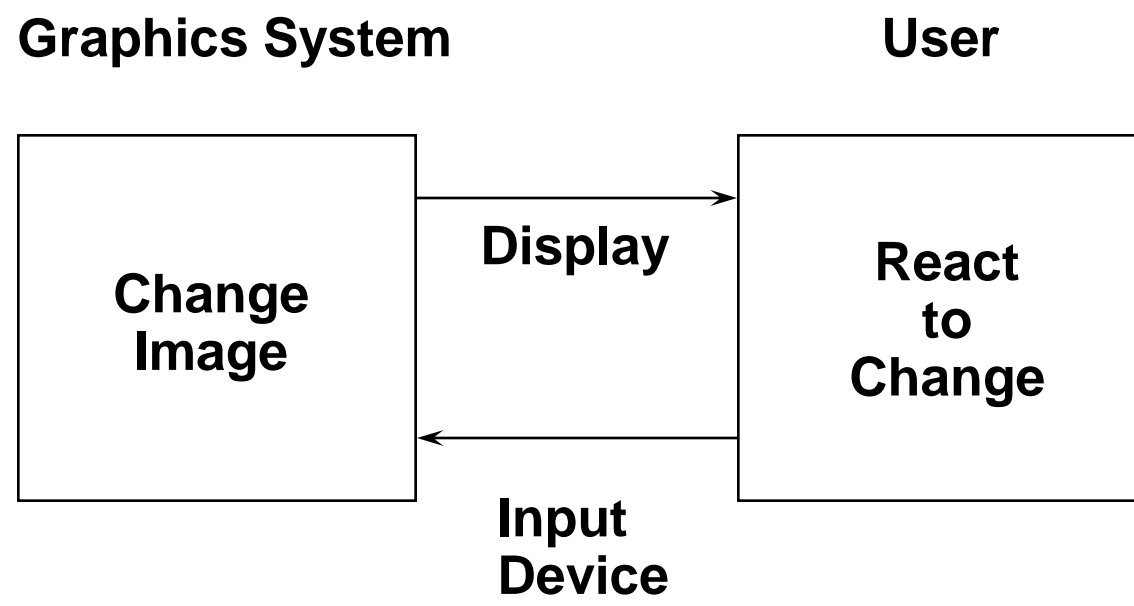
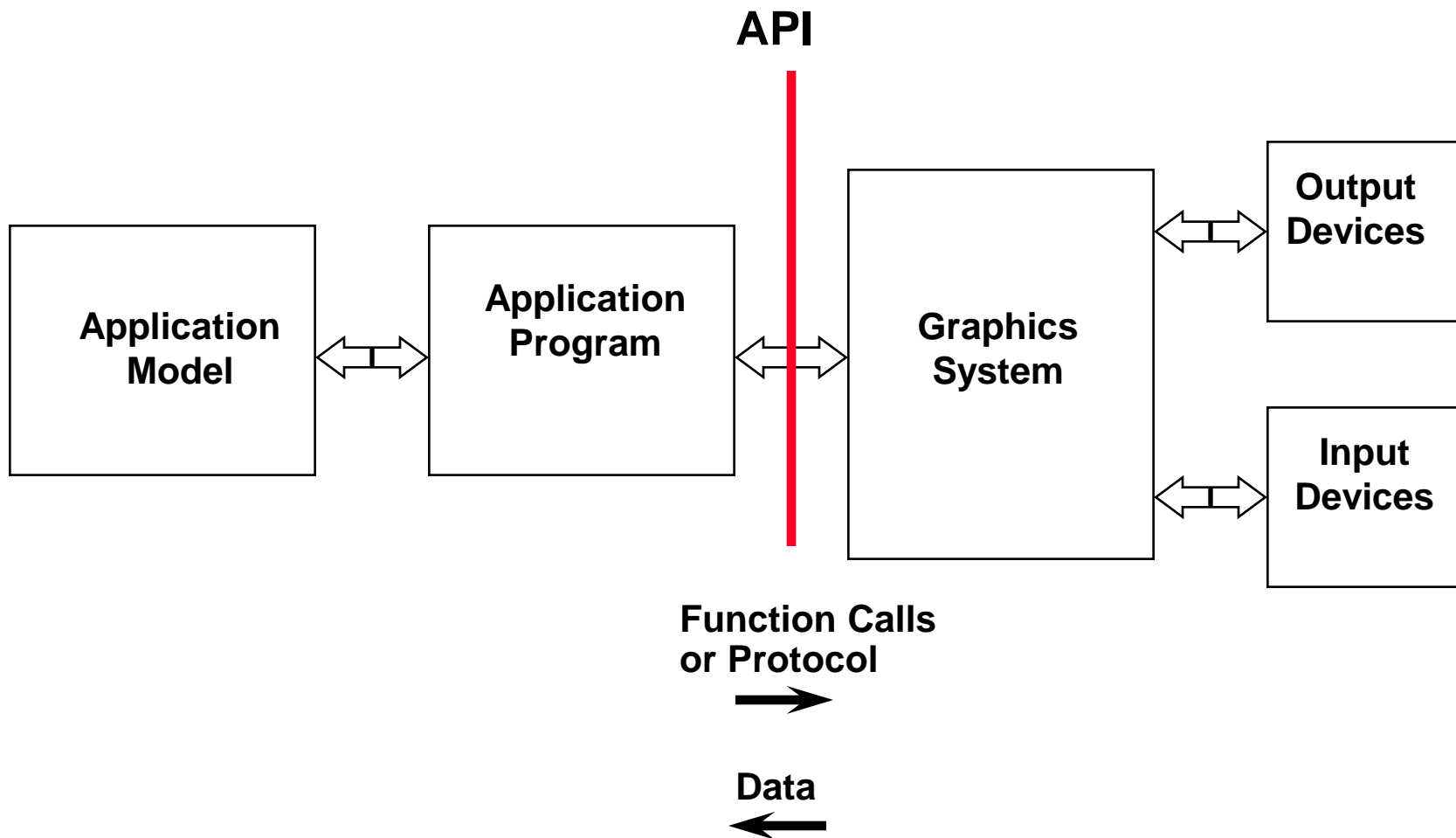


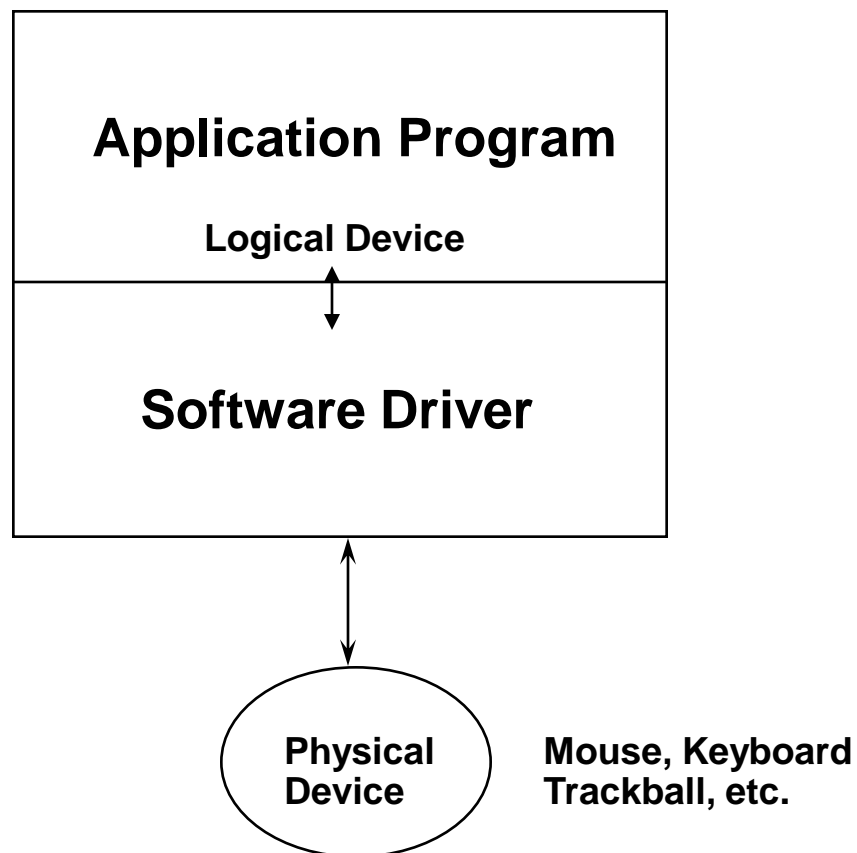
Interaction with Graphics System



Computer Graphics Conceptual Model



Physical vs. Logical Input Devices



Logical Input Devices

- **String**
 - Provides ASCII strings to user program
 - Usually implemented as physical keyboard
- **Locator**
 - Inputs position in defined coordinate system
 - Absolute vs. relative
 - Direct vs. indirect
- **Choice**
 - Choice among a discrete number of options
 - Implemented by menus or control button widget
- **Valuator (or Dial)**
 - Provides analog input to user program
 - Bounded vs. unbounded
 - Often implemented as sidebar widget

Logical Input Devices, contd.

- **Pick**
 - Returns identifier of an object to user program
 - Can be implemented by locator and choice

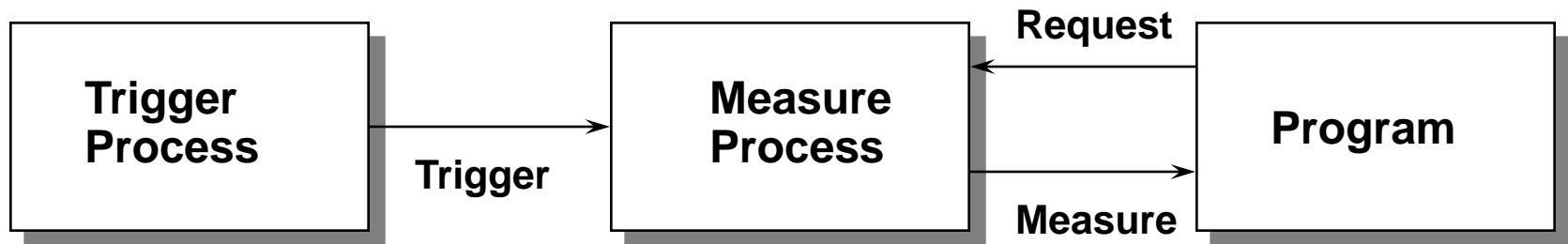
What is it?



Measure vs. Trigger

- **The measure of a device is the value returned to the user program.**
 - **String device: ASCII string**
 - **Locator: Location coordinates**
 - **Choice: Identifier of option chosen**
 - **etc.**
- **Trigger of a device is a physical signal that the user can use to signal the computer that the measure is available.**
 - **String device (keyboard): Return key**
 - **Locator (mouse): mouse button(s)**

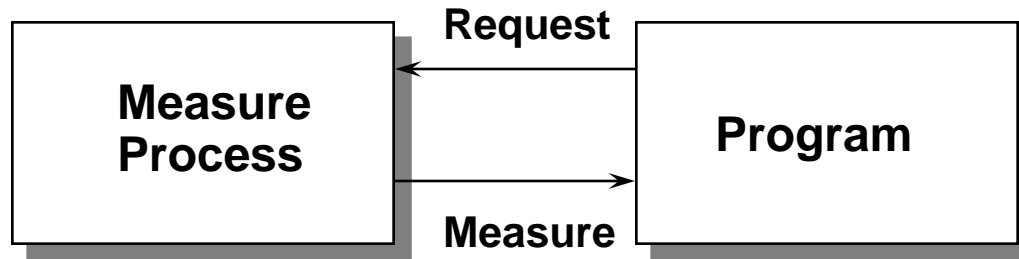
Request Mode Input



1. Program requests measure of a device and blocks.
2. Measure process maintains current measure.
3. Trigger process signals measure process to return measure.

```
request_locator(device_id, &measure);
```

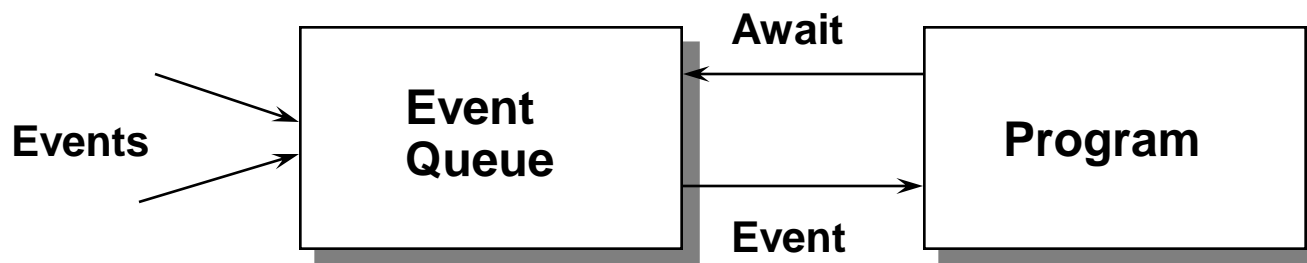
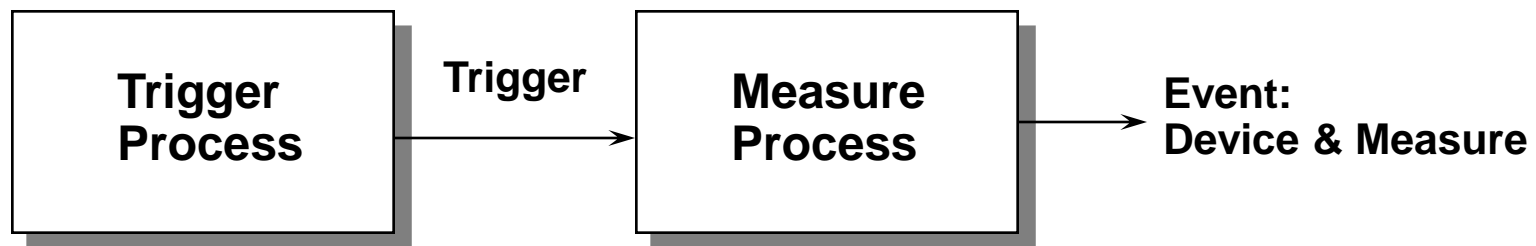
Sample Mode Input



1. Program requests measure of a device and blocks.
2. Measure process immediately returns current measure.

```
sample_locator(device_id, &measure);
```

Event Mode Input



Event Driven Interaction-Blocking

Initialize, including generating the initial image;

Activate input devices in event mode;

```
do {  
    wait for user-triggered event on any device;  
    switch (which device caused event)  
    {  
        case DEVICE_1: collect DEVICE_1 measure, process, respond;  
        case DEVICE_2: collect DEVICE_2 measure, process, respond;  
  
        ...  
    }  
}  
while (user does not request exit);
```

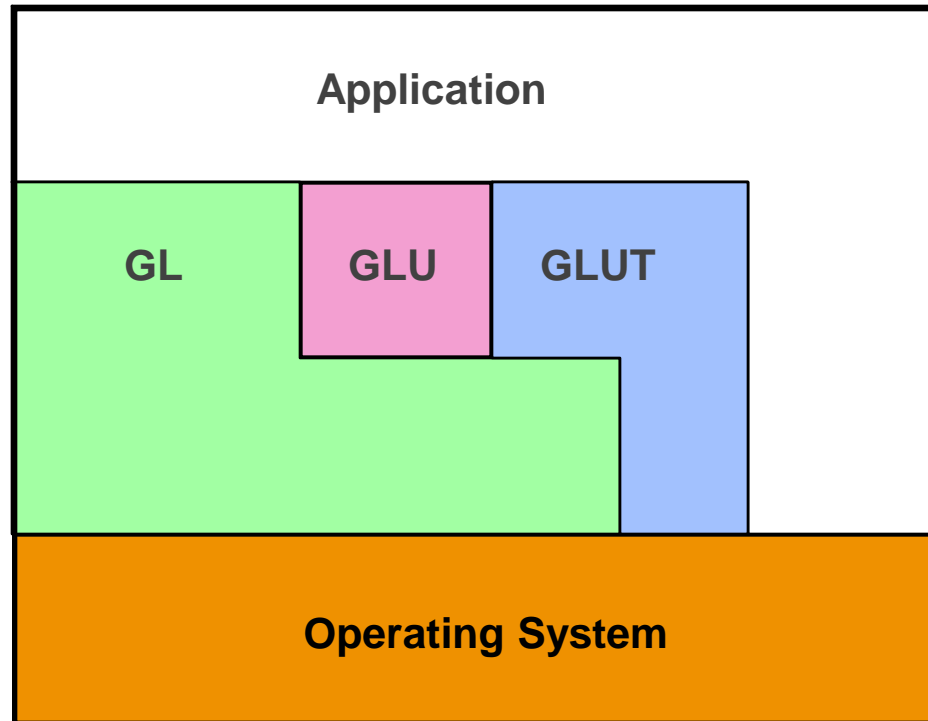
Event Driven Interaction-NonBlocking

Initialize, including generating the initial image;

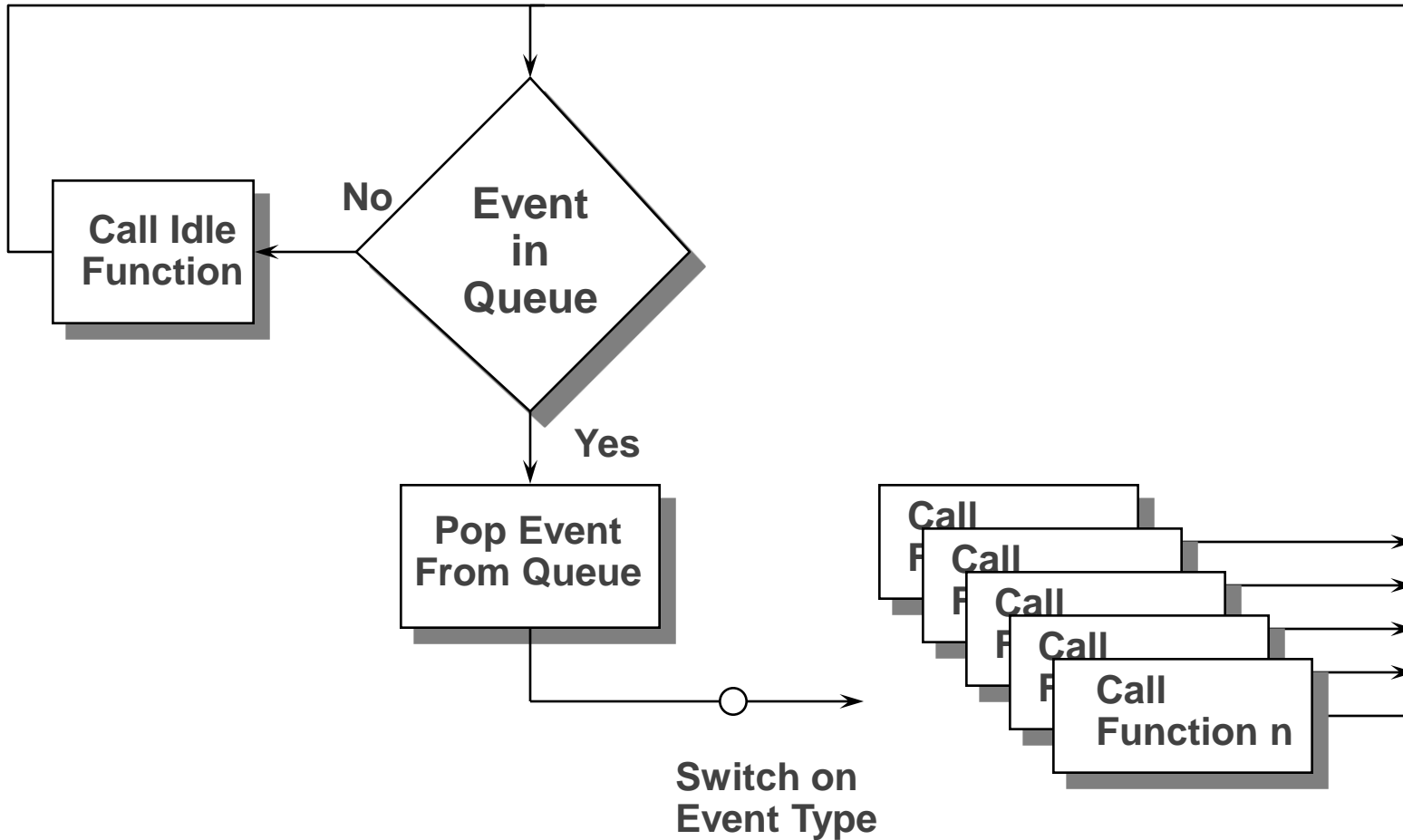
Activate input devices in event mode;

```
do {  
    if (there is an event on the event queue)  
        switch (which device caused event)  
        {  
            case DEVICE_1: collect DEVICE_1 data, process, respond;  
            case DEVICE_2: collect DEVICE_2 data, process, respond;  
  
            ...  
        }  
    Do background processing;  
}  
while (user does not request exit);
```

GL Library Organization



GLUT Event Processing



Callbacks

Event <u>Type</u>	“Registered” Function
Event A \rightarrow m_1, m_2	function_A(m_1, m_2)
Event B	function_B()
Event C	function_C()
Event D	
Event E	
Event F	
No Event	Idle function()

NB: Function parameters must match measures contained in associated event.

Display Event

Trigger: GLUT determines that the window needs to be redisplayed. A display event is generated when the window is first drawn.

Callback function form:

```
void display();
```

Registration:

```
glutDisplayFunc(display);
```

A display callback function must be registered for **each window.**

GLUT Mouse Event

Trigger: Any mouse button is depressed or released.

Callback function form:

```
void mouse_callback_func(int button, int state,  
                        int x, int y);
```

Registration:

```
glutMouseFunc(mouse_callback_function);
```

GLUT Defined Mouse Constants

```
GLUT_LEFT_BUTTON  
GLUT_MIDDLE_BUTTON  
GLUT_RIGHT_BUTTON
```

```
GLUT_UP  
GLUT_DOWN
```

Systems with only one mouse button can only generate a GLUT_LEFT_BUTTON callback.

GLUT Reshape Event

Trigger: Active window is resized

Callback function form:

```
void reshape_func(GLsizei w, GLsizei h);
```

Registration:

```
glutReshapeFunc(reshape_func);
```

GLUT Move Event

Trigger: The mouse is moved while one or more mouse buttons are pressed.

Callback function form:

```
void motion_func(int x, int y);
```

Registration:

```
glutMotionFunc(motion_func);
```

GLUT Keyboard Event

Trigger: Any key is depressed.

Callback function form:

```
void keyboard_function(unsigned char key,  
                      int x, int y);
```

Registration:

```
glutKeyboardFunc(keyboard_function);
```

Other Defined Events in GLUT

```
glutPassiveMotionFunc  
glutVisibilityFunc  
glutEntryFunc  
glutSpecialFunc  
glutSpaceballMotionFunc  
glutSpaceballRotateFunc  
glutSpaceballButtonFunc  
glutButtonBoxFunc  
glutDialsFunc  
glutTabletMotionFunc  
glutTabletButtonFunc  
glutMenuStatusFunc
```

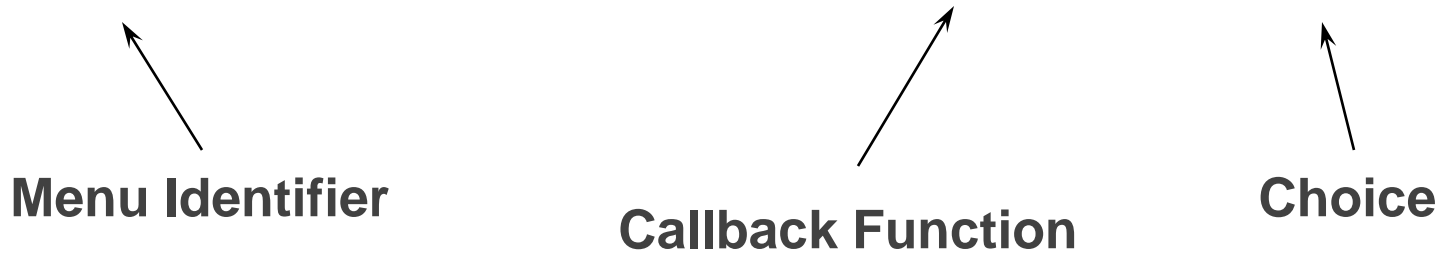
Implementing Choice: Menus in GLUT

- **Four steps:**

- **Create menu:** `glutCreateMenu(menu);`
- **Define menu entries:** `glutAddMenuEntry`
- **Attach menu to a mouse button:** `gluAttachMenu`
- **Define callback function:** `void menu(int id);`

Creating a Menu in GLUT

```
int glutCreateMenu(void (*func) (int value));
```



Creates a new pop-up menu.

Returns a unique integer identifier for the menu.

Takes as argument a pointer to a single callback function that takes an integer argument. The integer argument of the callback is mapped to the menu choice.

Sets the *current menu* to the newly created menu.

Associating a Menu with a Mouse Key

```
void glutAttachMenu(int button);
```

Associates the selected button with the *current* menu.

`button` **is selected from the GLUT defined button constants:**

```
GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON,  
GLUT_RIGHT_BUTTON
```

Adding Entries to the Menu

```
void glutAddMenuEntry(char *name, int value);
```



**String to appear
in menu entry.**



**Value to be passed
to callback function.**

Adds a menu entry to the bottom of the *current menu*.

Building a Sub-menu

```
void glutAddSubMenu(char *name, int menu);
```

ASCII string to display in the menu item from which to cascade sub-menu.



Identifier of menu to cascade from this sub-menu item.



Example User Interface Program

- **Open a window and clear it to black**
- **Provide a menu entry to select between drawing a square or a triangle**
- **Provide a menu entry to select the color of the drawn object.**
- **Use the mouse to select the position of the object**
- **Use the left mouse button to draw the object.**
- **Provide a menu item to clear the screen**
- **Provide a menu item to exit the program**
- **Clear the screen upon resizing the window**