Input and Interaction

Thanks to Ed Angel
Professor Emeritus of Computer Science,
University of New Mexico

Objectives

• Introduce the basic input devices
  – Physical Devices
  – Logical Devices
  – Input Modes
• Event-driven input
• Introduce double buffering for smooth animations
• Programming event input with javascript/HTML5
Project Sketchpad

• Ivan Sutherland (MIT 1963) established the basic interactive paradigm that characterizes interactive computer graphics:
  – User sees an object on the display
  – User points to (picks) the object with an input device (light pen, mouse, trackball)
  – Object changes (moves, rotates, morphs)
  – Repeat

Graphical Input

• Devices can be described either by
  – Physical properties
    • Mouse
    • Keyboard
    • Trackball
  – Logical Properties
    • What is returned to program via API
      – A position
      – An object identifier
  • Modes
    – How and when input is obtained
      • Request or event
Physical Devices

- mouse
- data tablet
- joy stick
- Multi-touch
- space ball

Incremental (Relative) Devices

- Devices such as the data tablet return a position directly to the operating system.
- Devices such as the mouse, trackball, and joy stick return incremental inputs (or velocities) to the operating system.
  - Must integrate these inputs to obtain an absolute position:
    - Rotation of cylinders in mouse
    - Roll of trackball
    - Difficult to obtain absolute position
    - Can get variable sensitivity
Logical Devices

- Consider the C and C++ code
  - C++: `cin >> x;`
  - C: `scanf ("%d", &x);`
- What is the input device?
  - Can’t tell from the code
  - Could be keyboard, file, output from another program
- The code provides *logical input*
  - A number (an `int`) is returned to the program regardless of the physical device

Graphical Logical Devices

- Graphical input is more varied than input to standard programs which is usually numbers, characters, or bits
- Two older APIs (GKS, PHIGS) defined six types of logical input
  - **Locator**: return a position
  - **Pick**: return ID of an object
  - **Keyboard**: return strings of characters
  - **Stroke**: return array of positions
  - **Valuator**: return floating point number
  - **Choice**: return one of n items
X Window Input

- The X Window System introduced a client-server model for a network of workstations
  - **Client**: OpenGL program
  - **Graphics Server**: bitmap display with a pointing device and a keyboard

Input Modes

- Input devices contain a *trigger* which can be used to send a signal to the operating system
  - Button on mouse
  - Pressing or releasing a key
- When triggered, input devices return information (their *measure*) to the system
  - Mouse returns position information
  - Keyboard returns ASCII code
Request Mode

- Input provided to program only when user triggers the device
- Typical of keyboard input
  - Can erase (backspace), edit, correct until enter (return) key (the trigger) is depressed

Event Mode

- Most systems have more than one input device, each of which can be triggered at an arbitrary time by a user
- Each trigger generates an event whose measure is put in an event queue which can be examined by the user program
Event Types

- Window: resize, expose, iconify
- Mouse: click one or more buttons
- Motion: move mouse
- Keyboard: press or release a key
- Idle: nonevent
  - Define what should be done if no other event is in queue

Callbacks

- Programming interface for event-driven input uses *callback functions* or *event listeners*
  - Define a callback for each event the graphics system recognizes
  - Browsers enters an event loop and responds to those events for which it has callbacks registered
  - The callback function is executed when the event occurs
Execution in a Browser

- Start with HTML file
  - Describes the page
  - May contain the shaders
  - Loads files
- Files are loaded asynchronously and JS code is executed
- Then what?
- Browser is in an event loop and waits for an event
onload Event

• What happens with our JS file containing the graphics part of our application?
  – All the “action” is within functions such as init() and draw()
  – Consequently these functions are never executed and we see nothing

• Solution: use the onload window event to initiate execution of the init function
  – onload event occurs when all files read
  – window.onload = init;

Example

• Canvas 0 and Canvas 1
Double Buffering

• Although we are rendering the square, it always into a buffer that is not displayed
• Browser uses double buffering
  – Always display front buffer
  – Rendering into back buffer
  – Need a buffer swap
• Prevents display of a partial rendering

Triggering a Buffer Swap

• Browsers refresh the display at ~60 Hz
  – redisplay of front buffer
  – not a buffer swap
• Trigger a buffer swap though an event
• Two options for triggering a buffer swap
  – Interval timer
  – requestAnimFrame
Interval Timer

- Repeatedly executes a function after a specified number of milliseconds
  - Also generates a buffer swap

```javascript
setInterval(draw, interval);
```

- Note an interval of 0 generates buffer swaps as fast as possible

Interval Timer

- GUI-setInterval.html
Timeout Timer

• Executes a function once after a specified number of milliseconds
  – Also generates a buffer swap

setTimeout(draw, delay);

• Note a delay of 0 generates buffer swaps as fast as possible

Timeout Timer

• GUI-setTimeout.html
requestAnimFrame

- Recursively call the drawing function
  - Also generates a buffer swap

- Buffer swaps as fast as possible

requestAnimFrame

- GUI-RAF.html

- GUI-mouseFocus.html
Putting it all together

• Learn to build interactive programs using event listeners
  – Buttons
  – Sliders
  – Menus
  – Mouse
  – Keyboard
  – Reshape

Adding a Button

• Let’s add a button to stop the ball.

• Clear the interval
  – clearInterval(animBall);
The Button

• In the HTML file

<button onclick="myStopFunction()">Stop Ball</button>

  – Uses HTML button tag
  – onclick gives the callback we can use in JS file
  – Text “Stop Ball” displayed in button

• Clicking on button generates a click event
• Note we are using default style and could use CSS or jQuery to get a prettier button

Button Event Listener

• We could have defined a listener
  – no listener and the event occurs but is ignored
• Two forms for event listener in JS file

```javascript
var myButton = document.getElementById("stopButton");
// needs an id for the button
myButton.addEventListener("click", function() {
  do something as this is the callback;
});

document.getElementById("stopButton").onclick = function() {do something as this is the callback;};
```
onclick Variants

```javascript
myButton.addEventListener("click", function() {
  if (event.button == 0) { direction = !direction;
  }
});

myButton.addEventListener("click", function() {
  if (event.shiftKey == 0) { direction = !direction;
  }
});

<button onclick="direction = !direction"></button>
```

Buttons

- GUI-setInterval.html
- GUI-setInterval-Stop.html
- GUI-setInterval-Start-Stop.html
- GUI-setInterval-Stop-Speed.html
Controlling Ball Speed

GUI-RAF-speedControl.html

GUI-RAF-speedControl-stop.html

Slider Element

• Puts slider on page
  – Give it an identifier
  – Give it minimum and maximum values
  – Give it a step size needed to generate an event
  – Give it an initial value

• Use div tag to put below canvas

```
<div>
speed 0 <input id="slide" type="range"
  min="0" max="100" step="10" value="50" />
100 </div>
```
onchange Event Listener

document.getElementById("slide").onchange = function() { delay = event.srcElement.value; };

Controlling Ball Speed

GUI-speedSlider.html
Menus

- Use the HTML select element
- Each entry in the menu is an option element with an integer value returned by click event

```html
<select id="Controls" size="4">
  <option value="0">Start</option>
  <option value="1">Stop</option>
  <option value="2">Slower</option>
  <option value="3">Faster</option>
</select>
```

Menu Listener

```javascript
document.getElementById("Controls").onclick = function(event) {
  switch(event.srcElement.index) {
    case 0:
      start = 1;
      raf = window.requestAnimationFrame(draw);
      break;
    case 1:
      start=0;
      window.cancelAnimationFrame(raf);
      break;
    case 2:
      delay += 10;
      break;
    case 3:
      delay -= 10;
      break;
  }
}
```
Menu

- GUI-RAF-speedControl-stop-menu.html

Using keydown Event

```javascript
window.onkeydown = function( event ) {
    var key = String.fromCharCode(event.keyCode);
    switch( key ) {
        case '0':
            start = 1;
            raf = window.requestAnimationFrame(draw);
            break;
        case '1':
            start=0;
            window.cancelAnimationFrame(raf);
            break;
        case '2':
            delay += 10;
            break;
        case '3':
            delay -= 10;
            break;
    }
};
```
Keydown event w/ menu

- GUI-RAF-speedControl-stop-menu-keyboard.html