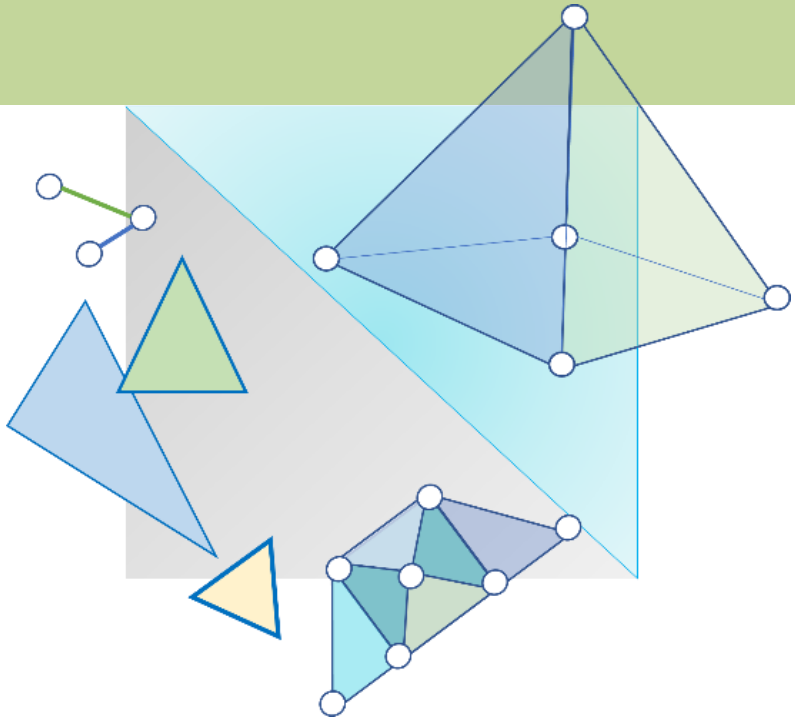
CITS3003 Graphics & Animation

Lecture 10:
Input, Interaction & Callbacks



Content

- Basic input devices
  - Physical Devices
  - Logical Devices
  - Input Modes
- Event-driven input
- Introduction to callbacks

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

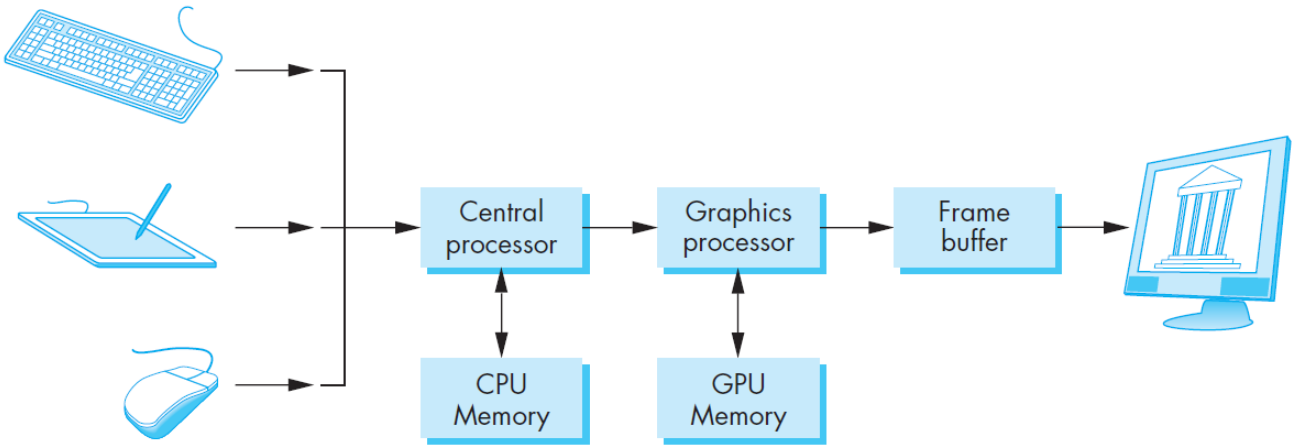
https://bimaplus.org/news/the-very-beginning-of-the-digital-representation-ivan-sutherland-sketchpad/

Ivan Sutherland's sketchpad demo: link

A Graphics System

There are six major elements in a graphics system

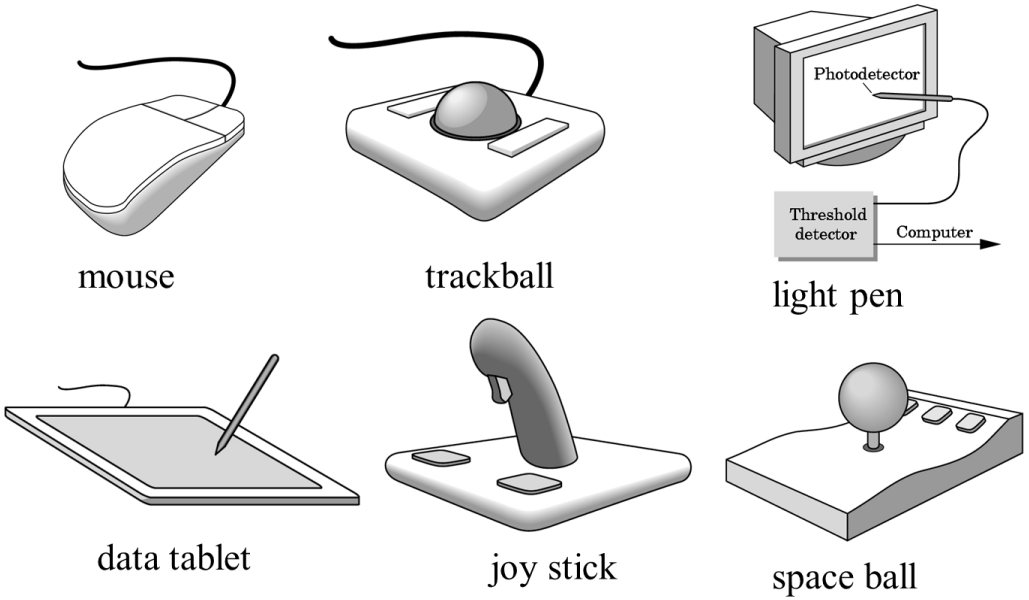
- Input devices
- Central Processing Unit
- Graphics Processing Unit
- Memory
- Frame Buffer
- Output Devices



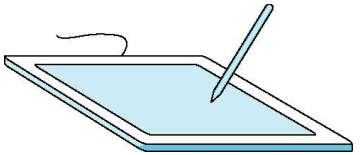
A Graphics System

Input Devices

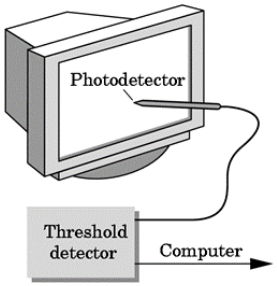
- Input devices can be viewed as
  - **Physical devices**:
    - **Pointing devices**: Allow user to indicate a position on the screen, e.g., mouse.
      - Incorporates one or more buttons
    - **Keyboard devices**: Return character codes (ASCII code), e.g., keyboard.



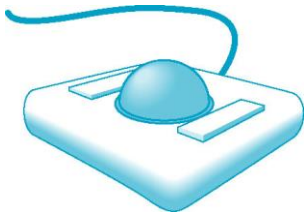
Pointing Devices

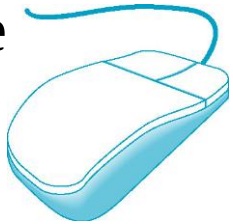


- **Absolute positioning**:
  - Devices such as the data tablet return a position directly to the operating system



- **Relative positioning:**
  - 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
      - Roll of trackball





Input Devices

- Input devices can also be viewed as
  - **Logical devices**: characterized by how they influence the application program, e.g., what is returned to the program via the API
    - An (x, y) position on the screen?
    - An object identifier for a menu item chosen?

    Both are performed by the same physical device (the mouse, in this case) but what is returned to the program is different.

Logical Input Devices

- Consider the C and C++ code
  - C++: **cin >> x;** \\ data in
  - C: **scanf(”%d”, &x);** \\data out
- What is the input physical device?
  - Can’t tell from the code
  - Could be keyboard, file, output from another program
  - A string is returned to the program regardless of the physical device

Logical Input Devices (cont.)

- Graphics APIs define different types of logical devices:
  - Locator: a device provides a position in world coordinates to the user program.
    - Usually implemented via pointing device e.g., mouse
  - Choice: a device that allows the user to select one of a discrete number of options, e.g., a menu item
    - OpenGL can use widgets provided by the windowing systems such as menus, and scroll bars
  - String: a device that provides ASCII strings to the user program, e.g., via key presses
    - String might be provided by a keyboard, or a file, or by pointing devices.

Logical Input Devices (cont.)

- Stroke: a device that returns an array of locations.
  - pushing down a mouse button, starts the transfer of data into the specified array and then releasing of the button, ends the transfer
- Valuator: a device that returns analog input to the user program, e.g., a widget such as a slidebar,
  - Widget is a graphical interface device
- Pick: a device that returns the identifier of an object on the displayto the user program.

- OpenGL and GLUT provide functions to handle all these

Input Modes

How physical and logical devices provide an input to the application program, can be described in terms of two entities:

- **Measure** – what is returned
- **Trigger** – when it is returned

- Input devices produce 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
  - Locator returns position information
  - Keyboard returns ASCII code (string)

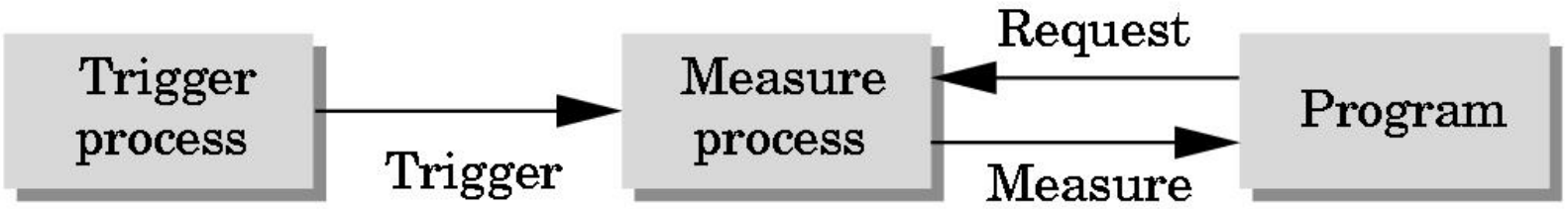
Input Modes

- We can obtain the measure of a device in three distinct modes.
  - *Request mode*, *sample mode* and *event mode*.

- Each mode is defined by the relationship between the measure process and the trigger

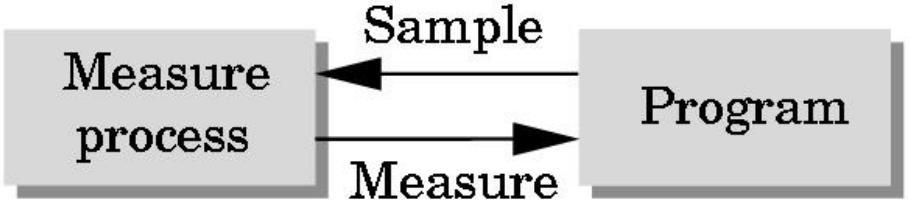
Request Mode

- For request mode input, the measure is provided to program only when user triggers the device
  - This mode is standard in non-graphical applications such as typical C programs
- A typical example is *keyboard input*:
  - The application program request a keyboard input from the user
  - The user can type, erase (backspace), correct.The application program hangs there until the *enter* (return) key (the trigger) is depressed



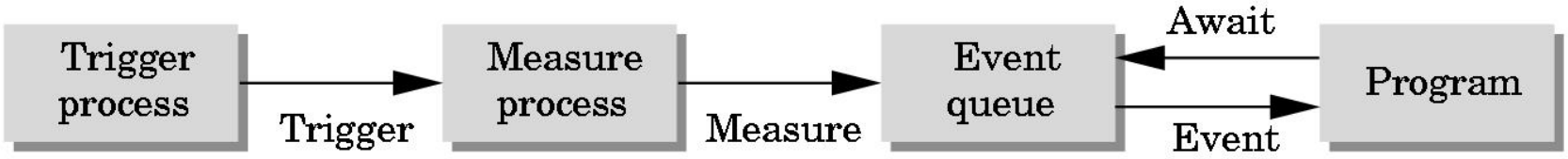
Sample Mode

- It is an immediate mode (samples the input when the function call is made)
- Expects the measure to be present already at the sampling time (call time)
  - Both request mode and sample mode are useful for situations where the program guides the user but are not useful in applications where the user controls the flow of the program.
  - For sample and request mode, user must identify which device will provide the input

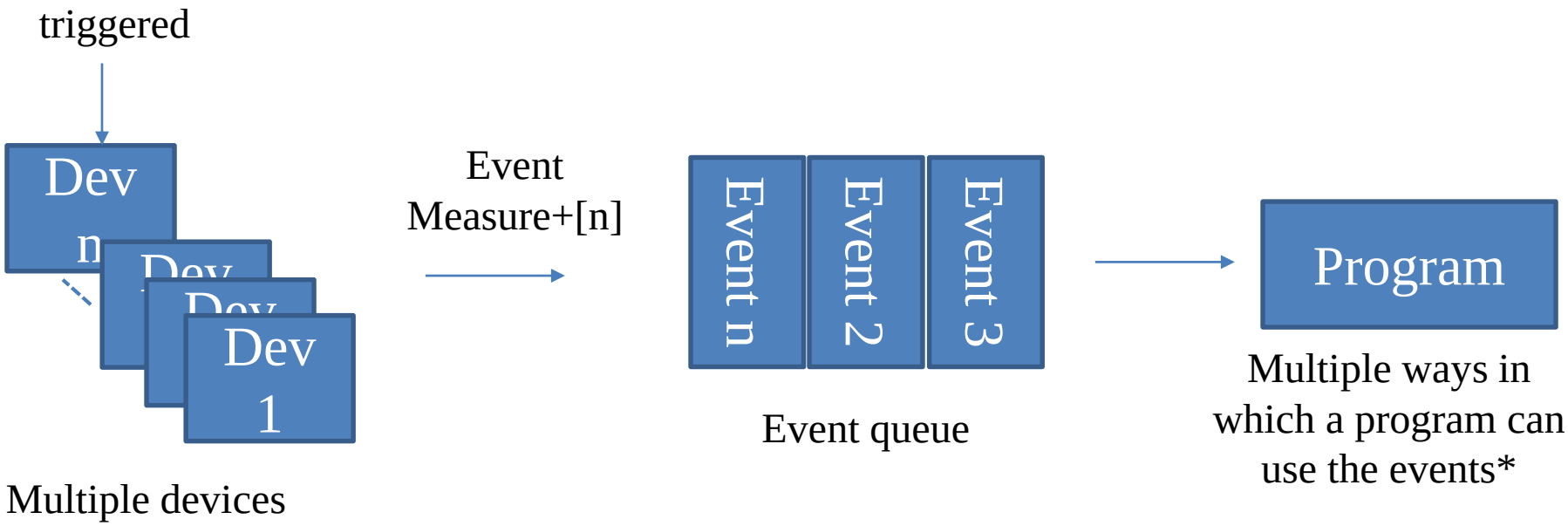


Event Mode

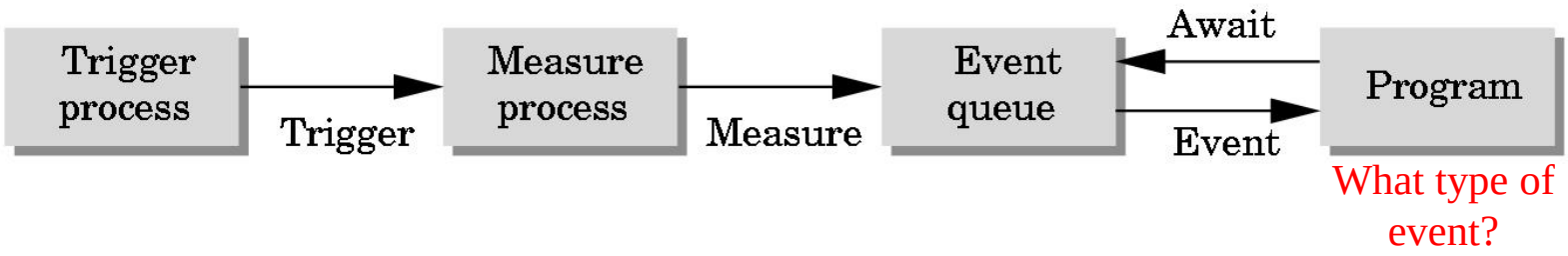
- Request and sample modes are **not suitable** for programs that need **to interact** with the user.
- Each time a device is triggered, an **event** is generated.
- Device measure is placed on an **event queue**



Event Mode (cont.)



*Option 1



*Option 2 associate a **callback function** with a specific type of event.

- The OS queries the event queue regularly, and execute the callbacks corresponding to the events in the queue

Event Categories

- Window event: resize, expose, iconify
- Mouse event: click one or more buttons
- Motion event: this refers to the *mouse move* event (when the cursor is inside the window of the application program)
- Keyboard: press or release a key
- Idle: non-event
  - Define what should be done if no other event is in the *event queue*

Callbacks

- We can define a ***callback*** *function* for each type of event that the graphics system recognizes
- This user-supplied function is executed when the event occurs
- GLUT example: **glutMouseFunc(mymouse)**

mouse callback function

GLUT callbacks

- GLUT recognizes a subset of the events recognized by any particular window system (Windows, X, Macintosh)
- Examples of the functions that set the callbacks
  - **glutDisplayFunc** // called when window needs to be redrawn
  - **glutCreateMenu //** called when menu item was selected
  - **glutMouseFunc** // called when mouse button pressed
  - **glutReshapeFunc** // called when window reshapes
  - **glutKeyboardFunc** // called when a key is struck
  - **glutIdleFunc** // called when there are no events
  - **glutMotionFunc** // called when mouse moves, and button held
  - **glutPassiveMotionFunc** // called when mouse moves

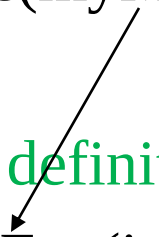
glutMouseFunc

% usage

void glutMouseFunc(void (*func)(int button, int state, int x, int y));

Example:

glutMouseFunc(myMouseFun);



% the function definition

void myMouseFun(int button, int state, int x, int y)

{

% this is where you write code for what you want to do

% when a mouse “event” happens

}

GLUT Event Loop

- Recall that the last line in **main.c** for a program using GLUT must be

  glutMainLoop();

  which puts the program in an infinite event loop
- In each pass through the event loop, GLUT
  1. looks at the events in the queue
  2. for each event in the queue, GLUT executes the appropriate callback function if one is defined
  3. if no callback is registered for the event, the event is ignored

The *display* callback

- The ***display*** callback is executed whenever GLUT determines that the window should be refreshed, for example
  - When the window is first opened
  - When the window is reshaped
  - When a window is exposed
  - When the user program decides it wants to change the display
- In **main()**
  - **glutDisplayFunc(mydisplay)** identifies the function to be executed
  - Every GLUT program must have a display callback

Posting redisplays

- Different events may need to invoke the display callback function
- We use **glutPostRedisplay();** to mark that the current window needs to be redisplayed, which sets a flag.
- GLUT checks to see if the flag is set at the end of the event loop
- If set then the display callback function is executed

Further Reading

"Interactive Computer Graphics – A Top-Down Approach with Shader-Based OpenGL" by Edward Angel and Dave Shreiner, 6th Ed, 2012

- Sec. 1.2.4-1.2.7 *Input Devices*, *Physical Input Devices*, *Logical Devices*, *Input Modes*
- Sec. 2.1 The Sierpinski Gasket; immediate mode graphics vs retained mode graphics
- C++ code in the Chapter02 to Chapter04 folders