ECE 3574: Applied Software Design

Event Driven Programming

Announcement

The relative Project 1 beta/final grade is being re-weighted.

- Project 1 beta is now 25 points. (5.7 % course grade)
- Project 1 final is now 75 points. (17 % course grade)

You can scale your grade from feedback.log to get the new beta: 25*(grade/45)

Today we will learn how to design systems that respond to internal and external events in an application.

- Events and Event Handlers
- Events from Windowing System
- Timers and other internal events
- Observer Pattern
- Callbacks versus Polymorphism for implementing event handlers
- Qt Event System
- Exercise

Events are inputs that are not predictable from the program flow.

Examples:

- Hardware Event: the user presses a key on a keypad
- Software Event: the user clicks on a button in a windowing system

The program should be able to respond to these events, i.e *handle* them, whenever they occur.

Typically events are collected in an event loop using polling

Round-Robbin

```
while(true){

  // check status of switch
  // handle if changed

  // ... etc.
}
```

Typically events are collected in an event loop using polling

Queuing, or *posting* the event (like onto a bulletin board)

```
while(true){

  // check status of switch
  // post the event, queue it to be handled

  // ... etc.

  // handle N events from the queue
}
```

The code that is run in response to an event is a *handler*.

The handler should:

- do the minimum amount of work possible
- never block execution

Otherwise the system lags to input or locks up and does not respond to events.

How much work can be done?

- Each iteration of the event loop should be limited in time.
- How much depends on the application
- in a user interface around 250ms
- in a control loop, perhaps as little as a 1ms
- Add up the total number of events and the time to execute each

How does one do more work in a handler?

- concurrency, let the OS handle it (see lectures 18-27)
- split work into small chunks, post an event itself
- implement a coroutine, a function that can be restarted where it left off (not discussed)

Examples of Events from a Windowing System

- show/draw/render the object
- focus the object
- mouse enter/leave
- mouse down, up for left, right, middle, etc
- key K press/release
- resize object
- move object
- gestures

Examples of internal events

- timers
- events posted by other handlers

Event systems are an example of the *Observer Pattern*

Observers are objects which observe other objects. Possible implementations:

- callback functions
- dynamic polymorphism (inheritance)

See example code.

Exercise

See website

Next Actions and Reminders

- Read about Qt Signals and Slots
- Project 1 Final is Due Friday, March 3rd 2017 at 5 PM EST.