CSE 142
Computer Programming I

Event Driven Programming

Event-Driven Programming

Modern programs tend to be "event-driven"

- Program starts, sets itself up.
- Program enters a 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

UW's GP142 graphics package follows this kind of 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 ( )

```
#define FALSE   0
#define TRUE    1

int main(void) {
   char command;
   int done;
   done = FALSE;
   while (! done){
      /* get command from user */
      command = ReadCommand( );
      /* execute appropriate command */
      if (command == 'A' || command == 'a'){
         /* execute command A */
         process_A( );
      } else if (command == 'B' || command ==  'b') {
         /* execute command B */
         process_B( );
      } else if (command == 'Q' || command == 'q') {
         /* quit */
         done = TRUE;
      } else {
         printf("Unrecognized command\n");
      }
   }
   return 0;
}
```