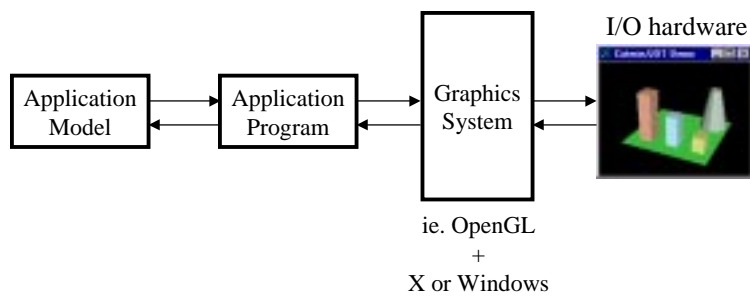


Handouts: Raster Graphics Hardware

Building Interactive Applications using OpenGL and GLUT

- Today: GLUT
- Friday: OpenGL

Conceptual Application Model

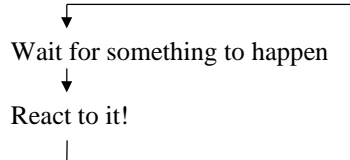


Handouts: Raster Graphics Hardware

Application Control Flow



- Interactive programs are *event driven*



- What are *events*?
 - User input
 - Window system, application generated

OpenGL



- 3D graphics library
 - Output only: render graphics
 - Only knows about "graphics contexts"
- Platform independent: No support for
 - Window creation
 - GLX
 - GLW
 - Input

Handouts: Raster Graphics Hardware

GLUT: OpenGL Utility Toolkit



- Supports OS independent OpenGL programs:
 - Multiple windows for OpenGL rendering.
 - Callback driven event processing.
 - Sophisticated input devices.
 - An "idle" routine and timers.
 - A simple, cascading pop-up menu facility.
 - Utility routines to generate various objects.
 - Support for bitmap and stroke fonts.
 - Miscellaneous window management functions.

Example Program:



```
#include "glut.h"

/* GLUT display callback function */
void display (void) { ... }

/* GLUT keyboard input callback function */
void keyin(unsigned char key, int x, int y) { ... }

/* main()
 * Initialize everything (window, color table, callbacks) and then
 * call the GLUT main loop to display the window and handle any events.
 */
int main (int argc, char **argv){ ... }
```

Handouts: Raster Graphics Hardware

GLUT display callback



```
/* window limits */
#define LEFT 17
#define RIGHT 66

void display (void) {
    GLint i;
    /* clear the window */
    glClear (GL_COLOR_BUFFER_BIT);
    /* draw the lines, changing the color for each */
    for (i = LEFT; i <= RIGHT; i++) {
        glIndexi (i);
        glBegin (GL_LINES);
        /* specify start and end vertex for each line */
        glVertex2f ((GLfloat)i, 0.0);
        glVertex2f ((GLfloat)LEFT, 1.0);
        glEnd ();
    }
    /* update the buffer immediately */
    glFlush ();
} /* end of display() */
```

Keyboard input callback



```
/* callback to handle keypress...exit on any key */

void keyin(unsigned char key, int x, int y){
    exit(0);
} /* end of keyin() */
```

Handouts: Raster Graphics Hardware

Main body of a GLUT program



```
/* Initialize everything (window, color table, callbacks) and then
 * call the GLUT main loop to display the window and handle any events. */

int main (int argc, char **argv){
    GLint i;
    GLfloat CurrBlue = 0.0;

    /* set up window size, position, and drawing coordinates */
    ... next slide ...

    /* set the display & keyboard callbacks and start the GLUT main loop */
    glutDisplayFunc (display);
    glutKeyboardFunc(keyin);
    glutMainLoop();
    /* include the return to keep lint happy */
    return (0);
} /* end of main() */
```

Initialization part of main()



```
glutInit (&argc, argv);
glutInitDisplayMode (GLUT_INDEX | GLUT_SINGLE);
glutInitWindowSize (500, 500);
glutInitWindowPosition (10,30);
glutCreateWindow ("Sample");
gluOrtho2D ((GLdouble)LEFT, (GLdouble)RIGHT, 0.0, 1.0);

/* set color map entries from 16 to 65 to increasingly brighter shades of
 * blue (first 16 colors are the ones often reserved by the windowing system
 * so don't mess with them */
for (i = LEFT; i <= RIGHT; i++) {
    glutSetColor (i, 0.0, 0.0, CurrBlue);
    CurrBlue += 0.02;
}
```

Handouts: Raster Graphics Hardware

Other GLUT commands



```
glutMouseFunc (void (*func) (int button, int state, int x, int y));
glutMotionFunc (void (*func) (int x, int y));

void mouseClick (int button, int state, int x, int y) { ... }
void mouseMove (int x, int y) { ... }

main () {
    ...
    glutMouseFunc (mouseClick);
    ...
}

button: GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON
state : GLUT_UP, GLUT_DOWN
```

Assignment #1



- Hand out Friday, due Friday Sept. 3rd
- Purpose:
 - learn how to write GLUT/OpenGL programs