OpenGL Programming

Contents

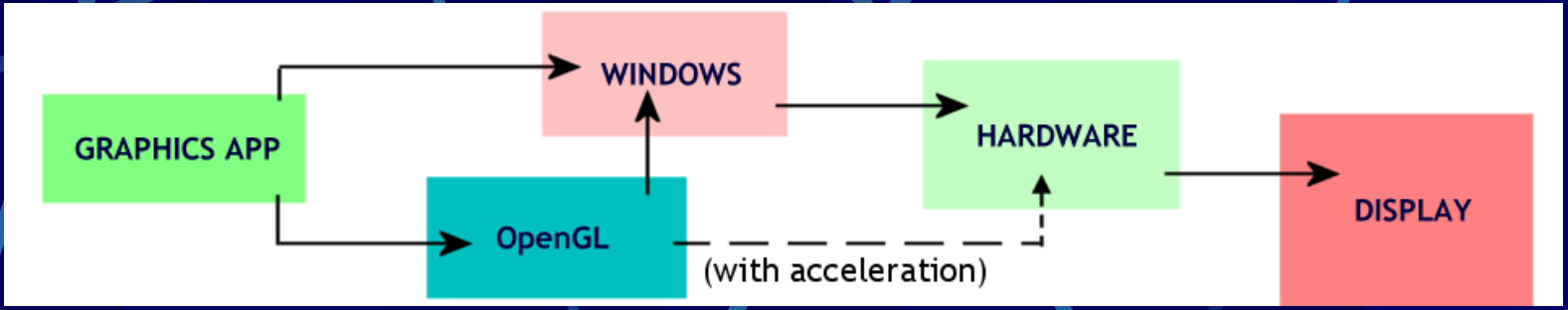
- What is OpenGL?
- Drawing geometry
- Coordinate systems
- OpenGL and windows integration (GLUT)
- GLUT code examples
- Graphics pipeline
- Graphics architecture
- Event processing
- Advanced features and GLU

What is OpenGL?

- Software interface to graphics hardware
- Low-level graphics rendering and imaging library
  - points, lines, polygons, rectangles
  - bitmaps and raster rectangles
- Hardware-independent interface
  - Microsoft Windows
  - Linux
  - Mac OS X

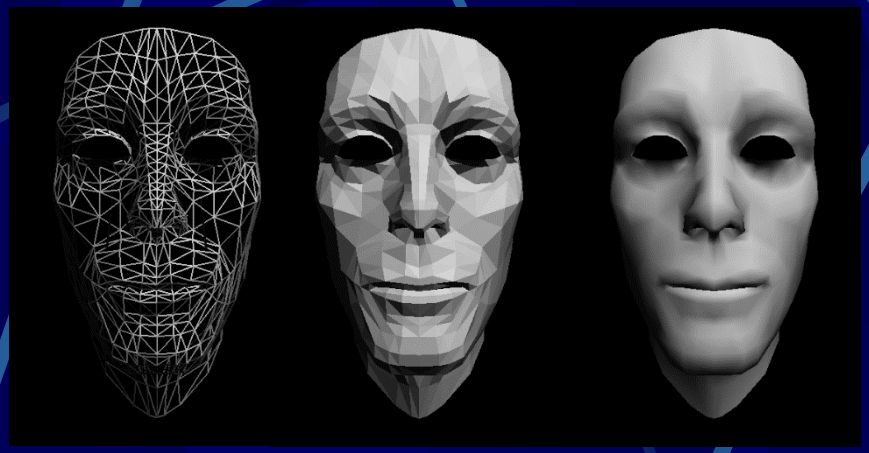
→ “no windowing task and no user input handling”

The OpenGL® API



http://www.opengl.org

OpenGL Examples



Simple Code Example

```
#include <whateverYouNeed.h>
main()
{
        InitializeAWindowPlease();

        glClearColor(0.0, 0.0, 0.0, 0.0);
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0, 1.0, 1.0);
        gluOrtho2D(0.0, 1.0, 0.0, 1.0);
        glBegin(GL_POLYGON);
                glVertex3f(0.25, 0.25, 0.0);
                glVertex3f(0.75, 0.25, 0.0);
                glVertex3f(0.75, 0.75, 0.0);
                glVertex3f(0.25, 0.75, 0.0);
        glEnd();
        glFlush();
        UpdateTheWindowAndCheckForEvents();
}
```

OpenGL Command Syntax

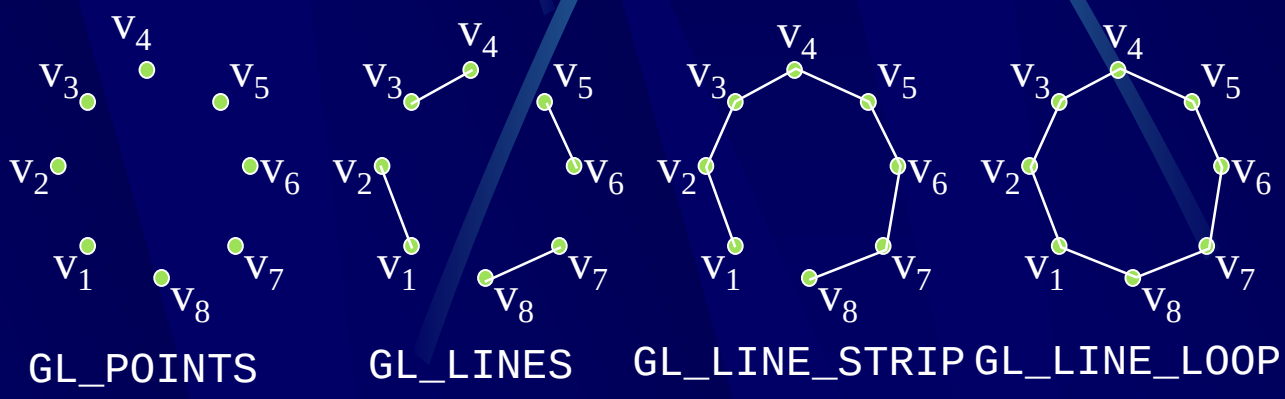
- glVertex{234}{sifd}[v](TYPE coords);
  - gl: prefix of OpenGL functions
  - GL_ : prefix of constant
  - {234}
    - number of components (2, 3, or 4)
  - {sifd}
    - s: signed short integer
    - i: signed integer
    - f: float
    - d: double
  - [v]
    - v indicates vector format, if present
  - Example: glVertex3fv(float coords[3]);

Drawing Geometry

- glBegin() and glEnd()
- glVertex*()
  - send down a vertex with current attributes
- Vertex attributes:
  - glColor*() / glIndex
  - glNormal*()
  - glMaterial*()
  - glTexCoord*()
  - ...
- Not all OpenGL commands are allowed between glBegin() and glEnd()

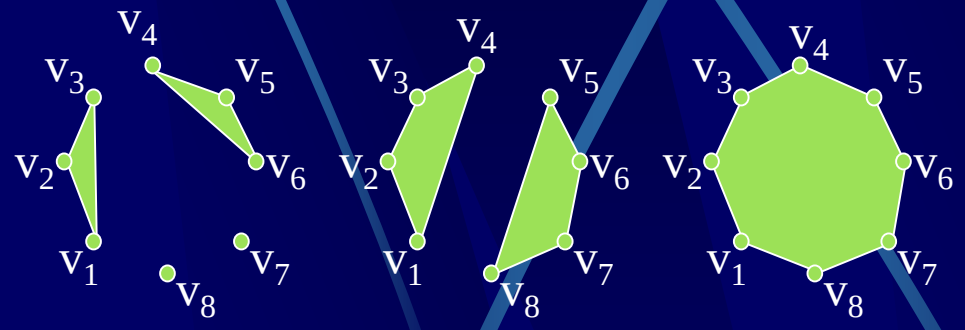
Drawing Geometry (2)

- All geometric objects in OpenGL are created from a set of basic primitives
- Certain primitives are provided to allow optimal geometry transmission for improved rendering speed
- glBegin (GLenum primitiveType)
- Line based primitives

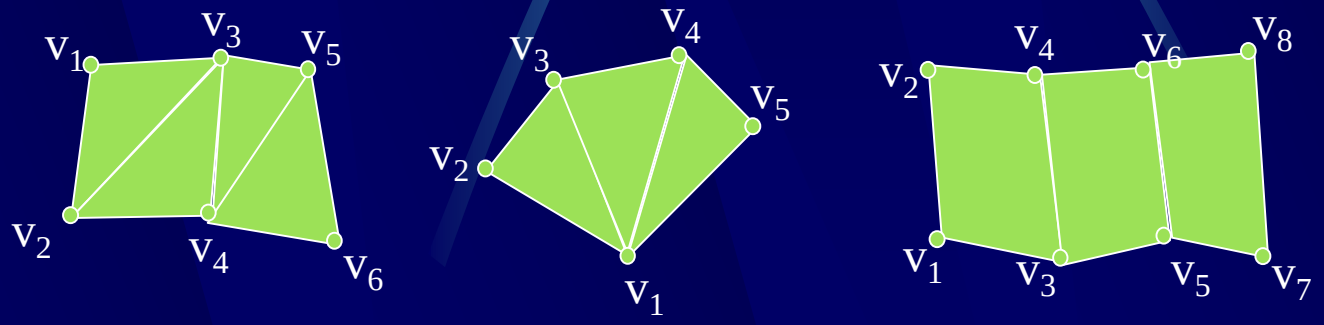


Drawing Geometry (3)

- Polygon primitives



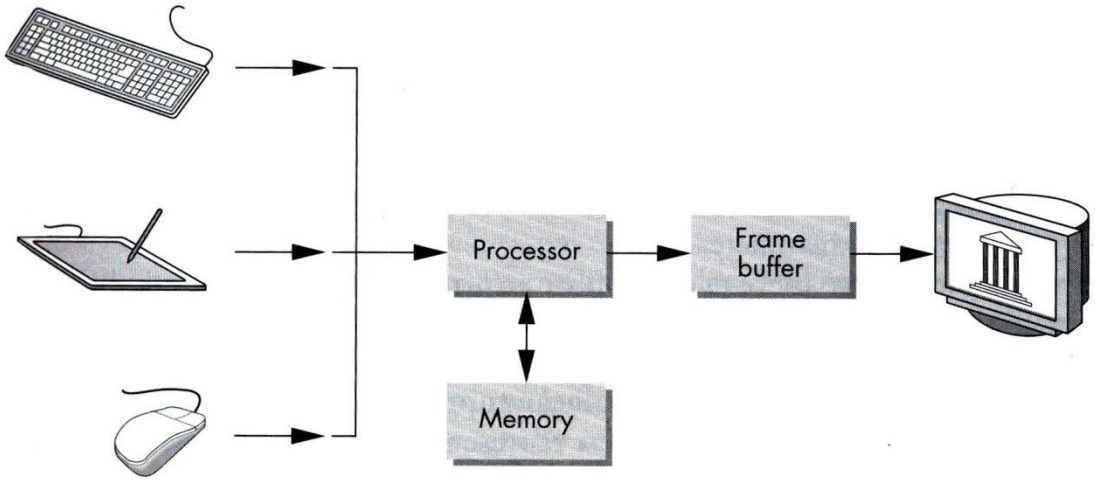
GL_TRIANGLES GL_QUADS GL_POLYGON

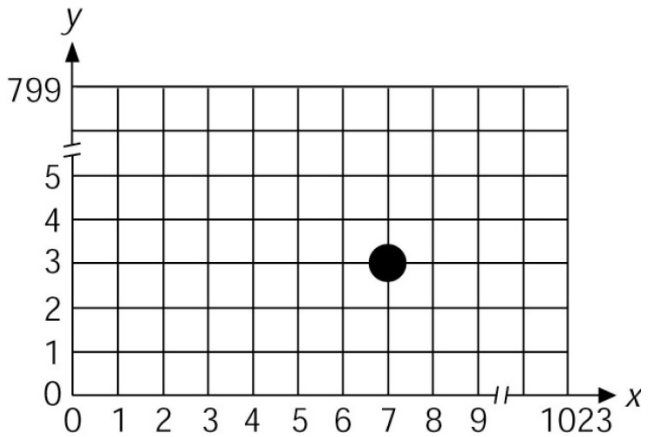


GL_TRIANGLE_STRIP GL_TRIANGLE_FAN GL_QUAD_STRIP

Coordinate Systems

- Coordinate system
  - origin and the coordinate axes
- Device coordinates
  - pixel coordinates dependent on display resolution

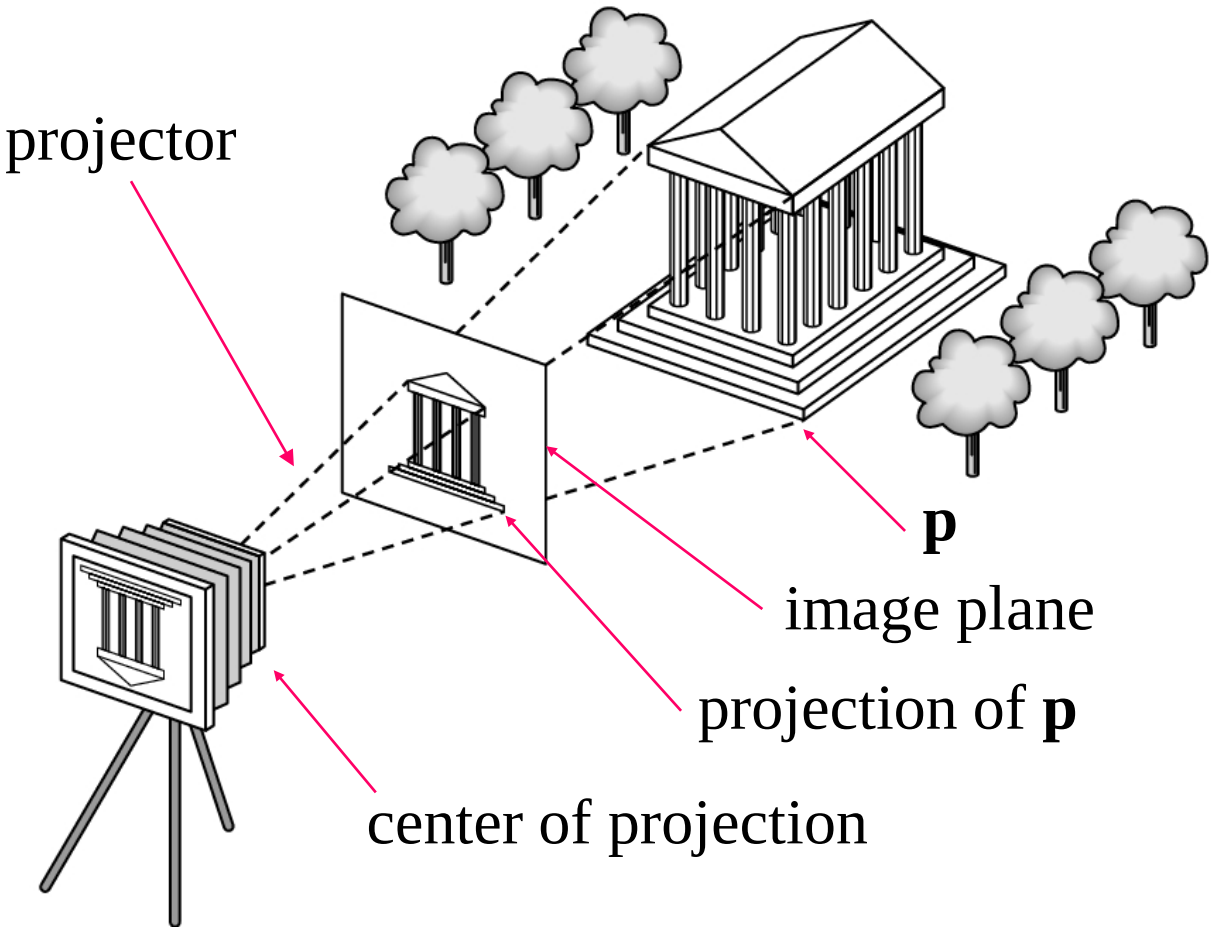




Simple 2D Game

- https://www.youtube.com/watch?v=3smytj9Bu_E
  - 19:20

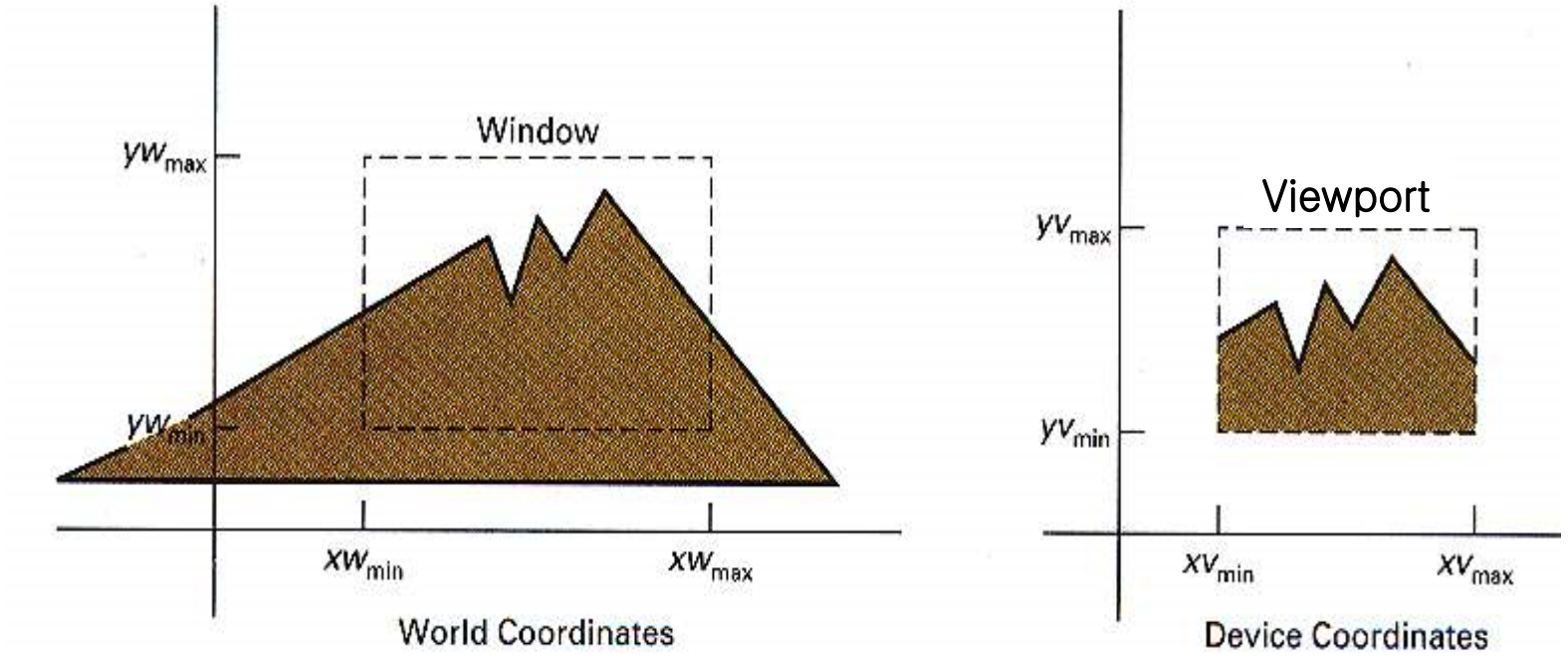
Synthetic Camera Model



Coordinate Systems (2)

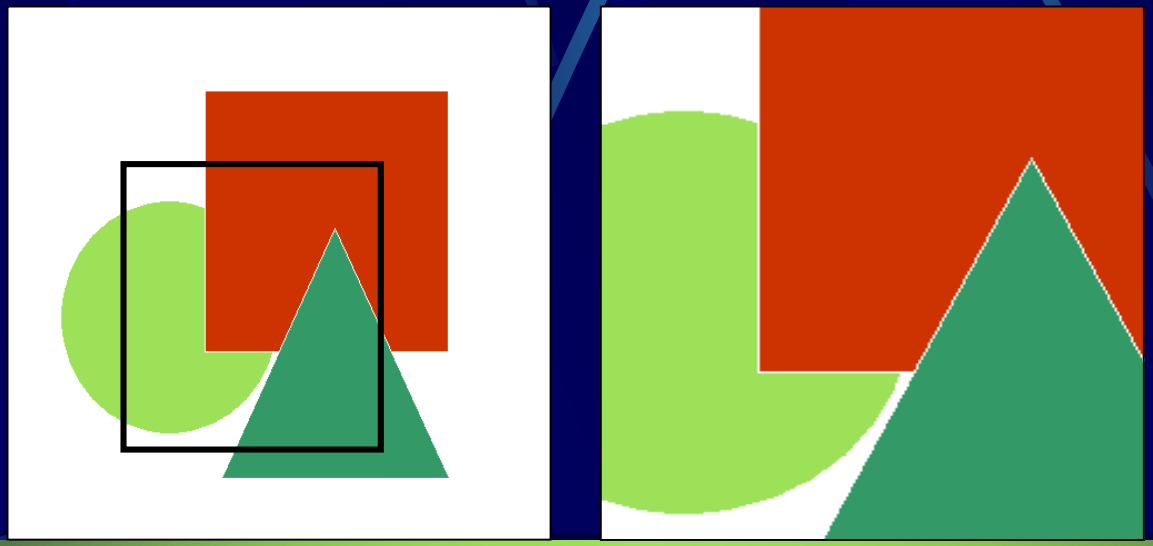
- (2D) World coordinates
  - imaginary coordinates for object representation
- (Clip) Window
  - usually a rectangular region
- Clipping
  - identifies those portions of a picture that are either inside or outside of a specified region (window)
- Device coordinates
  - pixel coordinates on a (hardware) display
- Viewport
  - area on a display to which a window is mapped

Coordinate Systems (3)



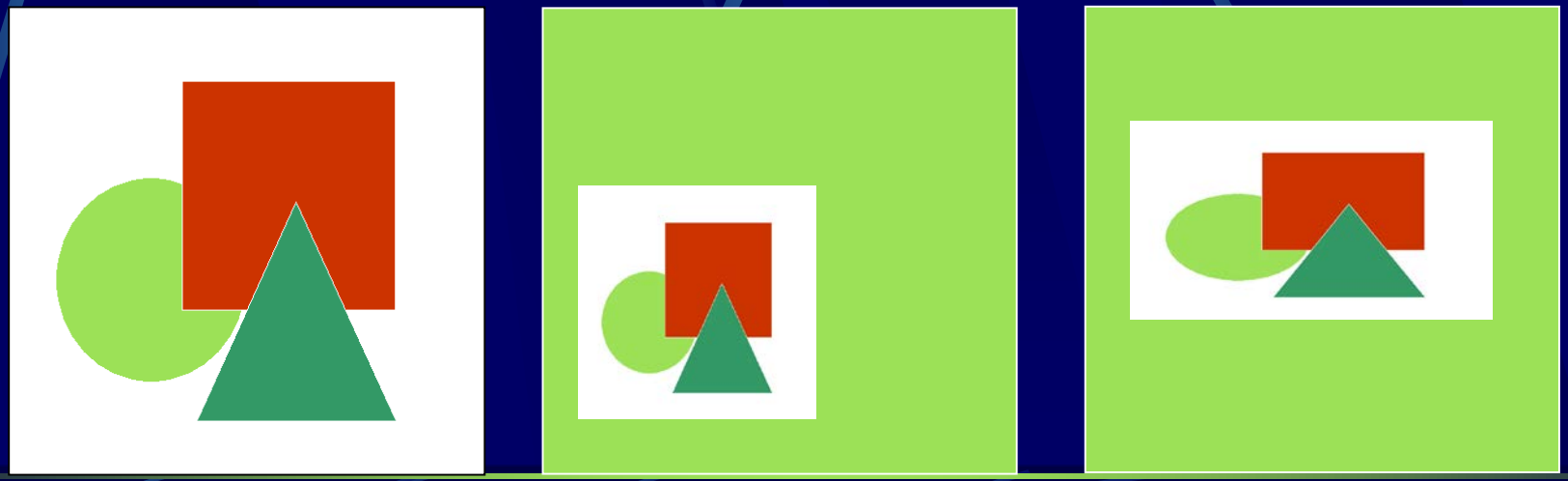
Coordinate System

- World coordinate system
  - imaginary coordinates for modeling the world
- Select a region that will be drawn
  - gluOrtho2D (left, right, bottom, top)



Viewport

- Device coordinate system
  - Photograph or screen
- Rectangular region of the screen window on which the image is drawn
  - glViewport (x, y, width, height)



OpenGL & Windows System Integration

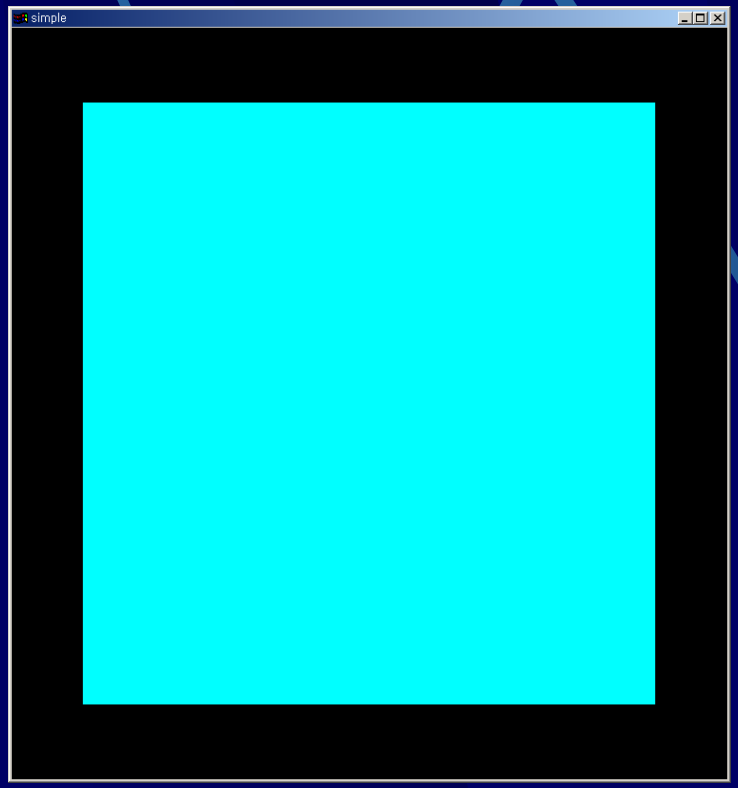
- Window system dependent
  - Microsoft Win32: WGL
- Window system independent
  - GLUT
  - GLFW
- Web-based graphics
  - WebGL

GLUT (OpenGL Utility Toolkit)

- A window system independent toolkit for writing OpenGL programs
- GLUT contains commands that
  - open windows
  - read events from the keyboard or mouse
  - create 3D objects such as a sphere, a torus, and a teapot

GLUT Code Example #1

- Draw a rectangle



GLUT Code Example #1 (2)

```
#include <gl/glut.h>    /* this includes the others */

void main (int argc, char **argv)
{
        glutInit (&argc, argv);
        glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
        glutCreateWindow (“simple”);
        glutReshapeFunc (myReshape);
        glutDisplayFunc (display);
        glutMainLoop();
}
```

GLUT Code Example #1 (3)

```
void myReshape (int w, int h) {
        glLoadIdentity ();
        glViewport (0, 0, w, h);
        gluOrtho2D (0.0, 100.0, 0.0, 100.0);
}
void display (void) {
        glClear (GL_COLOR_BUFFER_BIT);
        glColor3f (0.0, 1.0, 1.0);
        glRectf(10.0, 10.0, 90.0, 90.0);
        glutSwapBuffers ();
}
```

GLUT Code Example #2

- Animate a rectangle

GLUT Code Example #2 (2)

```
#include <windows.h>
#include <gl/gl.h>
#include <gl/glut.h>

int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow (argv[0]);
   init ();

   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutIdleFunc(moveObjects);
   glutMainLoop();
   return 0;
}
```

GLUT Code Example #2 (3)

```
typedef struct rect{
    float x;
    float y;
    float width;
    float height;
} rect;
rect    rectangle;

void init(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glShadeModel(GL_FLAT);

    rectangle.x = 0.1;
    rectangle.y = 0.1;
    rectangle.width = 0.1;
    rectangle.height = 0.15;
}
```

GLUT Code Example #2 (4)

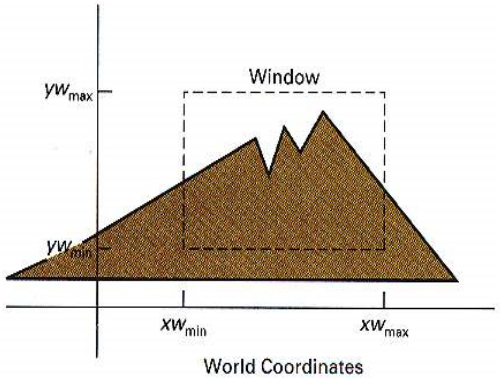
```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_LINE_LOOP);
        glVertex2f(rectangle.x, rectangle.y);
        glVertex2f(rectangle.x, rectangle.y + rectangle.width);
        glVertex2f(rectangle.x + rectangle.height, rectangle.y + rectangle.width);
        glVertex2f(rectangle.x + rectangle.height, rectangle.y);
    glEnd();

    glutSwapBuffers();
}
```

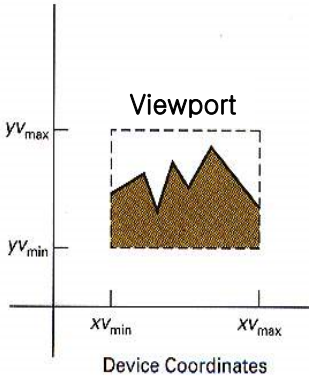
GLUT Code Example #2 (5)

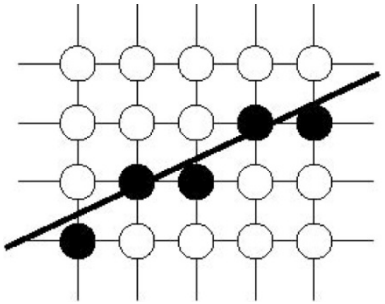
```
void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    gluOrtho2D(0, 1, 0, 1);
    glMatrixMode (GL_MODELVIEW);
    glLoadIdentity();
}

void moveObjects()
{
    rectangle.x += 0.001;
    rectangle.y += 0.001;

    glutPostRedisplay();
}
```

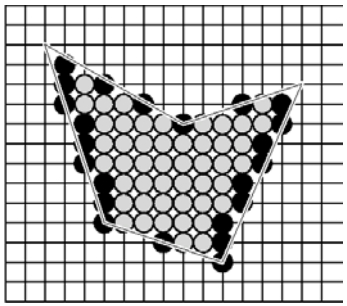
Viewing & Scan Conversion

- 2D viewing
  - viewing parameter specification
  - mapping of a part of a world-coordinate scene (window) to device coordinates, including clipping
- Scan conversion
  - rasterization
  - converts output primitives to a set of pixels



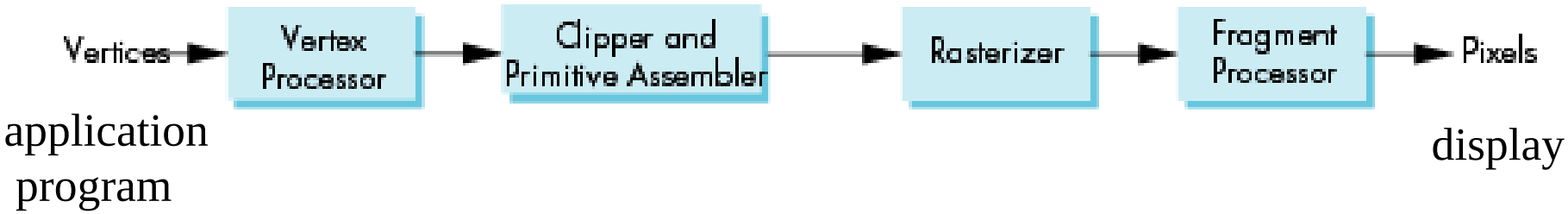






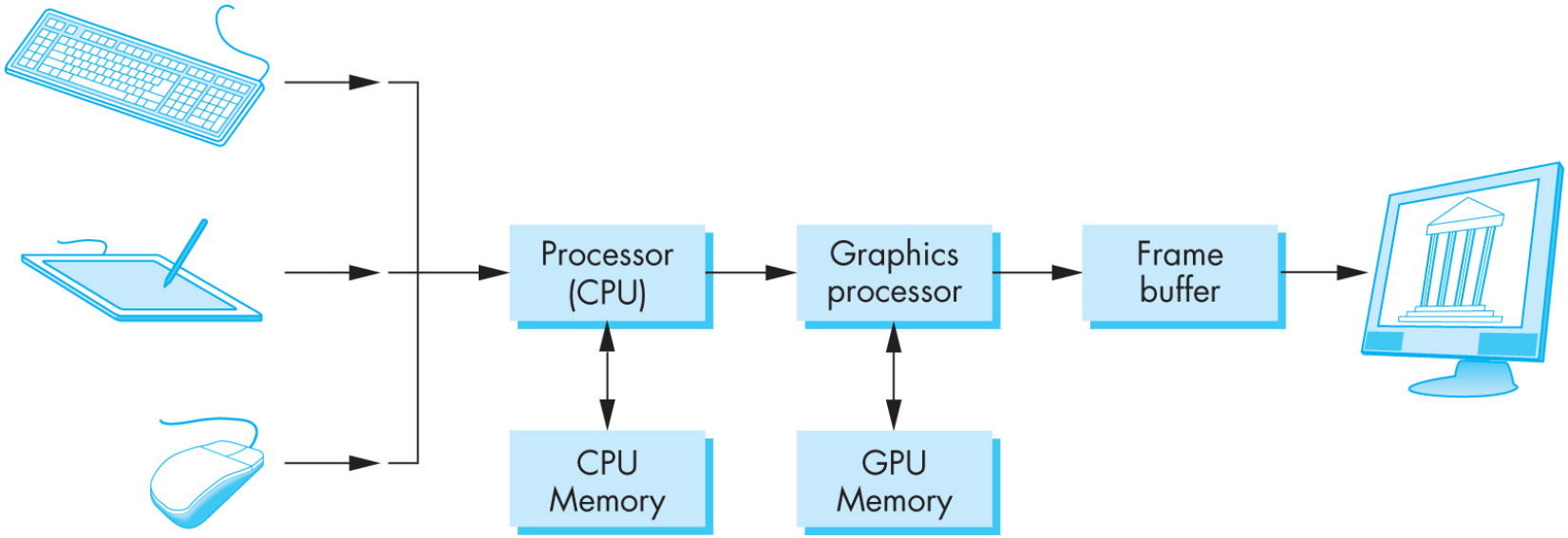
Graphics Pipeline

- Process objects one at a time in the order they are generated by the application
  - Can consider only local lighting
- Pipeline architecture



- All steps can be implemented in hardware on the graphics card

Graphics Architecture



Input devices

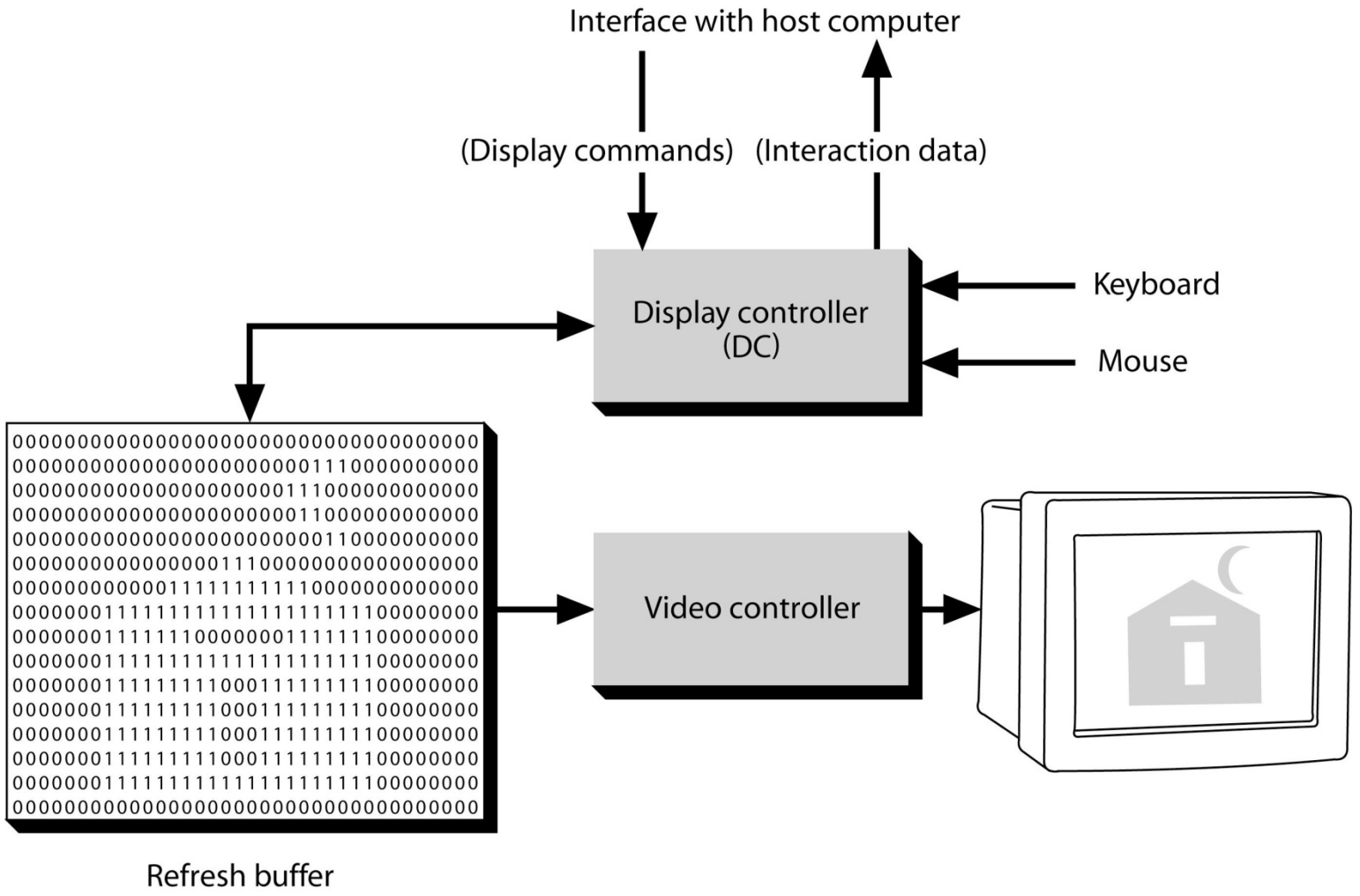
Output device

Image formed in frame buffer

Graphics Architecture (2)

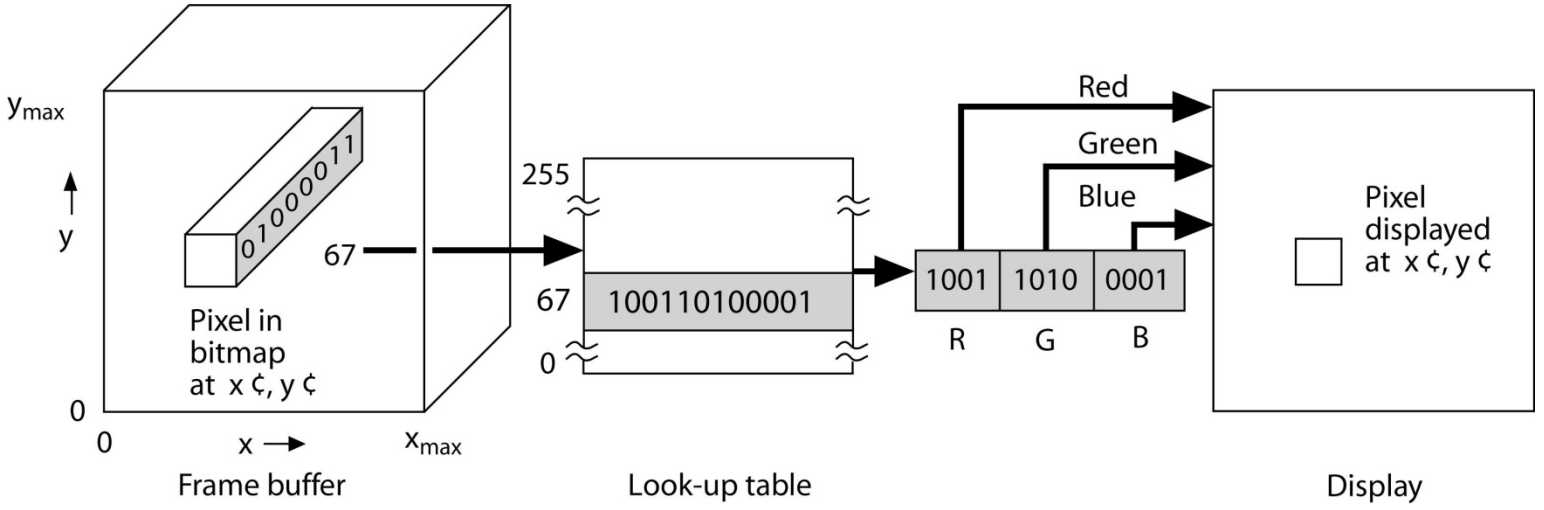
- Frame buffer (refresh buffer)
  - storage for pixel information
- Video controller
  - color display
  - screen refresh
- Display processor (controller)
  - viewing, scan conversion, transformations
  - high-level rendering operations

Graphics Architecture (3)



Graphics Architecture (4)

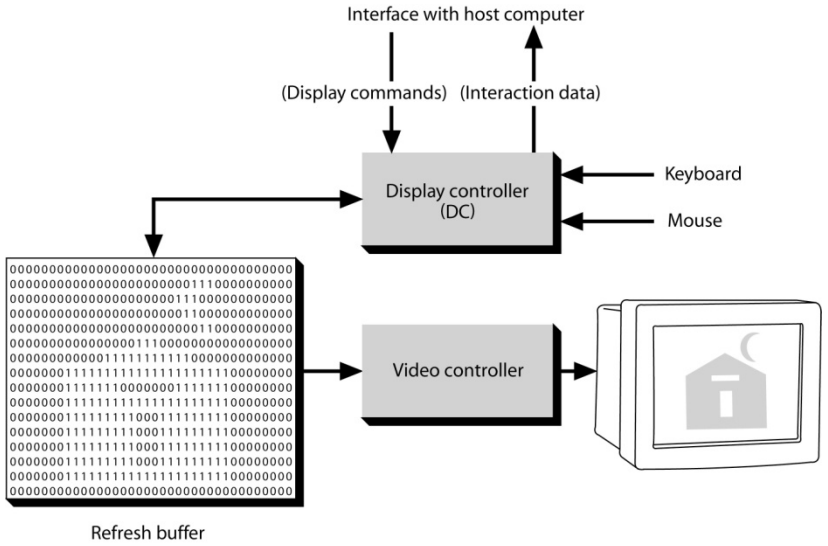
- Displaying colors
  - color table (look-up table, LUT)
    - has as many entries as there are pixel values
    - pixel value is used as an index into the color table
    - usually 8 bits per pixel, 256 entries in the table



Graphics Architecture (5)

- true color display
  - allocate bits for red, green, blue
  - 6, 6, 4 bits for RGB, or 8 bits for each of RGB

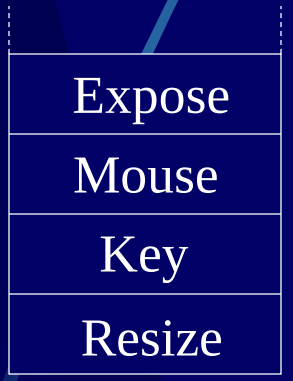
- Refreshing the display
  - display is refreshed at 60+ Hz with the frame buffer
  - double buffer for animation



Graphics Architecture (6)

- Sprites
  - multiple small, fixed-size pixmaps
  - superimposed on top of the frame buffer
  - used often in video games

OpenGL® GLUT Event Loop



Operating System

Application

glutMainLoop() =

```
while (TRUE)
e = get_next_event();
switch (e)
   case MOUSE:
       call registered MouseFunc
    case Resize
       call registered ResizeFunc
    ...
```

Beginning Event Processing

```
void glutMainLoop(void);
```

- glutMainLoop enters the GLUT event processing loop
- should be called only once in a GLUT program, never returns
- performs registered callbacks when necessary

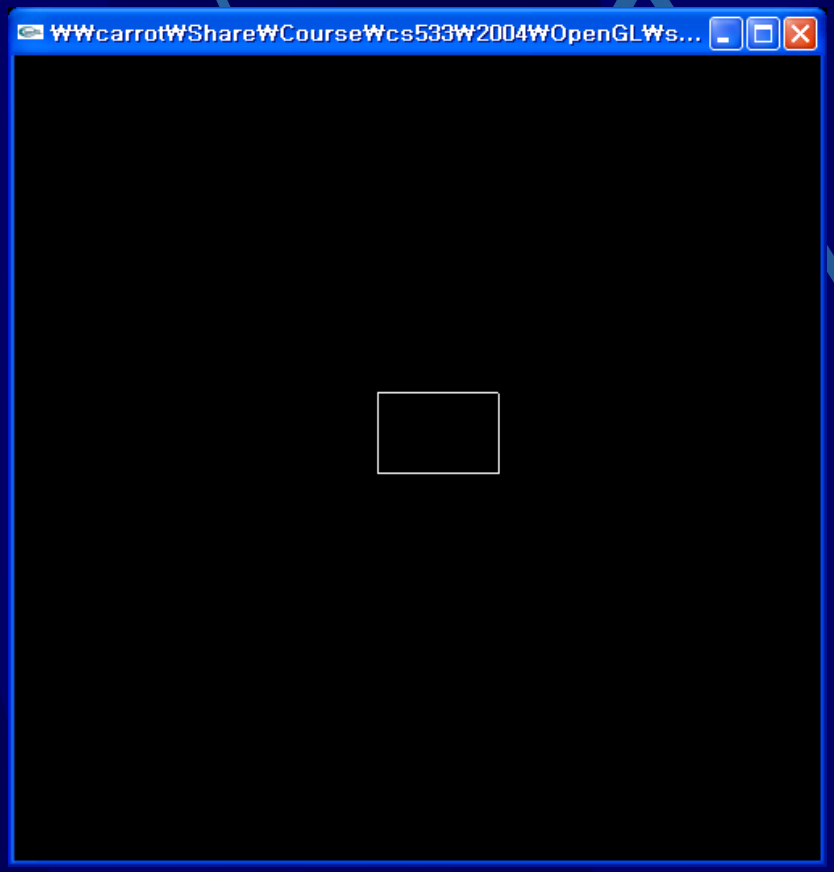
```
While (TRUE)
{
  e=getNextEvent();
  switch (e)
  {
     case (MOUSE_EVENT):
        call registered MouseFunc
        break;
     case (RESIZE_EVENT):
        call registered ReshapeFunc
        break;
        …
    }
```

Call Back Registration

- `void glutDisplayFunc(void (*func) (void));`
- `void glutReshapeFunc(void (*func) (int width, int height));`
- `void glutKeyboardFunc(void (*func) (unsigned char key, int x, int y));`
- `void glutMouseFunc(void (*func) (int button, int state, int x, int y));`
- `void glutIdleFunc(void (*func) (void));`
- `void glutTimerFunc(unsigned int msecs, void (*func) (int value), value);`

GLUT Code Example #3

- Move a rectangle with keyboard input



GLUT Code Example #3 (2)

```
#include <windows.h>
#include <gl/gl.h>
#include <gl/glut.h>

int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow (argv[0]);
   init ();

   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutKeyboardFunc(keyboard);
   glutSpecialFunc(specialkeyboard);
   glutMainLoop();
   return 0;
}
```

GLUT Code Example #3 (3)

```
typedef struct rect{
    float x;
    float y;
    float width;
    float height;
} rect;
rect    rectangle;

void init(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glShadeModel(GL_FLAT);

    rectangle.x = 0.45;
    rectangle.y = 0.48;
    rectangle.width = 0.1;
    rectangle.height = 0.15;
}
```

GLUT Code Example #3 (4)

```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_LINE_LOOP);
        glVertex2f(rectangle.x, rectangle.y);
        glVertex2f(rectangle.x, rectangle.y + rectangle.width);
        glVertex2f(rectangle.x + rectangle.height, rectangle.y + rectangle.width);
        glVertex2f(rectangle.x + rectangle.height, rectangle.y);
    glEnd();

    glutSwapBuffers();
}
```

GLUT Code Example #3 (5)

```
void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    gluOrtho2D(0, 1, 0, 1);
    glMatrixMode (GL_MODELVIEW);
    glLoadIdentity();
}
```

GLUT Code Example #3 (6)

```
void keyboard(unsigned char key, int x, int y)
{
   switch (key) {
      case 'i':
                rectangle.y += 0.005;
               break;
      case 'm':
                 rectangle.y -= 0.005;
               break;
      case 'k':
              rectangle.x += 0.005;
               break;
      case 'j':
                 rectangle.x -= 0.005;
               break;
   }
   glutPostRedisplay();
}
```

GLUT Code Example #3 (7)

```
void specialkeyboard(int key, int x, int y)
{
   switch (key) {
      case GLUT_KEY_UP:
                           rectangle.y += 0.005;
                           break;
      case GLUT_KEY_DOWN:
                           rectangle.y -= 0.005;
                           break;
      case GLUT_KEY_RIGHT:
                            rectangle.x += 0.005;
                           break;
      case GLUT_KEY_LEFT:
                           rectangle.x -= 0.005;
                           break;
   }
   glutPostRedisplay();
}
```

Advanced Features of OpenGL

- Lighting and shading
- Antialiasing
- Drawing pixels, bitmaps, fonts, and images
- Texture mapping
- Atmospheric effects
  - fog, haze, smoke, and smog
- NURBS (Non-Uniform Rational B-Splines)

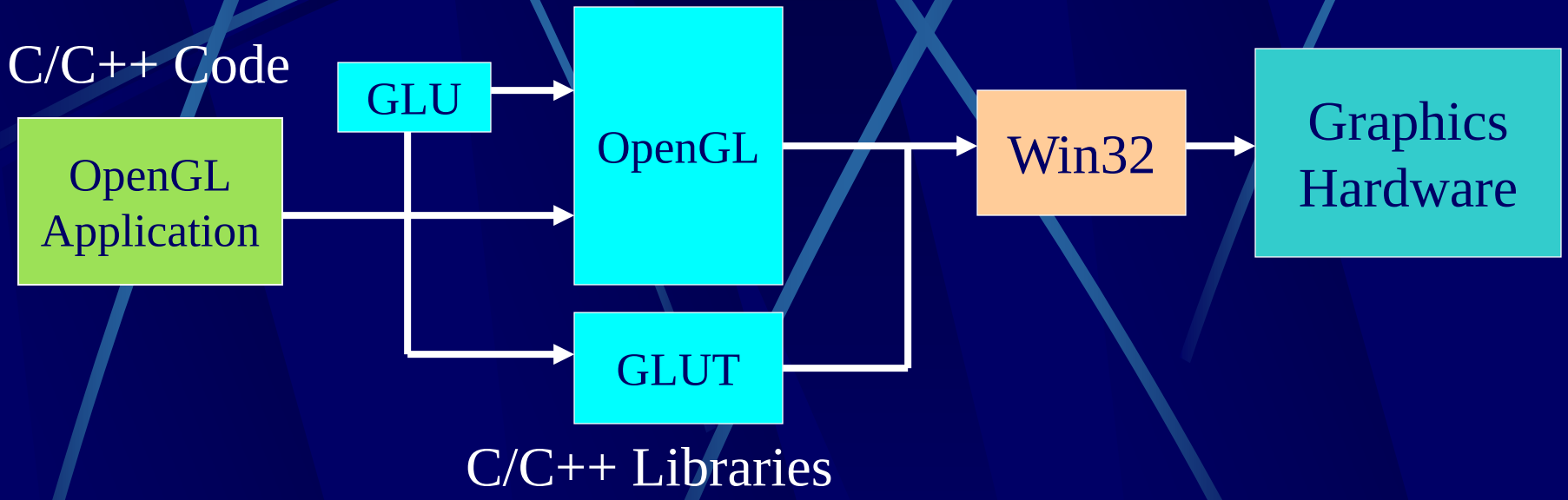
State Management

- OpenGL is a state machine
  - line and polygon stipple patterns, polygon drawing modes, material properties of the objects, ….
- Turn on and off OpenGL states
  - glEnable() and glDisable()
- Querying States
  - glIsEnabled(), glGetBooleanv(), glGetIntegerv(),…
  - glGetLight*(), glGetPolygonStipple(),…
- Save a collection of state variables
  - glPushAttrib() or glPopAttrib()

GLU (OpenGL Utility Library)

- A set of commonly used graphics routines
  - built on top of OpenGL
  - gluOrtho2D(), gluScaleImage(), gluLookAt(),…
- Polygonal surface routines
- Quadric surface routines
  - spheres, cones, open cylinders, and tesselated disks for circles and arcs
  - gluNewQuadric(), gluCylinder(), gluPartialDisk(),…
- NURBS (with trimming)
- Matrix and mipmap utilities

Library Relationship



- OpenGL has no windowing functions of its own
- We need to use something like the GLUT library for windowing operations etc.

Library Relationship (2)

- The **GL** library
  - core functions of OpenGL
  - modeling, viewing, clipping, lighting, ...
- The GL Utility **(GLU)** library
  - creation of common objects (spheres, quadrics, ...)
  - specification of standard views (perspective, ...)
- The GL Utility Toolkit **(GLUT)**
  - provides the interface with the windowing system
  - window management, menus, mouse interaction

Supplementary Slides

GLUT design philosophy

- GLUT requires very few routines to display a graphics scene rendered using OpenGL
- Like OpenGL the GLUT API is “stateful” and most initial states are predefined with defaults that are reasonable for simple programs
- The API is as much as possible windows system independent
- GLUT routines are logically organized into several sub-APIs according to functionality

GLUT sub-APIs

- Initialization
  - Command line processing, window system initialization initial window state
- Beginning event processing
  - Enter event processing loop
- Window management
- Overlay management
- Menu management

GLUT sub-APIs

- Callback registration
  - Registers procedures which will be called by GLUT event processing loop
- Color Index ColorMap management
- State retrieval
- Font rendering
- Geometric shape rendering

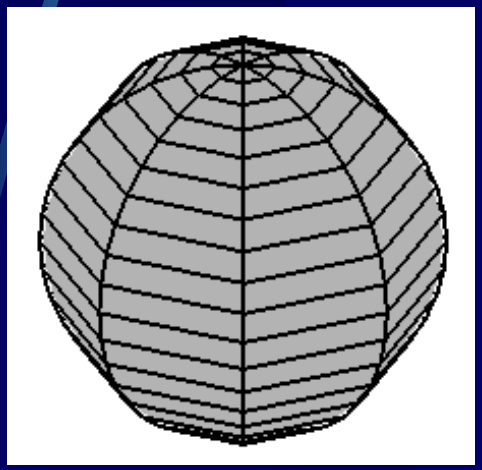
Initialization

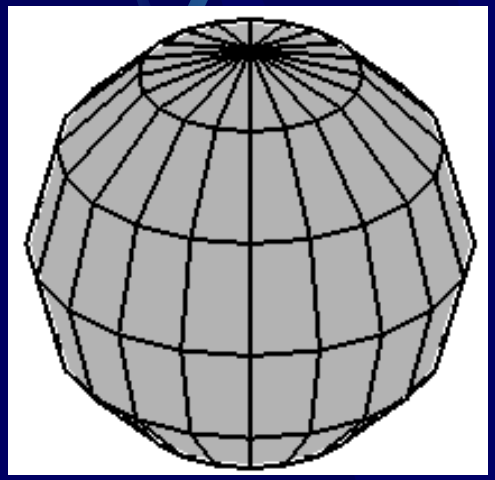
```
void glutInit(int* argcp, char** argv);
void glutInitWindowSize(int width, int
height);
void glutInitWindowPosition(int y, int y);
void glutInitDisplayMode(unsigned int mode);
```

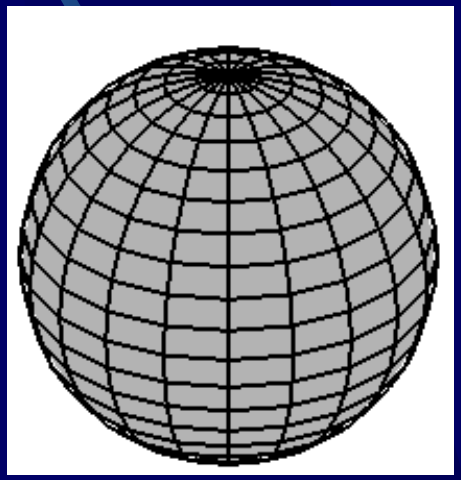
- glutInit initializes the GLUT library
- Display modes:
  - GLUT_RGBA, GLUT_RGB, GLUT_INDEX, GLUT_SINGLE, GLUT_DOUBLE, GLUT_ACCUM, GLUT_ALPHA, GLUT_DEPTH, GLUT_STENCIL, GLUT_MULTISAMPLE, GLUT_STEREO

Geometric Object Rendering

- `void glutSolidSphere(GLdouble radius, GLint slices, GLint stacks);`
- `void glutWireSphere(GLdouble radius, GLint slices, GLint stacks);`
- `void glutSolidCube(GLdouble size);`
- `void glutWireCube(GLdouble size);`







8, 20 20, 6 20, 20