CSE 142
Computer Programming I

Event Driven Programming

© 2000 UW CSE

Two Models of Programming

Traditional, batch processing model:
- "Input-process-output"
- Program starts up, reads some input, processes the input, produces some output, terminates.
- Especially suitable for programs that run with little or no human interaction

More modern model: *event-driven* programming

Event-Driven Programming

- Program starts, sets itself up.
- Program enters an "event loop", waiting for some event or command to happen:
  - mouse click, key click, timer, menu selection, etc.
- Program performs operation ("handles" the event or command)
- Program goes back to its wait loop

Programs using UW's GP142 graphics package follow this model

Simple Command Interpreter

Repeatedly read in "commands" and handle them.

Input (symbolized by single characters)
- a -- execute command A by calling *process_A( )*
- b -- execute command B by calling *process_B( )*
- q -- quit

Pseudocode for main loop:
- get next command
- if a, execute command A
- if b, execute command B
- if q, signal quit

Command Interpreter Loop Control Schema

- repeat until quit signal
- use variable "done" to indicate when done

```
set done to false
while not done {
    body statements
    if quit command, set done to true
}
```


Command Interpreter main ( )

```
int main(void) {
   char command;
   int done;

   done = FALSE;
   while (! done){    /* Input command from user */
      command = ReadCommand( );
      switch (command){
      case 'A':
      case 'a':
         process_A( );        /* Execute command A */
         break;
      case 'B':
      case 'b':
         process_B( );        /* Execute command B */
         break;
      case 'Q':
      case 'q':
         done = TRUE;         /* quit */
         break;
      default:
         printf("Unrecognized command\n");
      }
   }
   return 0;
}
```

QOTD: Event Loops without switch

Find the event loop in HW4.c. Rewrite it without using the switch statement. Which version do you prefer? Can any switch be written using only if statements? Is the reverse true? Here's the original (abbreviated):

```
while (!quit) {
        nextEvent = GP142_await_event(&mouseX, &mouseY, &keyPressed);
        switch (nextEvent) {
            case GP142_QUIT:
                    quit = TRUE; /* set flag to terminate loop */
                    break;
            case GP142_MOUSE:
                    break;
            case GP142_KBD:
                    if (keyPressed == 'q' || keyPressed == 'Q')
                            { quit = TRUE; }
                    else { /* The user has typed in some keyboard input. ... }
                    break;
            case GP142_PERIODIC:
                    if (gameStatus == GAME_PLAYING) {...}
            break;
            default:                                                    K-7
                    break;
            }
        }
```