Input and Interaction

Fall 2024
9/26/2024
Kyoung Shin Park
Computer Engineering
Dankook University

Overview

- Introduce the basic input devices
  - Physical input devices
    - Mouse, Keyboard, Trackball
  - Logical input devices
    - String, Locator, Pick, Choice, Valuators, Stroke device
- Input modes
  - Request mode
  - Sample mode
  - Event mode
- Devices & Event-driven programming
  - mouse, keyboard,..

Interaction

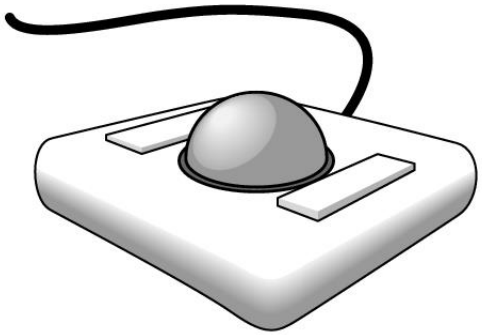
- One of the major advances in computer technology is that users can interact using computer screens.
- Interaction
  - The user takes action through an interactive device such as a mouse.
  - The computer detects user input.
  - The program changes it state in response to this input.
  - The program displays this new status.
  - The users sees the changed display.
  - The processes in which the user reacts to this change are repeated.

Graphical Input

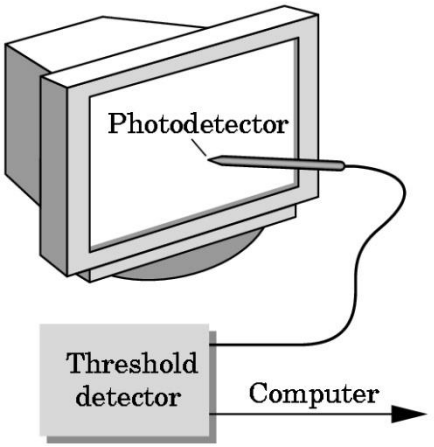
- Input devices can be described either by
  - Physical properties
    - Mouse, Keyboard, Trackball
  - Logical properties
    - Characterized by upper interface with application program, not by physical characteristics
- Input modes
  - The way an input device provides an input to an application program can be described as a **measurement** process and device **trigger**.
    - Request mode
    - Sample mode
    - Event mode

Physical Input Devices

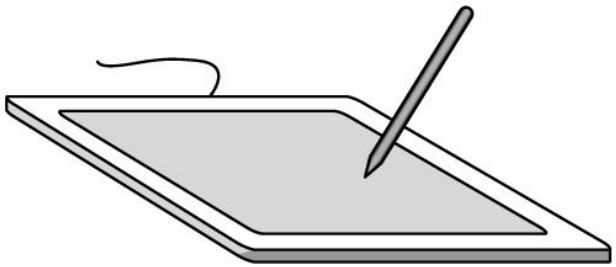
mouse



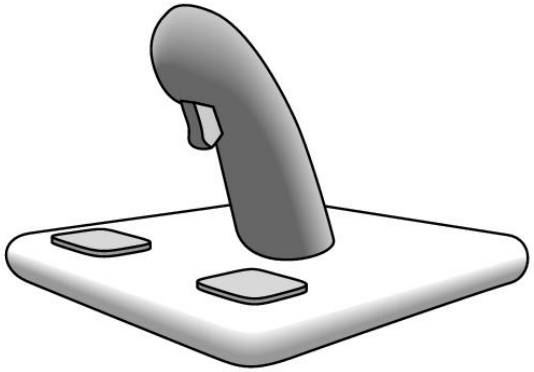
trackball



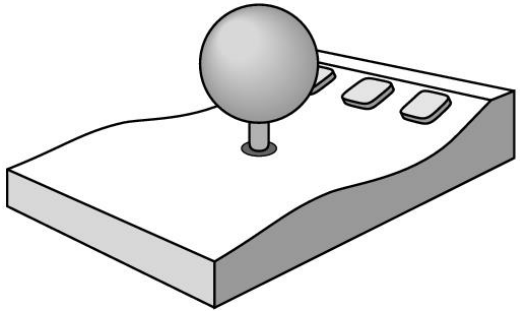
light pen



data tablet



joy stick



space ball

Physical Input Devices

- Physical input devices
  - Pointing devices
    - Allows the user to point to a location on the screen
    - In most cases, the user has more than one button to send a signal or interrupt to the computer.
    - Mouse, trackball, tablet, lightpen, joystick, spaceball
  - Keyboard devices
    - A device that returns a character code to a program
    - Keyboard

Relative Positioning Device

- 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 Input Devices

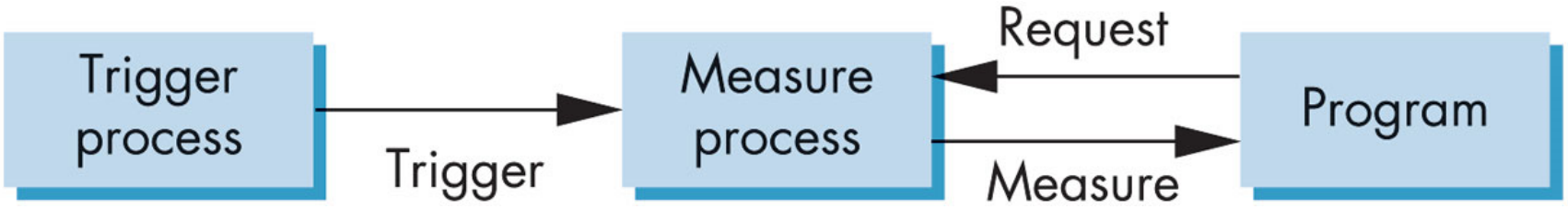
- String device - keyboard
  - Provide ASCII strings of characters to the program
- Locator device – mouse, trackball
  - Provide real world coordinate position to the program
- Pick device – mouse button, gun
  - Return the object's identifier(ID) to the program
- Choice device – widgets, function keys, mouse button
  - Let the user choose one of the options (menu)
- Valuators – slide bars, joystick, dial
  - Provide analog input (range of value) to the program
- Stroke – mouse drag
  - Return array of positions

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

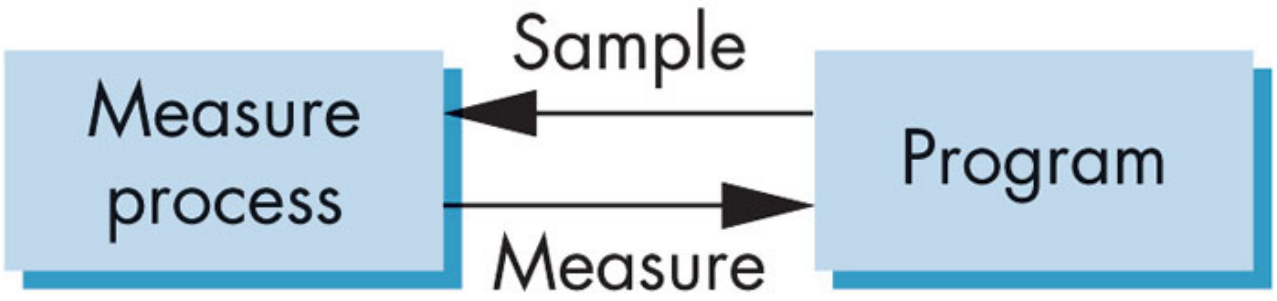
Request Mode

- In request mode, input measurement are not returned to the program until the user triggers the device.
- Standard for typical non-GUI program requiring character input
  - For example, when the C program's scanf function is used, the program stops while waiting for the terminal to type a character. Then, you can type and edit until you hit the enter-key(trigger).



Sample Mode

- Sample mode provides immediate input measures. As soon as the program encounters a function call, the measurement is returned. Therefore, no trigger is required.
- Example: getc function in C program



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.
- Use the callback function for a specific event.

Unity Input Class

- Input.GetAxis("Mouse X"|"Mouse Y") – mouse
- Input.GetAxis("Horizontal"|"Vertical") – joystick, WASD and arrow keys
  - moveAmount = Input.GetAxis("Vertical") * speed
  - turnAmount = Input.GetAxis("Horizontal") * rotSpeed
- Input.GetButtonDown("Fire1"|"Fire2"|"Fire3") – action-like events only
- Input.GetMouseButtonDown(0|1|2) – mouse button
  - Vector3 mousePos = Input.mousePosition
- Input.GetKey(KeyCode.UpArrow|"up") – holds down key
- Input.GetKeyDown(KeyCode.Space|"space")
- Input.GetTouch(0|...|Input.touchCount)
  - Vector2 touchDeltaPos = Input.GetTouch(0).deltaPosition

https://docs.unity3d.com/ScriptReference/Input.html

Keyboard Functions

- static bool GetAxis(string axisName)
  - Returns the value of the virtual axis identified by axisName.
- static bool GetKey(KeyCode key)
- static bool GetKey(string name)
  - Returns true while the user holds down the key
- static bool GetKeyDown(KeyCode key)
- static bool GetKeyDown(string name)
  - Returns true during the frame the user starts pressing down the key
- static bool GetKeyUp(KeyCode key)
- static bool GetKeyUp(string name)
  - Returns true during the frame the user releases the key

Keyboard Event Callback

- Call this function from the **Update()** function, since the state gets reset each frame.

```
public class Example : MonoBehavior {
    void Update()  {
        // The value is in the range -1 to 1
        float translation = Input.GetAxis("Vertical") * speed;
        float rotation = Input.GetAxis("Horizontal") * rotSpeed;

        // ESC-key exits the program
        if (Input.GetKeyDown(KeyCode.Escape)) {
            Application.Quit();
        }
    }
}
```

Mouse Functions

- static bool GetAxis(string axisName)
  - Returns the value of the virtual axis identified by axisName.
- static bool GetMouseButton(int button)
  - Returns whether the given mouse button is held down.
- static bool GetMouseButtonDown(int button)
  - Returns true during the frame the user pressed the given mouse button.
- static bool GetMouseButtonUp(int button)
  - Returns true during the frame the user releases the given mouse button.
- static Vector3 mousePosition
  - The current mouse position in pixel coordinates (read only)
- static Vector2 mouseScrollDelta
  - The mouse scroll delta (read only) -1~0~1

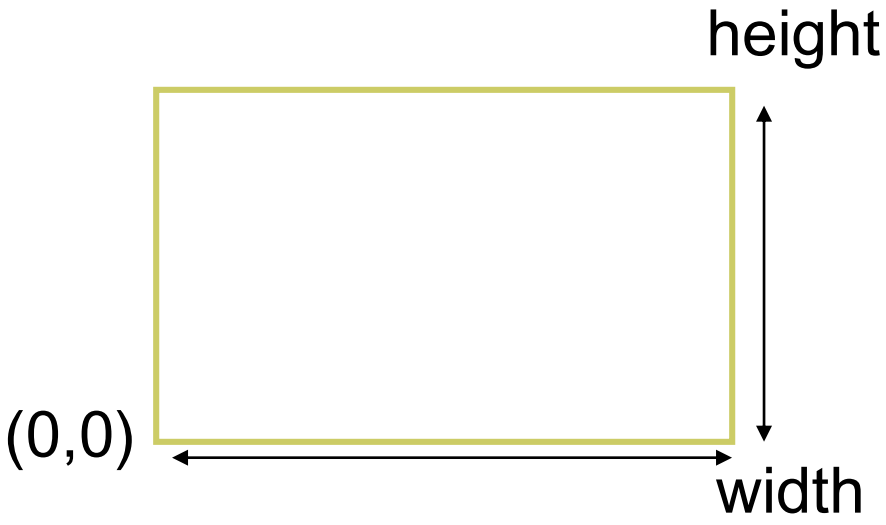
Mouse Event Callback

- Call this function from the **Update()** function, since the state gets reset each frame.

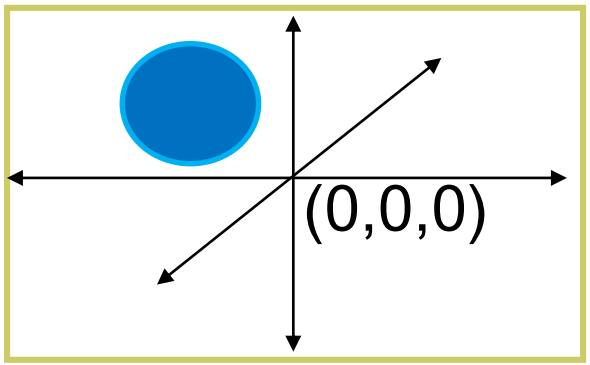
```
public class Example : MonoBehavior {
    void Update()  {
        // The value is in the range -1 to 1
        float h = Input.GetAxis("Mouse X") * rotSpeed;
        float v = Input.GetAxis("Mouse Y") * speed;

        // left-mouse holds to print the mouse position
        if (Input.GetMouseButton(0)) {
            Debug.Log(Input.mousePosition);
        }
    }
}
```

Mouse Positioning

- In Unity, the screen coordinate has the origin at the bottom-left corner, x+ is increasing to the right, y+ is increasing upwards.



2D screen coordinates



3D world space coordinates

Mouse Positioning

```
Vector3 worldPosition
// 2D mouse position -> 3D world position
void Update()
{
    Vector3 mousePos = Input.mousePosition; // Screen Space
    mousePos.z = Camera.main.nearClipPlane;
    worldPosition = Camera.main.ScreenToWorldPoint(mousePos);
}
```

Mouse Positioning

```
void OnGUI() {
    Vector3 point = new Vector3();
    Event   currentEvent = Event.current;
    Vector3 mousePos = new Vector3();

    // Get the mouse position from Event.
    // Note that the y position from Event is inverted. (GUI Space) –> 2D screen space
    mousePos.x = currentEvent.mousePosition.x;
    mousePos.y = Camera.main.pixelHeight - currentEvent.mousePosition.y;
    mousePos.z = Camera.main.nearClipPlane;
    point = Camera.main.ScreenToWorldPoint(mousePos); // 2D screen -> 3D world

    GUILayout.BeginArea(new Rect(20, 20, 250, 120));
    GUILayout.Label("Screen pixels: " + Camera.main.pixelWidth + ":" +
    Camera.main.pixelHeight);
    GUILayout.Label("Mouse position: " + mousePos);
    GUILayout.Label("World position: " + point.ToString("F3"));
    GUILayout.EndArea();
}
```