Dr. Sarah Abraham
University of Texas at Austin
Computer Science Department

Interactivity

Elements of Graphics
CS324e

Input Devices

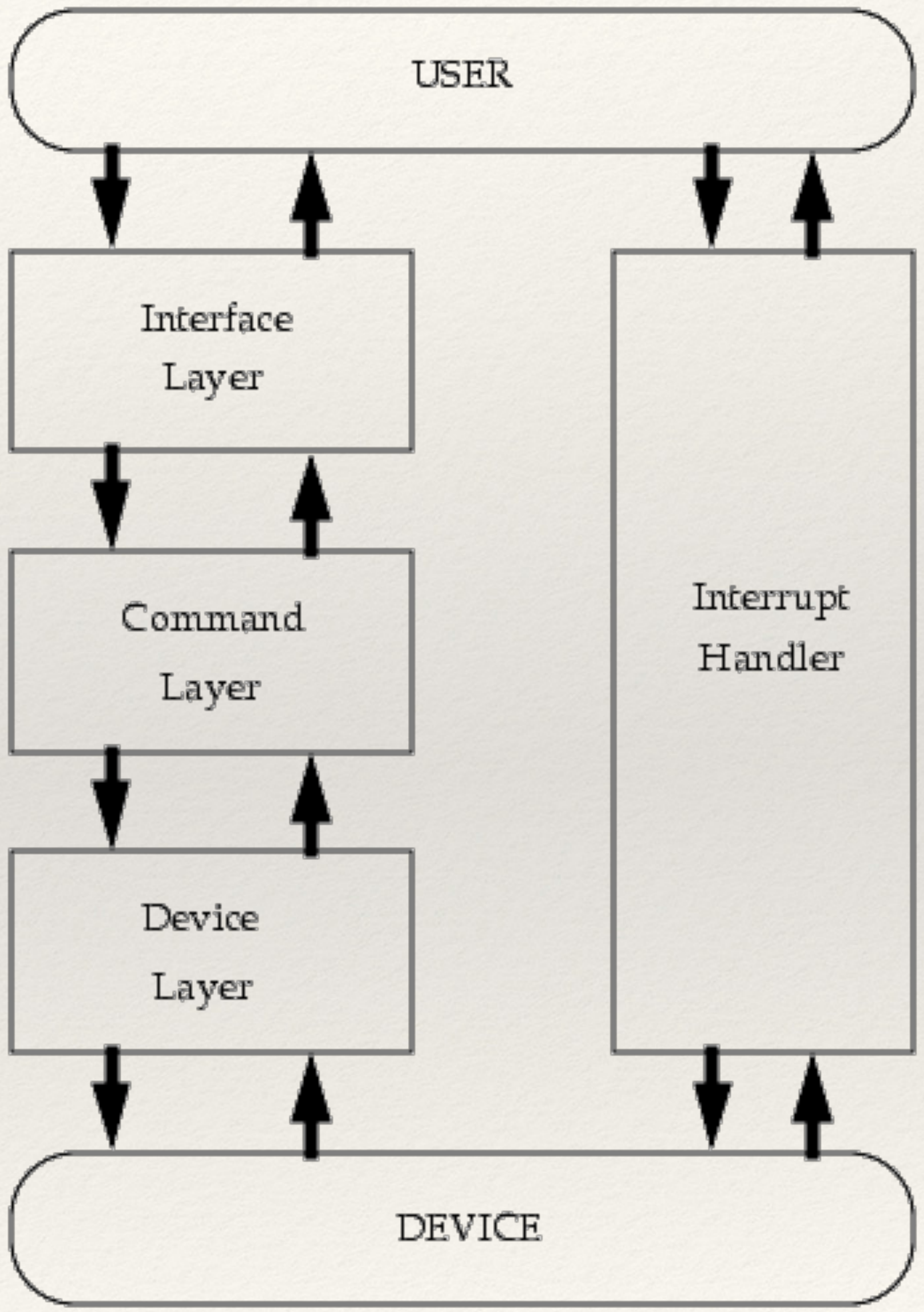
- Input devices allow humans to issue commands more easily to computers
  - Mouse
  - Keyboard
  - Many, many others

Device Interface

- Devices and computers must communicate
- The “bus” or communications system provides necessary hardware and software
- Drivers provide software interface to access device information

Input Pipeline

- Program issues a driver routine
- Driver communicates with device
- Device triggers *interrupt* to notify program of event



Events

- Events are triggered occurrences that are handled by the program
- Event-driven programming allows for efficient handling of:
    - Device input
    - Timers
    - Event loops
- But for now let's focus on device input…

Mouse Input

- Variables, `mouseX` and `mouseY`, register the mouse’s x and y coordinates
    - Store the coordinate data as `ints`
    - Values registered only if `draw()` commands are issued
- Variables, `pmouseX` and `pmouseY`, store the mouse values from the previous frame

Mouse Buttons

- `mousePressed` stores whether or not a mouse button is pressed: true or false
  - `if (mousePressed) { //do something }`
- `mouseButton` stores mostly recently pressed button: LEFT, CENTER, or RIGHT
  - `if (mouseButton == LEFT) { //do something }`

Consider…

```
if (mousePressed) {
   if (mouseButton == LEFT) {
      background(0);
   } else {
      background(255);
   }
}
fill(110);
ellipse(mouseX, mouseY, 30, 30);
```

Keyboard Input

- `keyPressed` stores whether a key is pressed: true or false
- `key` stores the most recently pressed key value
- `key` contains values of ASCII-specified characters
    - Alphanumeric values
    - BACKSPACE, TAB, ENTER, RETURN,* ESC, DELETE
- `keyCode` stores non-ASCII-specified characters
    - ALT, CONTROL, SHIFT, UP, DOWN, LEFT, RIGHT

* ENTER and RETURN depend on the target platform

Consider…

```
if (keyPressed && (key == ‘a’ || key == ‘A’)) {
  text(key, mouseX, mouseY);
}
if (keyPressed && key == CODED) {
  if (keyCode == DOWN) {
    background(110);
  }
}
```

Events in Processing

- Events allow for better flow within the program
- Event functions only called when event occurs
- Key and mouse inputs are stored until the end of `draw()`

Mouse and Keyboard Events

- Key and mouse events called **only** when event occurs
- Inputs stored until the end of `draw()`
- Implementable methods to handle events:
  - `mousePressed()`
  - `mouseReleased()`
  - `mouseMoved()`
  - `mouseDragged()`
  - `keyPressed()`
  - `keyReleased()`

Draw Loop

- A kind of system-generated event
- Called every 16ms by default
- Renders programmer-dictated content to screen every time it is run
- Requests a new `draw()` event upon completion
- Programmer has control over:
  - Content `draw()` renders
  - When `draw()` renders

Modifying the Draw Loop

- `noLoop()` stops the `draw()` command
- `loop()` resumes the `draw()` command
- `redraw()` executes the `draw()` command only once

Hands-on: Triggering Events

- Today's activities:
  1. Use variables `mousePressed` and `mouseButton` in the draw loop to control the sketch's background color
  2. Reimplement this behavior in the `mousePressed()` function
  3. Use variables `mouseX` and `mouseY` in the `mouseMoved()` function to draw a point that follows the mouse
  4. Display different objects to screen using the `keyPressed` variable. These objects should remain on screen even after the key is released
  5. Reimplement this behavior in the `keyPressed()` function