CS 432 – Interactive Computer Graphics

Lecture 3 – Part 3

Input and Interactions

Reading

- Angel: Chapter 2
- Red Book: Chapter 2

Events

- OpenGL is an event-driven API.
- We enter an infinite event loop
- As events are requested, they are put in a queue to be served.
- We've already been allowing for, and serving several events
  - Display
  - Reshape
  - Timer
- But let's look at more of them…

Event Types

- Window: resize, expose, iconify
- Mouse: click one or more buttons
- Motion: move mouse
- Keyboard: press or release a key
- Idle: Nonevent
    - What should be done if nothing else is in the event 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);
```

Custom mouse callback function

GLUT callbacks

- GLUT recognizes a subset of the events recognized by most/all window systems
  - `glutDisplayFunc`
  - `glutMouseFunc`
  - `glutReshapeFunc`
  - `glutKeyboardFunc`
  - `glutKeyboardUpFunc`
  - `glutSpecialFunc`
  - `glutSpecialUpFunc`
  - `glutIdleFunc`
  - `glutMotionFunc`
  - `glutPassiveMotionFunc`

Display Callback

- The display callback is executed whenever GLUT determines that the window should be refreshed.
- For example
  - When the window if first opened
  - When the window is reshaped
  - When the window is exposed
  - When the user program decides it wants to change the program.
- Every GLUT program must have a display callback:

```
glutDisplayFunc(mydisplay)
```

Posting Redisplays

- As was mentioned, the user program can request a redisplay.
- To do this we call the `glutPostRedisplay();` function
- Why not just call the callback function directly?
  - Many things might we calling the display callback.
  - We want to only do it once per event loop.

The Mouse Callback

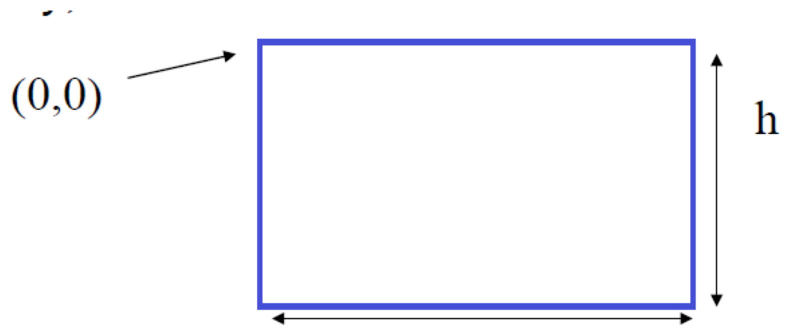
- `glutMouseFunc(mymouse);`
- `void mymouse(GLint button, GLint state, GLint x, GLint y);`
- Passes in
  - Which button (`GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT BUTTON`) caused the event
  - State of the button (`GLUT_UP, GLUT_DOWN`);
  - Position in window.

Positioning

- The mouse callback (as well as other callbacks) returns the *pixel* clicked on.
- This position is usually measured from the top-left corner.
- Recall that our camera coordinates are in the range $(-1,-1) \rightarrow (1,1)$ so we need to do a quick/simple transformation to see where we clicked in the camera coordinate system.

Positioning

- Given:
  - (x_{click}, y_{click})
- Want
  - $-1 \leq (x_{camera}, y_{camera}) \leq 1$
- $x_{cam} = 2\frac{x_{click}}{w} - 1$
- $y_{cam} = 2\frac{(h - y_{click})}{h} - 1$



Motion Callback

- If a mouse button is depressed and the mouse is moving, then a motion callback is made: `glutMotionFunc(mymotion);`
- We can also have a callback be made if the mouse is moving *without* a button being depressed: `glutPassiveMotionFunc(mymotion);`

Using the Keyboard

- `glutKeyboardFunc(mykey);`
- `void mykey(unsigned char key, int x, int y);`
- Passes in
  - ASCII code of key depressed and mouse location
-
```
void mykey(unsigned char key, int x, int y){
     if(key==‘Q’ || key == ‘q’)
           exit(0);
}
```

Special Keys

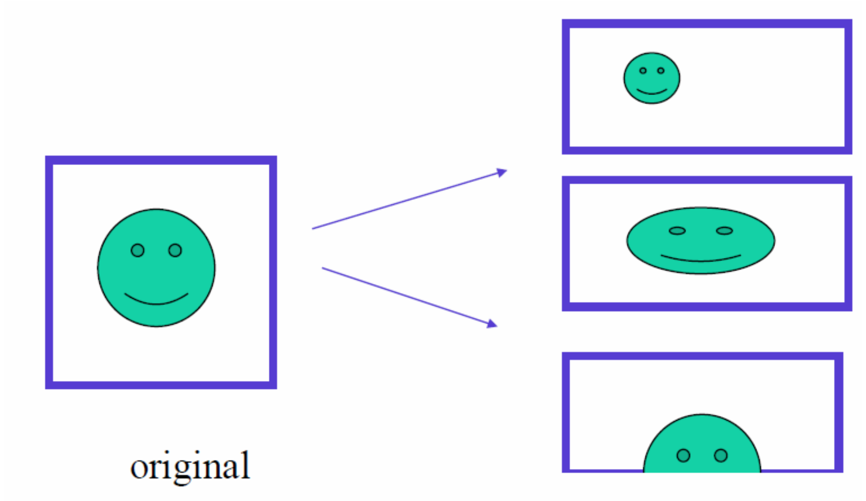
- GLUT defines special keys
  - Function key 1: GLUT_KEY_F1
  - Up arrow key: GLUT_KEY_UP
- These are handled in their own callback, `glutSpecialFunc`
- Within this callback we can also check if one of the modifiers are being depressed via the `glutGetModifiers()` function
- Modifiers include
  - GLUT_ACTIVE_SHIFT
  - GLUT_ACTIVE_CTRL
  - GLUT_ACTIVE_ALT

Using the Keyboard

- Sometimes you might want to keep track of which keys are currently being held down.
- You could do this by combining the `glutKeyboardFunc` with the `glutKeyboardUpFunc` callbacks.
- The `glutKeyboardUpFunc` (or `glutSpecialUpFunc`) is triggered when a key is released.
- How could we use this?
  - Have some Boolean value set when key down and unset when key up…

Reshape Callback

- As mentioned, we can resize the window several ways.
- When this is done, the display callback will be called.
- But we also have the opportunity to decide if we want to draw to the entire display or a sub-window.
    - This allows us to determine the aspect ratio



Reshape Callback

- `glutReshapeFunc(reshape)`
- `void reshape(int w, int h);`
- Here we can force drawing to a particular viewport

```
void reshape(int w, int h){
   glViewport(0,0,w,h);
}
```

Timers

- Timers are callbacks that are triggered after a specified number of milliseconds.
- We can also pass an integer ID if we want
- `glutTimerFunc(1000, timer,ID);`