

## Graphics Programming

### Input and Interaction

Interaction

Hofstra University

1

## Interaction

- Early 60's, Ivan Sutherland's **Project Sketchpad**
- **Basic Paradigm**: User see an image, reacts with an interactive device, image changes in response to input, . . .

Interaction

Hofstra University

2

## Interaction

- OpenGL does not support interaction directly
- Increase portability – work in a variety of environments
- Windowing and input functions left out of API – emphasis on rendering
- Use toolkits, GLUT

Interaction

Hofstra University

3

## Physical Input Devices

- **Pointing Device** – indicates position on screen
- **Keyboard Device** – return character codes to a program

Interaction

Hofstra University

4

## Logical Input Devices

Major Characteristics:

- **What** measurements the device returns to the program
- **When** the device returns those measurements

Interaction

Hofstra University

5

## Application Input

- **Measure** - what the device returns to the user program
- **Trigger** – signal send to the computer
- **Three distinct modes** defined by the relationship between the measure process and the trigger.
- **Request Mode, Sample Mode, Event Mode**

Interaction

Hofstra University

6

## Request Mode



- The measure of the device is not returned to the program until the device is triggered

## Sample Mode



- Measure is returned immediately after the function is called in the user program.
- No trigger needed
- Need to identify which device provides input
- Useful in apps where the program guides the user (e.g., simulators)

## Event Mode



- Can handle multiple inputs
- When device triggered an **event** is generated
- Identifier for device placed in the **event queue**
- Event queue process is independent of the application, asynchronous
- A **callback function** with a specific type of event

## Clients and Servers

- The building blocks of a distributed world are entities called **servers** that can perform tasks for **clients**.
- **Graphics server** provides input and output of graphical services
- OpenGL programs are **clients** of the graphics server
- X-Window system was one of the most successful

## Event-Driven Programming

- **Callback functions** determine how the application program responds to events
- *Examples, each of the events below needs a callback function to handle it.*  
 Pointing Device  
 Window Events  
 Keyboard Events  
 Display & Idle Callbacks  
 Window Management

## Pointing Device

- Almost always a mouse
- **Move event** – when mouse is moved with button depressed;
- **Passive Move Event** – move without pressing button
- **Mouse Event** – mouse button is depressed or released

`glutMouseFunc (mouse);`

## Pointing Device – Mouse Callback

```
void mouse(int btn, int state, int x, int y)
{
    if(btn==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
        exit(0);
}
```

Depressing the right Button terminates the program

Interaction

Hofstra University

13

## Pointing Device – Reshape Event

```
int main(int argc, char** argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutCreateWindow("square");
    myinit ();
    glutReshapeFunc (myReshape);
    glutMouseFunc (mouse);
    glutMotionFunc(drawSquare);
    glutDisplayFunc(display);

    glutMainLoop();
}
```

Called whenever the window is resized

Interaction

Hofstra University

14

## Window Events (Resizing – Dragging)

- Redraw all the objects?
- How do we handle aspect ratio?
- Change size of attributes and primitives?

Interaction

Hofstra University

15

## Square Program

- This program illustrates the use of the glut library for interfacing with a Window System
- The program opens a window, clears it to black, then draws a box at the location of the mouse each time the left button is clicked. The right button exits the program
- The program also reacts correctly when the window is moved or resized by clearing the new window to black

Interaction

Hofstra University

16

## Global Variables

```
/* globals */
GLsizei wh = 500, ww = 500; /* initial window size */
GLfloat size = 3.0; /* half side length of square */
```

- Size of window
- Viewport position and size
- Size of clipping window

Interaction

Hofstra University

17

## Window Event - Reshape

```
void myReshape(GLsizei w, GLsizei h)
{
    /* adjust clipping box */
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, (GLdouble)w, 0.0, (GLdouble)h, -1.0, 1.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    /* adjust viewport and clear */
    glViewport(0,0,w,h);
    glClearColor (0.0, 0.0, 0.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glFlush();

    /* set global size for use by drawing routine */
    ww = w;
    wh = h;
}
```

reshaping routine called whenever window is resized or moved

Interaction

Hofstra University

18

## Pointing Device – Motion Callback

```

void drawSquare(int x, int y)
{
    y=wh-y;
    glColor3ub( (char) rand()%256, (char) rand()%256,
               (char) rand()%256);
    glBegin(GL_POLYGON);
        glVertex2f(x+size, y+size);
        glVertex2f(x-size, y+size);
        glVertex2f(x-size, y-size);
        glVertex2f(x+size, y-size);
    glEnd();
    glFlush();
}
    
```

*Called by  
glutMotionFunc(drawSquare):  
if button held down*

*Origin is top left  
of window system*

Interaction Hofstra University 19

## Keyboard Events

```

void keyboard(unsigned char key, int x, int y)
{
    if (key=='q' || key=='Q') exit(0);
}

glutKeyboardFunc(keyboard);
    
```

Interaction Hofstra University 20

## Window Management


```

id=glutCreateWindow("second window");

glutSetWindow(id);
    
```

Interaction Hofstra University 21

## Menus

- Pop-up menu 
  - Common Steps:
    - Define the entries
    - Link the menu to a mouse button
    - Define callback function
- Interaction Hofstra University 22

## Menus

```

glutCreateMenu(demo_menu);
glutAddMenuEntry("quit", 1);
glutAddMenuEntry("increase square size", 2);
glutAddMenuEntry("decrease square size", 3);
glutAttachMenu(GLUT_RIGHT_BUTTON);
    
```

*identifier passed  
to callback*

Interaction Hofstra University 23

## Menus

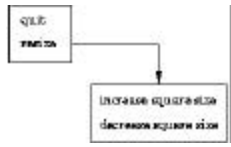
```

void demo_menu(int id)
{
    if (id==1) exit (0);
    else if (id==2) size = 2* size;
    else if (size > 1) size = size/2;
    glutPostRedisplay();
}

requests redraw through the  
glutDisplayFunc (redrawn  
w/o menu)
    
```

Interaction Hofstra University 24

## Hierarchical Menus



## Hierarchical Menus

```
sub_menu = glutCreateMenu(size_menu);
glutAddMenuEntry("increase square size", 2);
glutAddMenuEntry("decrease square size", 3);
glutCreateMenu(top_menu);
glutAddMenuEntry("quit", 1);
glutAddSubMenu("Resize", sub_menu);
glutAttachMenu(GLUT_RIGHT_BUTTON);
```

## Picking

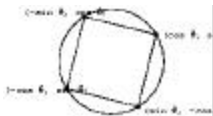
- Picking – identify an object on the display
- Don't return an x,y position
- NOT an easy process
- Two methods: selection and bounding rectangles.

## Selection

Adjusting the clipping region and viewport to keep track of which primitives in a small clipping region are rendered into a region near the cursor. Primitives are entered into a **hit list** to be examined by the user program later on.

## Rotating Cube Program

- We want to create animation
- We continuously keep changing the value of



## Idle Function

```
void spinCube()
{
  /* Idle callback, spin cube 2 degrees about selected axis */
  theta[axis] += 2.0;
  if( theta[axis] > 360.0 ) theta[axis] -= 360.0;
  /* display() */
  glutPostRedisplay();
}
```

`glutIdleFunc(spinCube);`

The idle function works while we do nothing

## Double Buffering

- A complex display may not be drawn in a single refresh cycle
- Double Buffering solves the problem
- Assume two frame buffers: front buffer and back buffer
- Swap these from the application program invoking a display callback

Interaction

Hofstra University

31

## Double Buffering

```
void display(void)
{
    /* display callback, clear frame buffer and z buffer, rotate cube
    and draw, swap buffers */
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    ...
    glRotatef(theta[2], 0.0, 0.0, 1.0);
    colorcube();
    glFlush();
    glutSwapBuffers();
}
```

```
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
```

Updating back buffer and swapping

Interaction

Hofstra University

32

## Good Interactive Programs

- Smooth display
- Interactive devices on display
- Variety of methods to input data
- Easy-to-use interface
- Feedback to user
- Tolerance
- Human consideration (HCI)

Interaction

Hofstra University

33