

Input Devices and Interaction

Tom Kelliher, CS 320

Feb. 21, 2011

1 Administrivia

Announcements

Project 1 due Wednesday.

Assignment

Read 3.4, 3.5, 3.7–3.9.

From Last Time

Color, projections, viewports, project lab.

Outline

1. Input devices, programming models.
2. API.

Coming Up

Display lists, menus, picking.

2 Input Devices

Physical devices: Keyboard, mouse, trackball, data tablet, light pen, touch screen, joy stick.

How do mice, light pens work?

Pointing device necessary to interact with graphics.

What about 3-D interaction? (Space ball, data gloves)

2.1 Logical Input Devices

1. String.
2. Locator: Returns (x, y) . Convert window coordinates to world coordinates.
3. Pick: Select an object. Must determine what object was selected.
4. Choice: widget menus.
5. Dial: scroll bars. Again, widgets.
6. Stroke: mouse drag.

3 Input Device Program Interaction Models

Terminology:

1. Measure: The data — (x, y) , input string, etc.
2. Trigger: User indication that the measure should be taken — “Enter” key, mouse click.

Interaction modes:

1. Request (synchronous wait) mode.
Measure not returned until trigger.
Advantages/disadvantages.
2. Sample (asynchronous poll) mode.
Measure returned any time.
Advantages/disadvantages.
3. Event mode.
Queue of (trigger, measure) pairs. Asynchronous.
Advantages/disadvantages.
OpenGL, callbacks, and `glutMainLoop()`.

4 Input Device API

1. `glutMouseFunc(pointerToMouseCallbackFunction)`
2. `void MouseCallbackFunction(int button, int action, int x, int y)`
 - (a) `GLUT_LEFT_BUTTON`, etc.
 - (b) `GLUT_UP`, `GLUT_DOWN`.
 - (c) `x` and `y` are *window*-relative coordinates.

Example:

```
// ...  
  
    glutMouseFunc(mouse);  
  
// ...  
  
void mouse(int btn, int action, int x, int y)
```

```

{
  if (btn == GLUT_LEFT_BUTTON && action == GLUT_DOWN)
  {
    myInit(rows, cols, 1);
    visit(1, 1);
    glutPostRedisplay();
  }
  else if (btn == GLUT_RIGHT_BUTTON && action == GLUT_UP)
    exit(0);
}

```

3. `glutMotionFunc(pointerToMotionFunction)`

Also, `glutPassiveMotionFunc()`.

4. `void MotionFunction(int x, int y)`

- (a) Active motion — mouse button depressed.
- (b) How do we know which mouse button is depressed?
- (c) Again, window-relative coordinates.

5. `glutKeyboardFunc(pointerToKeyboardFunction)`

6. `void KeyboardFunction(unsigned char key, int x, int y)`

- (a) `key` is ASCII of key depressed.
- (b) Yet again, window-relative coordinates.
- (c) See `glutSpecialFunc()` for non-ASCII keys.

Example:

```

#define ESC 0x1b
// ...

glutKeyboardFunc(keyboard);

```

```

// ...

void keyboard(unsigned char key, int x, int y)
{
    switch (key)
    {
        case 'w':
        case 'W':
            printf("The Clinton people took all these keys.\n");
            break;

        case ESC:
            exit(0);
            break;

        case '!':
            globalThermonuclearWar();
            // Not reached.
            break;

        // ...

        others:
            fatal("Un-recognized key.\n");
            break;
    }
}

```

7. `glutDisplayFunc(pointerToDisplayFunction)`

8. `void DisplayFunction(void)`

(a) Callback generated by window system events.

(b) Can self-generate with `glutPostRedisplay()`.

9. `glutReshapeFunc(pointerToReshapeFunction)`

10. `void ReshapeFunction(GLsizei w, GLsizei h)`

As previously discussed, have to reconcile clipping region aspect ratio to window aspect ratio.