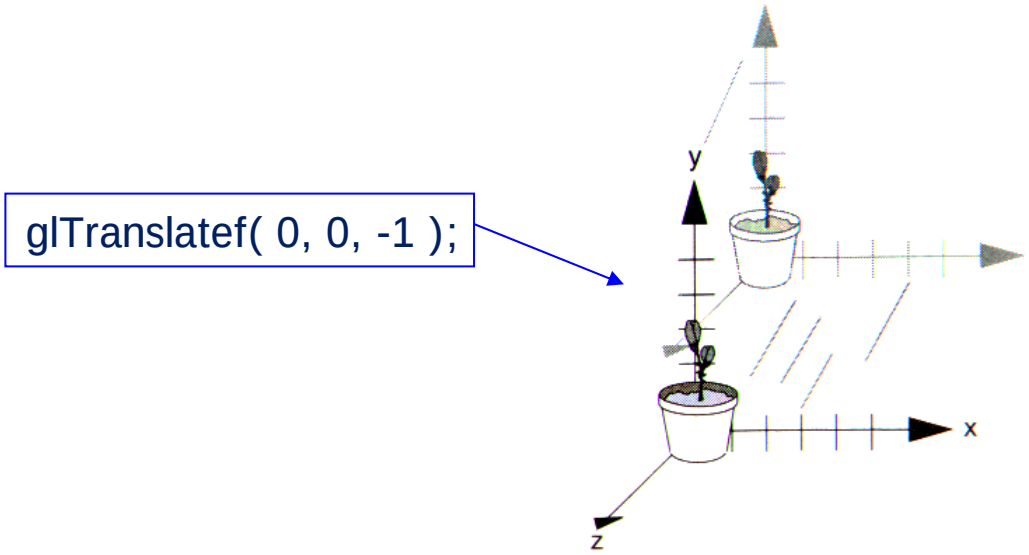


Modeling transformation 1/4

- Perform rotate, translate, scale and combinations of these transformations
- In OpenGL, modeling and viewing transformation are combined into the modelview matrix before the transformation are applied.

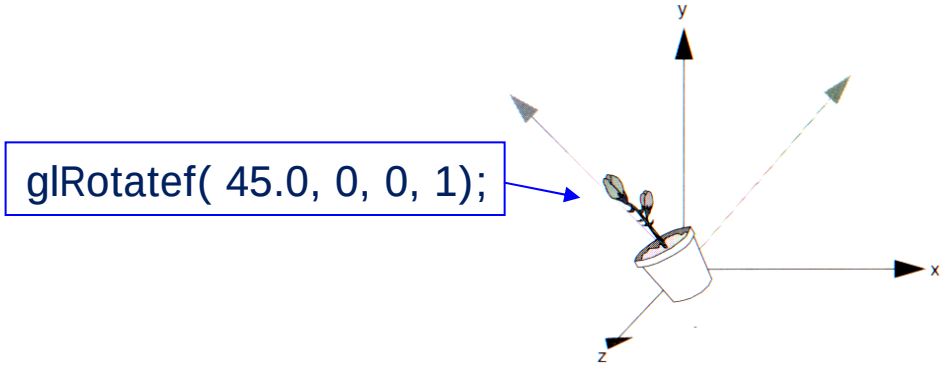
Modeling transformation 2/4

- glTranslate{fd}(*TYPE x, TYPE y, TYPE z*);
  - Multiplies current matrix by a matrix that moves an object by x,y,z
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc03_9a05.asp



Modeling transformation 3/4

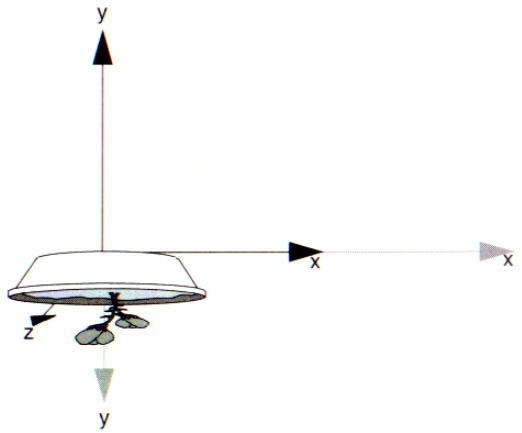
- glRotate{fd}(*TYPR angle, TYPE x, TYPR y, TYPE z*);
  - Multiplies current matrix by a matrix that rotates an object in a counterclockwise direction about the ray from origin to (x,y,z) with angle as the degrees.
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc03_21d1.asp

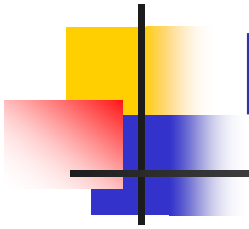


Modeling transformation 4/4

- glScale{fd}(*TYPE x, TYPE y, TYPE z*);
  - Multiplies current matrix by a matrix that scales an object along axes.
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc03_1691.asp

glScalef(2.0, -0.5, 1.0);



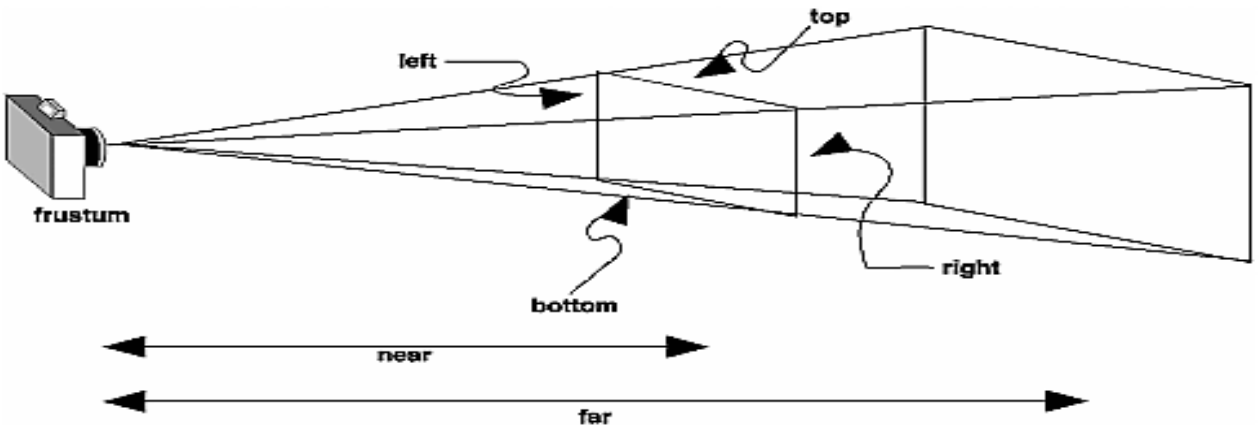


Projection transformation 1/5

- Determine what the field of view (or viewing volume) is and how objects are *projected* onto the screen.

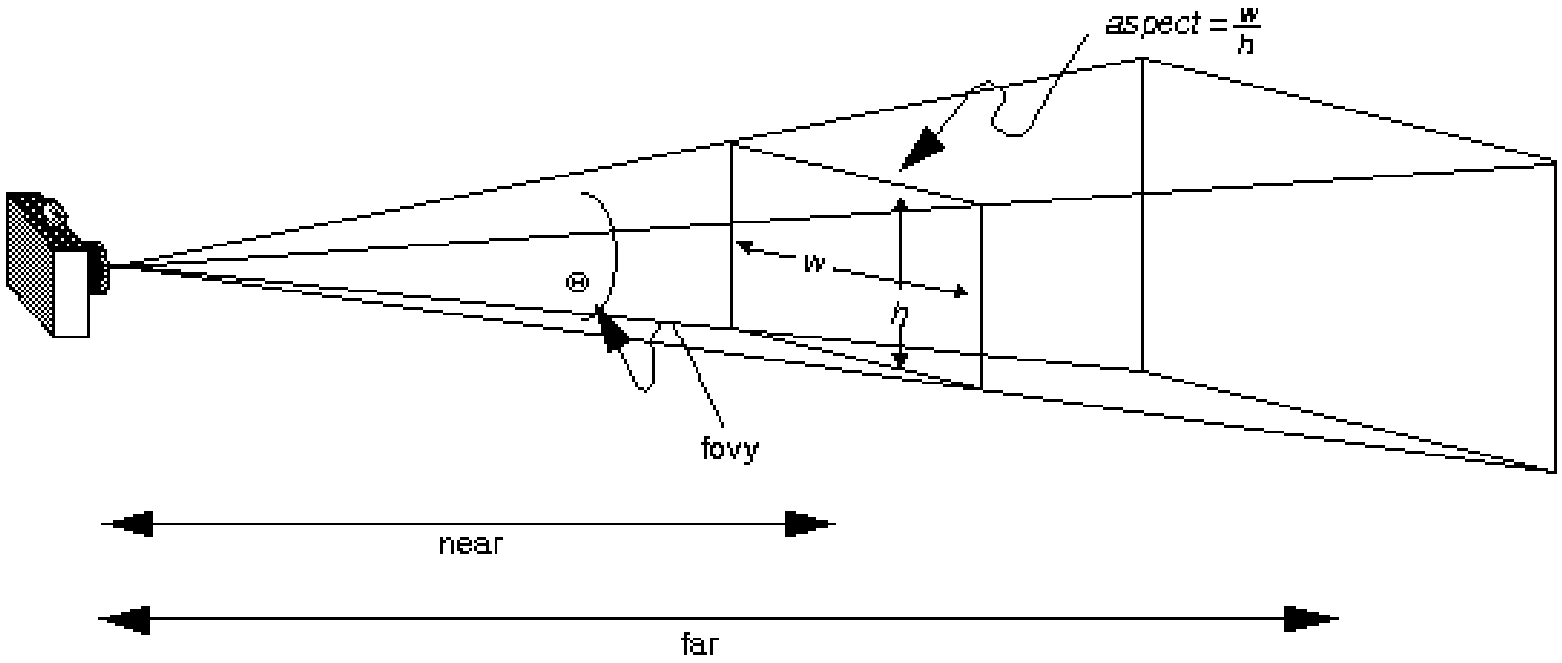
Projection transformation 2/5

- Perspective Projection
  - glFrustum(*GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far*);
    - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc02_0oj1.asp



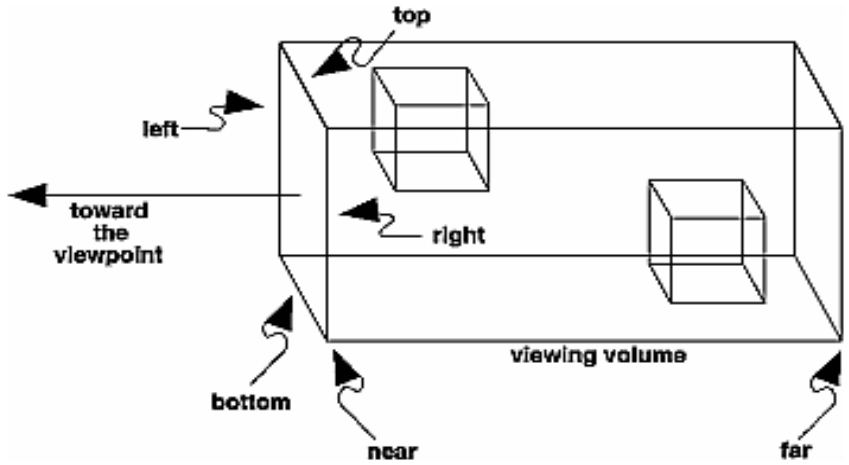
Projection transformation 3/5

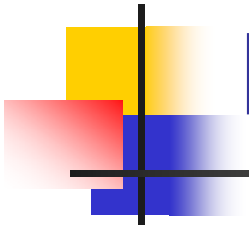
- gluPerspective(*GLdouble fovy, GLdouble aspect, GLdouble near, GLdouble far*);
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glufnc01_6m79.asp



Projection transformation 4/5

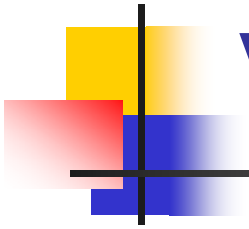
- Orthographic projection
  - glOrtho(*GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far*);
    - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc03_8qnj.asp





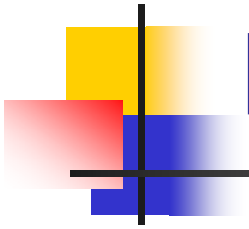
Projection transformation 5/5

- gluOrtho2D(*GLdouble left, GLdouble right, GLdouble bottom, GLdouble top*);
  - Equal to calling **glOrtho** with near = 1 and far = 1.
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glufnc01_5alg.asp



Viewport transformations

- void glViewport(*GLint x, GLint y, GLsizei width, GLsizei height*);
  - Transform the final image into some region of the window
  - *x, y* :The lower-left corner of the viewport rectangle, in pixels. The default is (0,0).
  - *width, height* :The width and height of the viewport. The default is set to the dimensions of that window
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc03_5ur8.asp



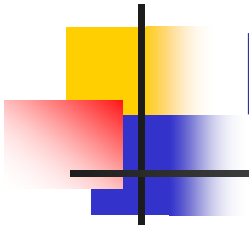
Matrix Manipulation 1/4

- void glMatrixMode(*GLenum mode*);
  - Switch between three modes
    - GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE
  - Each matrix mode has its own matrix stack.
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc03_4vs5.asp
- void glLoadIdentity();
  - Set current matrix to the 4x4 identity matrix
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc03_6h2x.asp

Matrix Manipulation 2/4

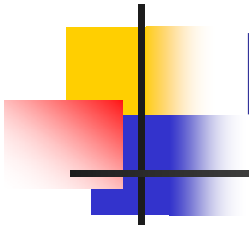
- glLoadMatrix{f,d}(*const TYPE* m*);
  - Replaces current matrix by a user defined matrix
  - The user defined matrix m is a 4x4 array
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc03_3260.asp

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



Matrix Manipulation 3/4

- glMultMatrix{f,d}(*const TYPE* m*);
  - Multiplies current matrix by a user defined matrix
  - The user defined matrix m is a 4x4 array.
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc03_0xmg.asp



Matrix Manipulation 4/4

- void glPushMatrix();
  - Push current matrix into matrix stack.
- Void glPopMatrix();
  - Pop matrix from matrix stack
  - These stack operations of matrices are very useful for constructing a hierarchical structure.
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/opengl/glfunc03_246w.asp