

CS770/870 Fall 2008

Introduction to User Interaction

Using GLUT

12:41

CS770/870 Fall 2008 Bergeron/Plumlee

1

Preview

- Problem of “listening” to what user wants
- Menus
- Mouse input & problem with GLUT
- Quick SCons intro

12:41

CS770/870 Fall 2008 Bergeron/Plumlee

2

The Problem

- How do you “listen” to what the user wants—respond to user input?
- The user may want to...
 - Invoke an operation
 - Specify a location
 - Select or specify a value
 - Indicate a range of values
 - Control an ongoing process
 - ...

12:41

CS770/870 Fall 2008 Bergeron/Plumlee

3

Some Solutions

- Provide special hardware for each thing the user might want to do
 - Car radio, DVD player, cockpit
 - Hard to extend product, transfer training
- Provide a few versatile controls, and map what the user might want to do to these
 - Mouse, keyboard
 - iPod wheel & buttons
 - Easier to extend/overload, transfer training

12:41

CS770/870 Fall 2008 Bergeron/Plumlee

4

Interaction Using GLUT

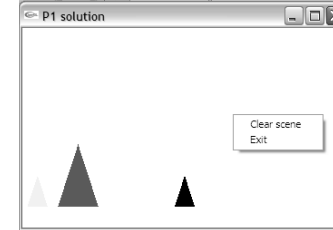
- GLUT supports mouse & keyboard input
 - Mouse-initiated menus (option/command selection)
 - Mouse button (up or down events, position)
 - Mouse motion (position)
 - ... a few other interpretations of the mouse
 - Keyboard key (which key, mouse position)

12:41

CS770/870 Fall 2008 Bergeron/Plumlee

5

Using GLUT Menus



- GLUT only supports pop-up menus
 - Pop-up where you press the mouse button
 - Register callback as if a different kind of event
 - Notion of *current* menu during creation

12:41

CS770/870 Fall 2008 Bergeron/Plumlee

6

Using GLUT Menus

```
enum {MY_EXIT_CMD, MY_CLEAR_CMD};
void processRightButtonMenu( int command )
{
    if (command == MY_EXIT_CMD) exit(0);
    if (command == MY_CLEAR_CMD) shapes.clear();
}
int main( int argc, char **argv )
{ ...
    glutCreateMenu( processRightButtonMenu );
    glutAddMenuEntry( "Clear scene", MY_CLEAR_CMD );
    glutAddMenuEntry( "Exit", MY_EXIT_CMD );
    //attach current menu to a right-click
    glutAttachMenu( GLUT_RIGHT_BUTTON );
}
```

12:41

CS770/870 Fall 2008 Bergeron/Plumlee

7

Want Another GLUT Menu?

```
int main( int argc, char **argv )
{ ...
    glutCreateMenu( processRightButtonMenu );
    glutAddMenuEntry( "Clear scene", MY_CLEAR_CMD );
    glutAddMenuEntry( "Exit", MY_EXIT_CMD );
    //attach current menu to a right-click
    glutAttachMenu( GLUT_RIGHT_BUTTON );
    //---Second menu---
    glutCreateMenu( processLeftButtonMenu );
    glutAddMenuEntry( "Red", MY_RED_OPTION );
    glutAddMenuEntry( "Blue", MY_BLUE_OPTION );
    //attach current menu to a left-click
    glutAttachMenu( GLUT_LEFT_BUTTON );
}
```

12:41

CS770/870 Fall 2008 Bergeron/Plumlee

8

Handling GLUT Mouse Events

```

void mouseButtonCallback(int button, int state,
                        int x, int y)
{
    if (state == GLUT_DOWN)
        std::cerr << button << "pressed " << x << ", "
                  << y << std::endl;
    else
        std::cerr << button << "released " << x << ", "
                  << y << std::endl;
}

int main( int argc, char **argv )
{ ...
  glutMouseFunc (mouseButtonCallback);
  ...
}

```

12:41

CS770/870 Fall 2008 Bergeron/Plumlee

9

GLUT Callback problem

- How do you get mouse input to classes that need it, without main knowing about them?
- Example: Tool palette that...
 - requires mouse-press event to select tool
 - initiates interaction mode (like rubberbanding) that uses mouse press, motion, & release events

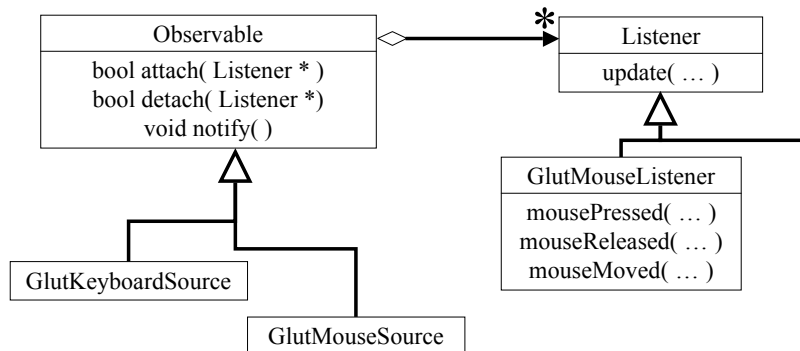
12:41

CS770/870 Fall 2008 Bergeron/Plumlee

10

An O-O Solution

- Observer pattern



12:41

CS770/870 Fall 2008 Bergeron/Plumlee

11

Sample Observable Code

```

/** Add the passed listener to our list, return
    true if it works
 */
bool Observable::attach(Listener *l)
{
    listeners.push_back(l);
    return true;
}

```

12:41

CS770/870 Fall 2008 Bergeron/Plumlee

12

Sample Observable Code

```
/** Remove the passed listener to our list,
 * return true if it was there, else false
 */
bool Observable::detach(Listener *l)
{
    std::vector<Listener *>::iterator it;
    for(it = listeners.begin();
        it < listeners.end(); it++)
    {
        if (*it == l)
        {
            listeners.erase(it);
            return true;
        }
    }
    return false;
}
12:41 CS770/870 Fall 2008 Bergeron/Plumlee 13
```

Sample Observable Code

```
/** Go through our list of listeners and let
 * them know about the change.
 */
void Observable::notify(int info, int x, int y)
{
    std::vector<Listener *>::iterator it;
    for(it = listeners.begin();
        it < listeners.end(); it++)
    {
        (*it)->update(info, x, y);
    }
}
12:41 CS770/870 Fall 2008 Bergeron/Plumlee 14
```

Sample GlutMouseSource Code

```
/** Register our callbacks with GLUT
 */
GlutMouseSource::GlutMouseSource()
{
    glutMouseFunc(mouseButtonCallback);
    glutMotionFunc(mouseMotionCallback);
}

GlutMouseSource::~~GlutMouseSource()
{
    glutMouseFunc(NULL);
    glutMotionFunc(NULL);
}
12:41 CS770/870 Fall 2008 Bergeron/Plumlee 15
```

Sample GlutMouseSource Code

```
enum {MOUSE_BUTTON_1 = 0x01,
      MOUSE_BUTTON_2 = 0x02,
      MOUSE_BUTTON_3 = 0x03,
      MOUSE_BUTTON_MASK = 0x0F,
      MOUSE_MOTION = 0x10,
      MOUSE_BUTTON_DOWN = 0x20,
      MOUSE_BUTTON_UP = 0x40,
      MOUSE_ACTION_MASK = 0xF0
};
```

Action	Button

Sample GlutMouseSource Code

```
//Notify all listeners about a button state change
void GlutMouseSource::mouseButtonCallback(
    int button, int state, int x, int y)
{
    int info = button;
    if (state == GLUT_DOWN) //add mouse action flag
        info |= MOUSE_BUTTON_DOWN;
    else
        info |= MOUSE_BUTTON_UP;
    instance->notify(info, x, y);
}
// Notify all listeners about a mouse motion change
void GlutMouseSource::mouseMotionCallback(
    int x, int y)
{
    int info = MOUSE_MOTION; //add mouse action flag
    instance->notify(info, x, y);
}
12:41 CS770/870 Fall 2008 Bergeron/Plumlee 17
```

Another Problem: Constructor

- GlutMouseSource will do what we want now, but can only have one
- How do we enforce only one instance?
- Solution: Singleton Pattern

12:41

CS770/870 Fall 2008 Bergeron/Plumlee

18

GlutMouseSource as Singleton

```
class GlutMouseSource : public Observable
{
public:
    static GlutMouseSource *getInstance();
    enum {MOUSE_BUTTON_1 = 0x01,
        ... };
protected:
    GlutMouseSource();
    virtual ~GlutMouseSource();

    static GlutMouseSource *instance;
    static void mouseButtonCallback(
        int button, int state, int x, int y);
    static void mouseMotionCallback(int x, int y);
};
12:41 CS770/870 Fall 2008 Bergeron/Plumlee 19
```

GlutMouseSource as Singleton

```
GlutMouseSource *GlutMouseSource::instance = NULL;
...

/** Return the singleton instance.
 * If it does not exist, create one.
 */
GlutMouseSource *GlutMouseSource::getInstance()
{
    if (!instance)
        instance = new GlutMouseSource();
    return instance;
}
```

12:41

CS770/870 Fall 2008 Bergeron/Plumlee

20

Review

- GLUT mouse & keyboard functions as one solution to user interaction problem
- GLUT Pop-up menus
- GLUT mouse callbacks
- Observer pattern to handle GLUT callbacks
- Singleton pattern ensures exactly one source

12:41

CS770/870 Fall 2008 Bergeron/Plumlee

21

Quick SCons Intro

- SCons is a cross-platform build tool
- Built on Python
 - Cross-platform scripting language
 - Object-oriented
- Basic model:
 - Create SConstruct file that lists needed parts
 - SCons figures out how to use compilers and other tools to generate program, libraries, etc.
 - Handles many dependencies automatically

12:41

CS770/870 Fall 2008 Bergeron/Plumlee

22

Sample Scons file

```
env = Environment(CPPPATH = ['../glut'],  
                 LIBPATH = ['../glut'])
```

```
env.Program('Demol',  
           ['main.cpp', 'House.cpp',  
            'GraphicalObject.cpp'])
```

- Specify `Environment` class to suit needs
- Python allows...
 - Method arguments in any order by giving name
 - Lists as arguments using brackets: `[x, y, z]`

12:41

CS770/870 Fall 2008 Bergeron/Plumlee

23

Resources

- Official Scons Site (w/ download links)
<http://www.scons.org/>
- SCons User Guide:
<http://www.scons.org/doc/1.0.1/HTML/scons-user/book1.html>
- Python Site
<http://www.python.org/>
- Python Tutorials (if you need more info):
<http://docs.python.org/tut/tut.html>

12:41

CS770/870 Fall 2008 Bergeron/Plumlee

24

Next

- GLUI—A basic user interface API