Input and Interaction
Project Sketchpad

Ivan Sutherland (MIT 1963) established the basic interactive paradigm that characterizes interactive computer graphics:

- User sees an object on the display
- User points to (picks) the object with an input device (light pen, mouse, trackball)
- Object changes (moves, rotates, morphs)
- Repeat
Event Types

Window: resize, expose, iconify
Mouse: click one or more buttons
Motion: move mouse
Keyboard: press or release a key
Idle: nonevent

Define what should be done if no other event is in queue
Callbacks

Programming interface for event-driven input

Define a *callback function* for each type of event the graphics system recognizes

This user-supplied function is executed when the event occurs

GLUT example: `glutMouseFunc(mymouse)`

mouse callback function
GLUT Callbacks

GLUT recognizes a subset of the events recognized by any particular window system (Windows, X, Macintosh)

- glutDisplayFunc
- glutMouseFunc
- glutReshapeFunc
- glutKeyboardFunc
- glutIdleFunc
- glutMotionFunc, glutPassiveMotionFunc
Remember that the last line in main.c for a program using GLUT must be

```c
glutMainLoop();
```

which puts the program in an infinite event loop.

In each pass through the event loop, GLUT

- looks at the events in the queue
- for each event in the queue, GLUT executes the appropriate callback function if one is defined
- if no callback is defined for the event, the event is ignored
The Display Callback

The display callback is executed whenever GLUT determines that the window should be refreshed, for example

- When the window is first opened
- When the window is reshaped
- When a window is exposed
- When the user program decides it wants to change the display

In `main.c`

```c
glutDisplayFunc(mydisplay)
```

identifies the function to be executed

Every GLUT program must have a display callback.
Post Redisplay

Many events may invoke the display callback function.

To avoid multiple displays:

```c
glutPostRedisplay();
```

GLUT checks to see if the flag is set at the end of the event loop.

If set then the display callback function is executed.
Idle and Timers

The idle callback is executed whenever there are no events in the event queue

```c
glutIdleFunc(myidle)
```

The timer callback is executed at specified times

```c
glutTimerFunc(33, myTimerCB)
```

Must remember to reschedule the timer
Using Globals – common approach

The form of all GLUT callbacks is fixed

```c
void mydisplay()
void mymouse(GLint button, GLint state,
             GLint x, GLint y)
```

Must use globals to pass information to callbacks

```c
float t; /*global */

void mydisplay()
{
    /* draw something that depends on t
}
```
The Mouse Callback

`glutMouseFunc(mymouse)`

`void mymouse(GLint button, GLint state, GLint x, GLint y)`

Returns

- which button (`GLUT_LEFT_BUTTON`, `GLUT_MIDDLE_BUTTON`, `GLUT_RIGHT_BUTTON`) caused the event
- state of that button (`GLUT_UP`, `GLUT_DOWN`)

Position in window
The position in the screen window is usually measured in pixels with the origin at the top-left corner.

OpenGL uses a coordinate system with origin at the bottom left:

- Must invert $y$ coordinate returned by callback by height of window
  - $y = h - y$
Keyboard Callback

```c
glutKeyboardFunc (mykey)

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

GLUT defines the special keys in `glut.h`
- Function key 1: GLUT_KEY_F1
- Up arrow key: GLUT_KEY_UP

Can check these in `glutSpecialFunc()` callback
Can also check if one of the modifiers
- GLUT_ACTIVE_SHIFT
- GLUT_ACTIVE_CTRL
- GLUT_ACTIVE_ALT

int mod = glutGetModifiers()
Menus

• GLUT supports pop-up menus
  - A menu can have submenus
• Three steps
  - Define entries for the menu
  - Define action for each menu item
    • Action carried out if entry selected
  - Attach menu to a mouse button
Menus

• In main.c

```c
menu_id = glutCreateMenu(mymenu);
glutAddMenuEntry("clear Screen", 1);
gluAddMenuEntry("exit", 2);
glutAttachMenu(GLUT_RIGHT_BUTTON);
```

entries that appear when right button depressed

identifiers

clear screen
exit
Menus

- Menu callback

```c
void mymenu(int id)
{
    if(id == 1) glClear();
    if(id == 2) exit(0);
}
```

- Add submenus by

```c
glutAddSubMenu(char *submenu_name, submenu id)
```

name in parent menu
Submenus

```c
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);
```
Viewports

Do not have to use the entire window for the image: \texttt{glViewport(x,y,w,h)}
Reshaping the window

We can reshape and resize the OpenGL display window by pulling the corner of the window

What happens to the display?
The Reshape callback

\texttt{glutReshapeFunc(myreshape)}
\texttt{void myreshape( int w, int h)}

Returns width and height of new window (in pixels)
A redisplay is posted automatically at end of execution of the callback
GLUT has a default reshape callback to set the viewport to the window size
Reshape Possibilities

original

reshaped
Default Window Sizing (No Reshape CB)

Resizes the viewport to match the window
Does not change the view volume
Window resizing – Do Nothing

Write a Reshape function, but do nothing in it.

Does not change view volume AND does not change viewport
Better Answer

Make view volume aspect ratio match the viewport AR (which matches the window AR)

Square view volume

Resized window and a viewport with no distortion
void myReshape(int w, int h)
{
    glViewport(0,0,w,h);
    float ar = w/h;
    if( ar < startAR) { // (w <= h){ //taller
        proj = Ortho(vl, vr, vl * (GLfloat) h / (GLfloat) w,
                      vr * (GLfloat) h / (GLfloat) w, 0.1,10.0);
    } else //wider
    {
        proj = Ortho(vb * (GLfloat) w / (GLfloat) h, vt *
                     (GLfloat) w / (GLfloat) h, vb, vt,0.1,10.0);
    }
}