Hochiminh city University of Technology
Faculty of Computer Science and Engineering

COMPUTER GRAPHICS

CHAPTER 04:

Input & Interaction

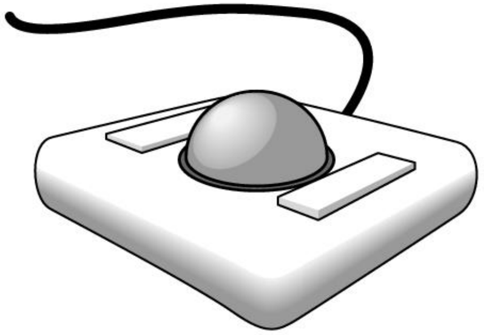
OUTLINE

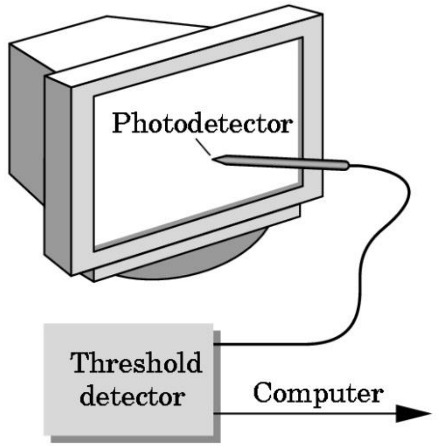
- ❑ Basic Input Devices
- ❑ Window-based Programming
- ❑ Keyboard Event
- ❑ Mouse Event
- ❑ Reshape Event
- ❑ Idle Event

Basic input devices

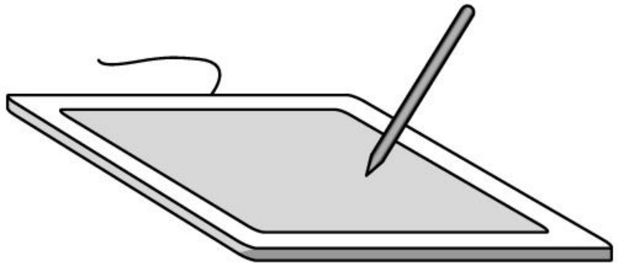
❑ Physical Devices

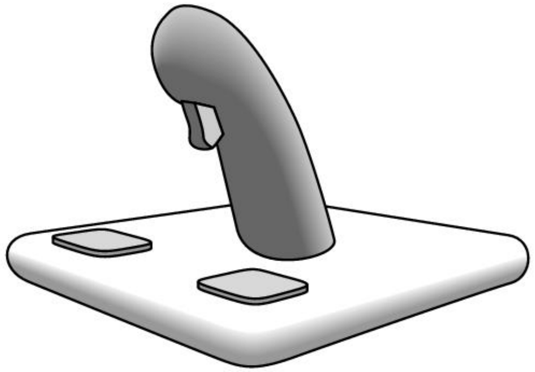
mouse

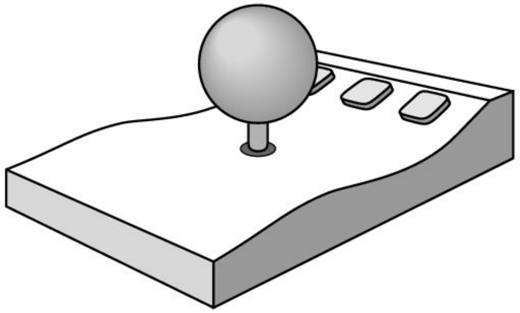

trackball



light pen

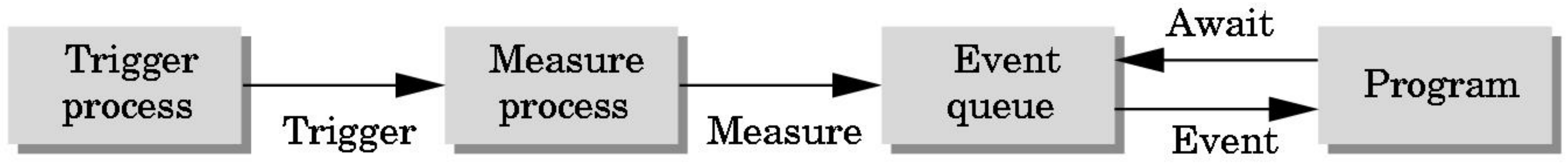

data tablet


joy stick


space ball

Windows-based programming

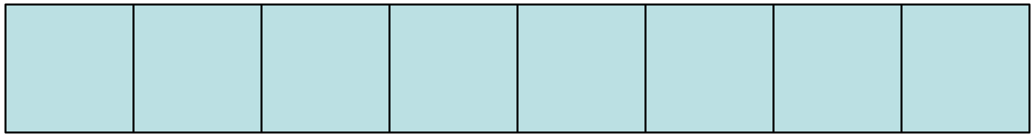
- Event Mode
  - Most systems have more than one input device, each of which can be triggered at an arbitrary time by a user
  - Each trigger generates an *event* whose measure is put in an *event queue* which can be examined by the user program

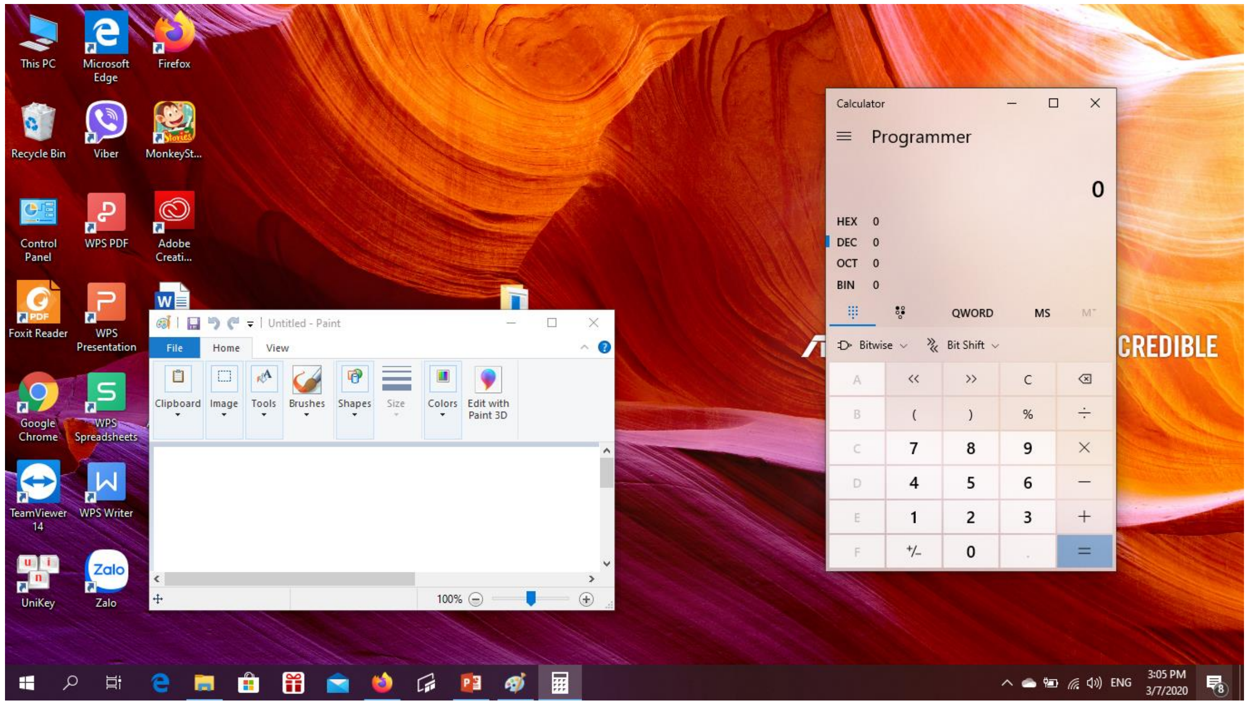


Windows-based programming

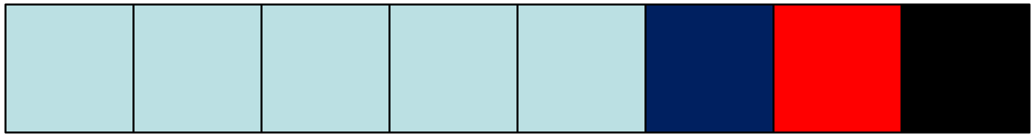
- Event-driven programming
- Event queue
- Callback function
- Register callback function
    - glutDisplayFunc(myDisplay)
    - glutReshapeFunc(myReshape)
    - glutMouseFunc(myMouse)
    - glutKeyboardFunc(myKeyboard)

Windows-based programming





Windows-based programming



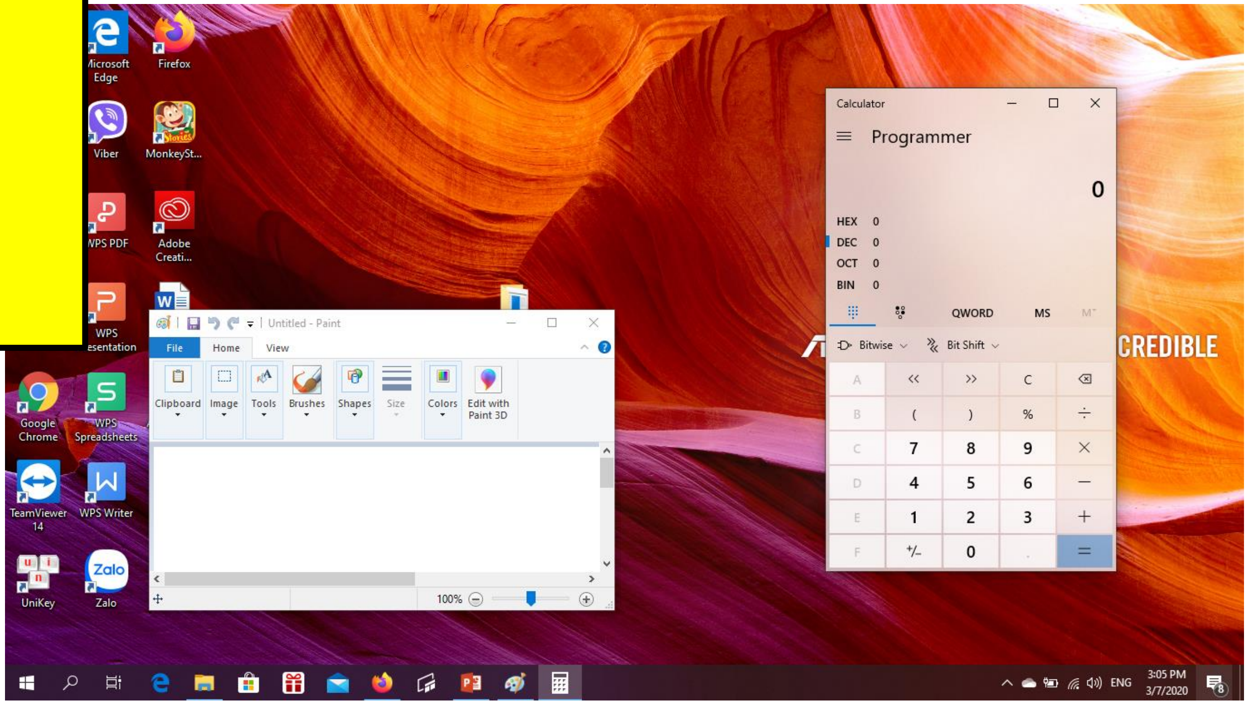
ID:….

Event:…

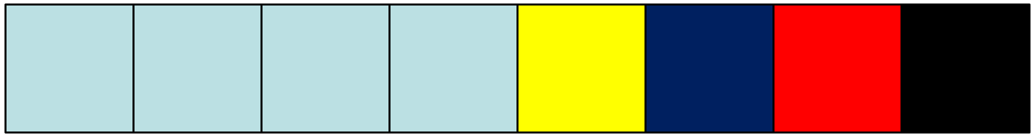
X: …

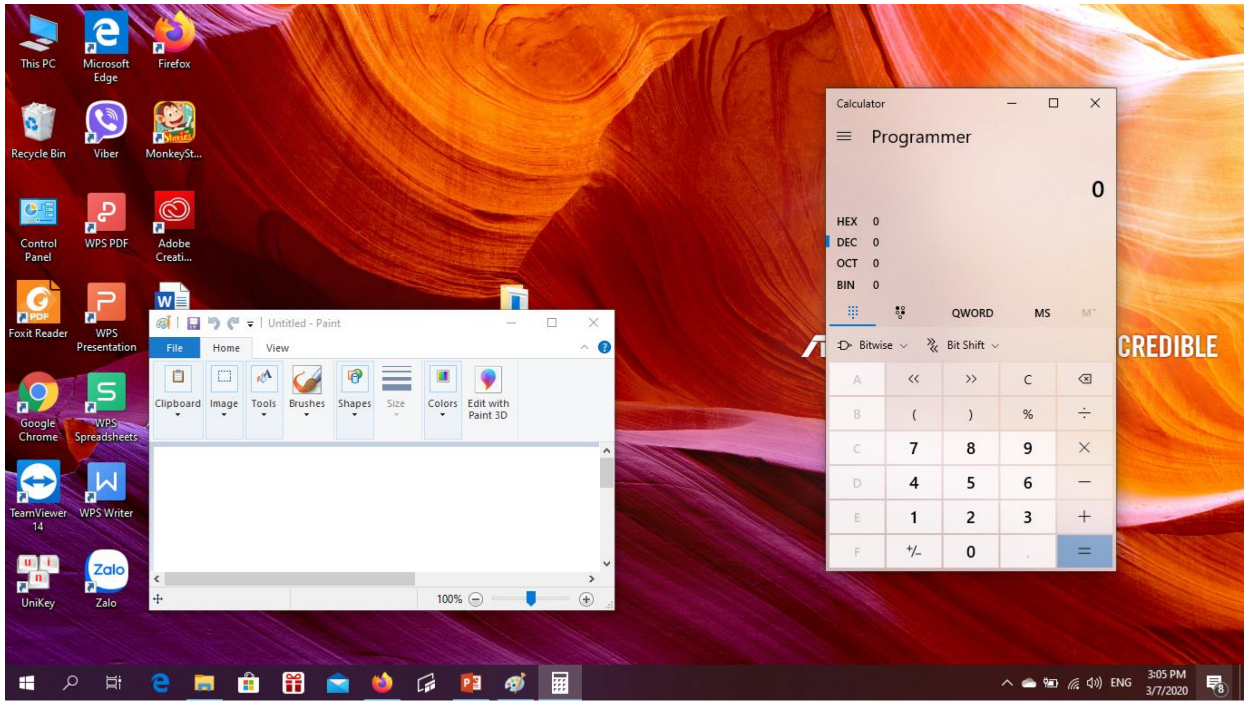
Y: …

……



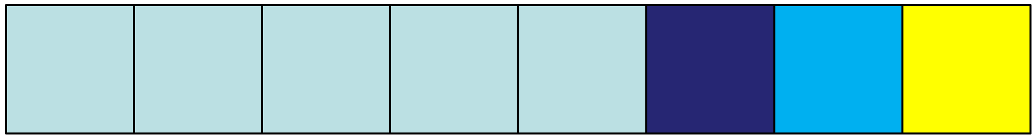
Windows-based programming



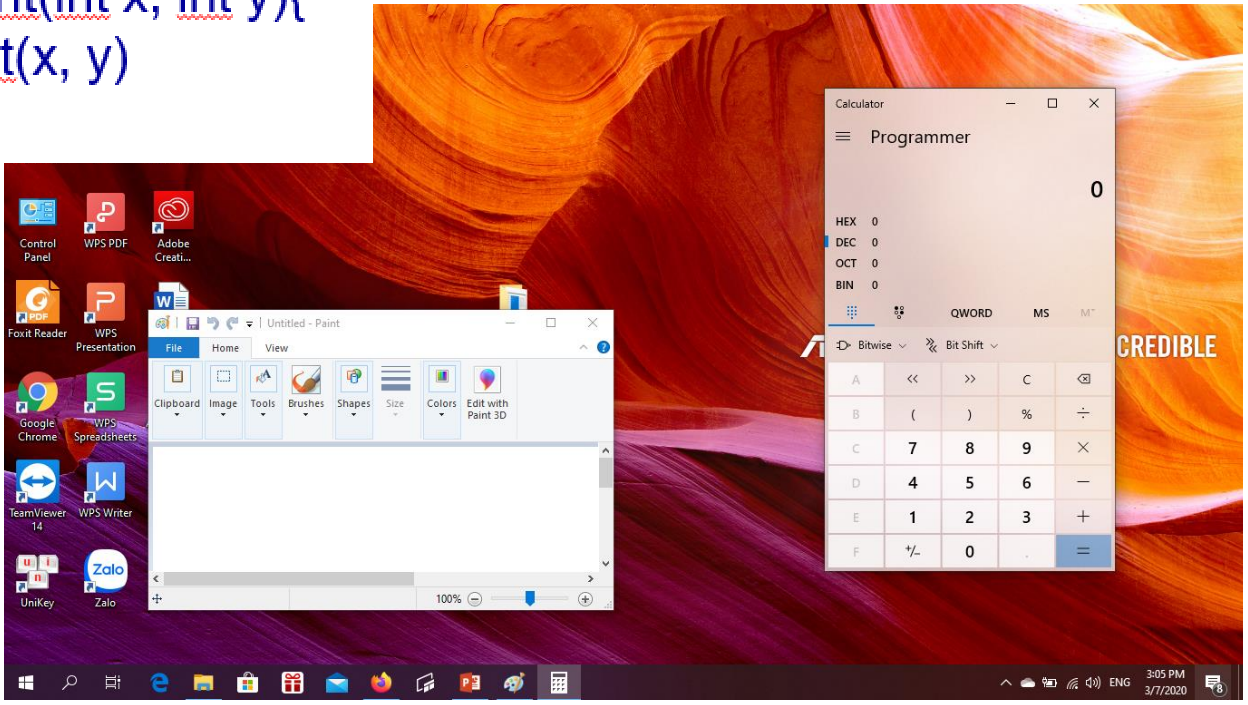


Windows-based programming

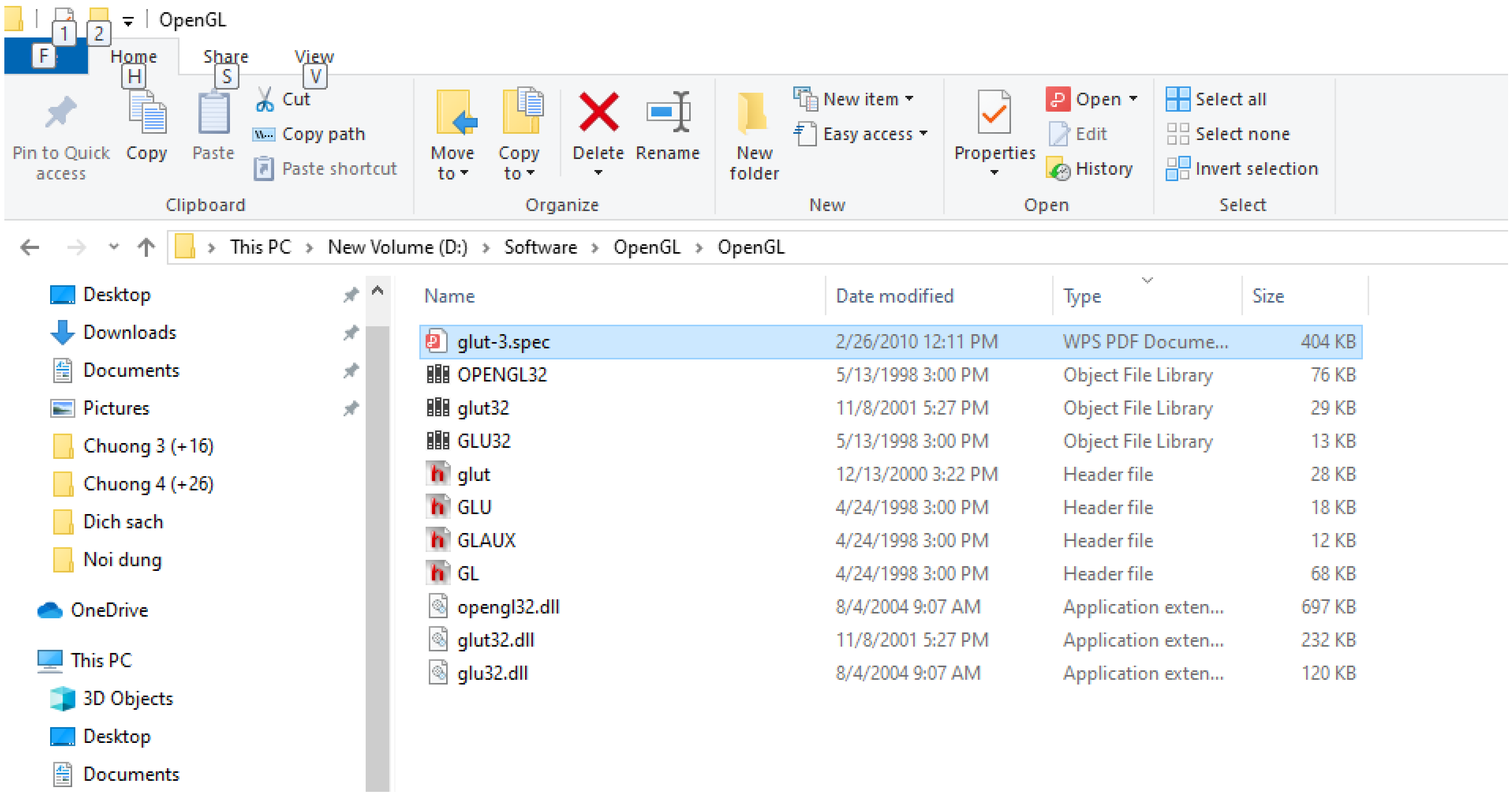
Event Queue



```
void drawPoint(int x, int y){
    DrawPoint(x, y)
}
```

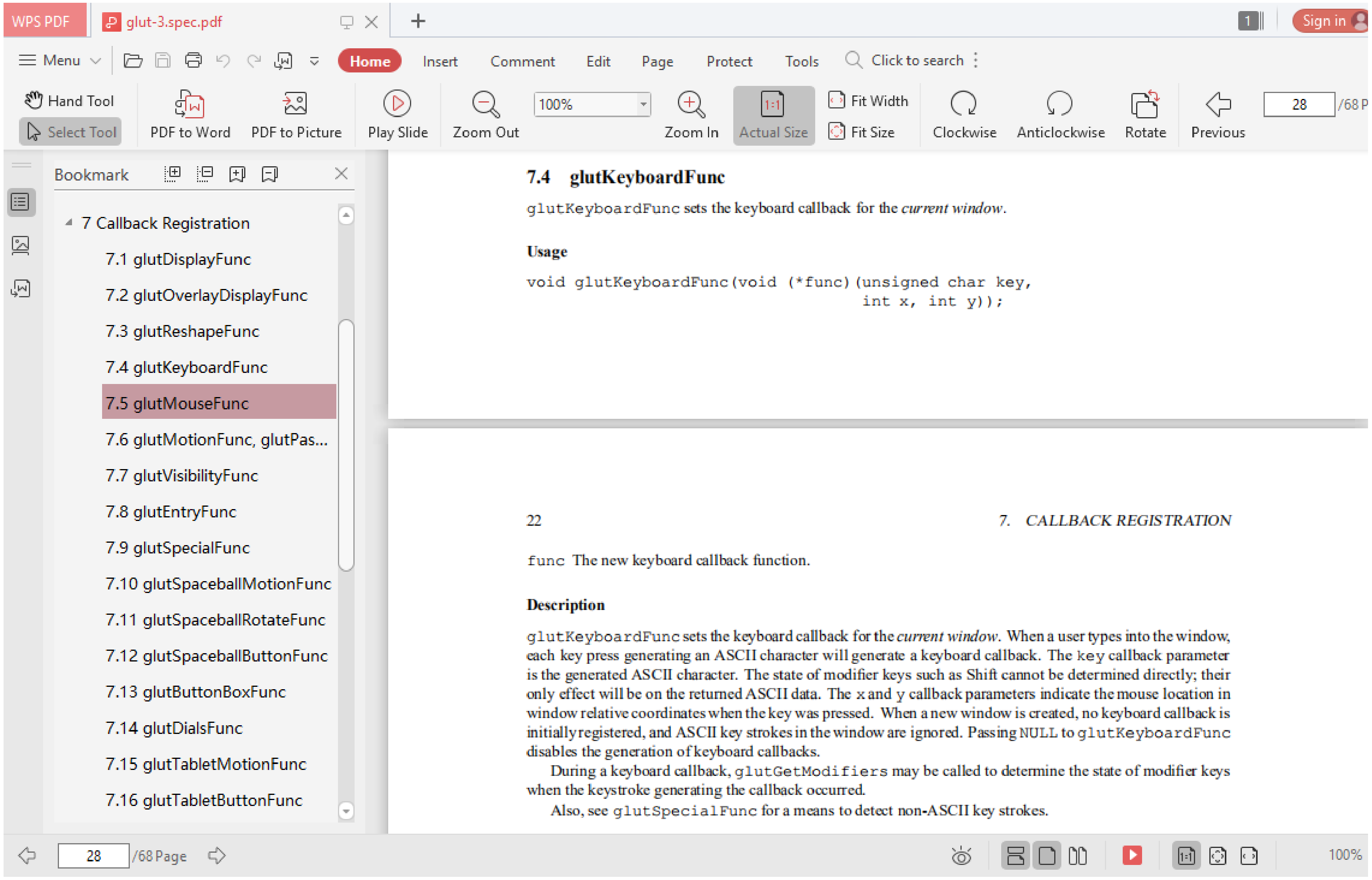


Windows-based programming



Name	Date modified	Type	Size
glut-3.spec	2/26/2010 12:11 PM	WPS PDF Docume...	404 KB
OPENGL32	5/13/1998 3:00 PM	Object File Library	76 KB
glut32	11/8/2001 5:27 PM	Object File Library	29 KB
GLU32	5/13/1998 3:00 PM	Object File Library	13 KB
glut	12/13/2000 3:22 PM	Header file	28 KB
GLU	4/24/1998 3:00 PM	Header file	18 KB
GLAUX	4/24/1998 3:00 PM	Header file	12 KB
GL	4/24/1998 3:00 PM	Header file	68 KB
opengl32.dll	8/4/2004 9:07 AM	Application exten...	697 KB
glut32.dll	11/8/2001 5:27 PM	Application exten...	232 KB
glu32.dll	8/4/2004 9:07 AM	Application exten...	120 KB

Windows-based programming



7.4 glutKeyboardFunc

`glutKeyboardFunc` sets the keyboard callback for the *current window*.

Usage

```
void glutKeyboardFunc(void (*func)(unsigned char key,
                                    int x, int y));
```

22   *7. CALLBACK REGISTRATION*

`func` The new keyboard callback function.

Description

`glutKeyboardFunc` sets the keyboard callback for the *current window*. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The `key` callback parameter is the generated ASCII character. The state of modifier keys such as Shift cannot be determined directly; their only effect will be on the returned ASCII data. The `x` and `y` callback parameters indicate the mouse location in window relative coordinates when the key was pressed. When a new window is created, no keyboard callback is initially registered, and ASCII key strokes in the window are ignored. Passing `NULL` to `glutKeyboardFunc` disables the generation of keyboard callbacks.

During a keyboard callback, `glutGetModifiers` may be called to determine the state of modifier keys when the keystroke generating the callback occurred.

Also, see `glutSpecialFunc` for a means to detect non-ASCII key strokes.

Keyboard Event

- Normal Key

  - Callback function Prototype:

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

  - Register callback function

```
glutKeyboardFunc(myKeyboard);
```

  - Example

```
void myKeyboard(unsigned char key, int x, int y){
     if(key == ‘Q’ | key == ‘q’)
               exit(0);
}
```

Keyboard Event

- Function Key

  - Callback function Prototype:

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

  - Register callback function

```
glutSpecialFunc(myKeyboard);
```

  - Example

```
void myKeyboard(unsigned char key, int x, int y){
    if (key == GLUT_KEY_LEFT)            angle = angle + 5;
    else if (key == GLUT_KEY_RIGHT)      angle = angle - 5;
    glutPostRedisplay();
}
```

Mouse Event

- Mouse Click
  - Callback function Prototype:

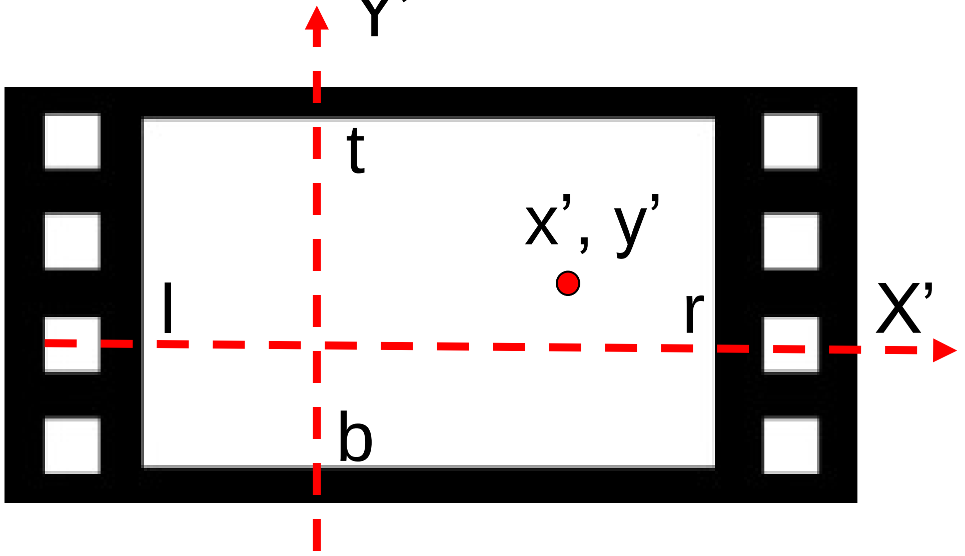
    void myMouse(int button, int state, int x, int y);

    - **which button (**GLUT_LEFT_BUTTON**,** GLUT_MIDDLE_BUTTON**,** GLUT_RIGHT_BUTTON**) caused event**
    - **state of that button (**GLUT_UP**,** GLUT_DOWN**)**
    - **Mouse Position in window**
  - Register callback function

    glutMouseFunc(myMouse);

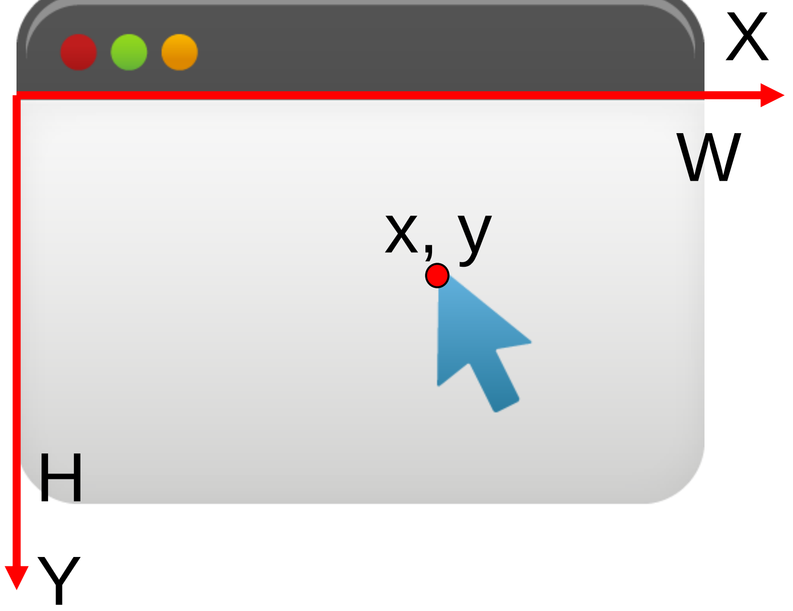
Mouse Event

- Mouse Click



glOrtho(l, r, b, t, -1, 1);

glutInitWindowSize(W, H);



$x' = A.x + B$

$r = A.W + B$ $\quad x' = ((r-l)/W).x + l$ $\qquad y' = ((b-t)/H).y + t$

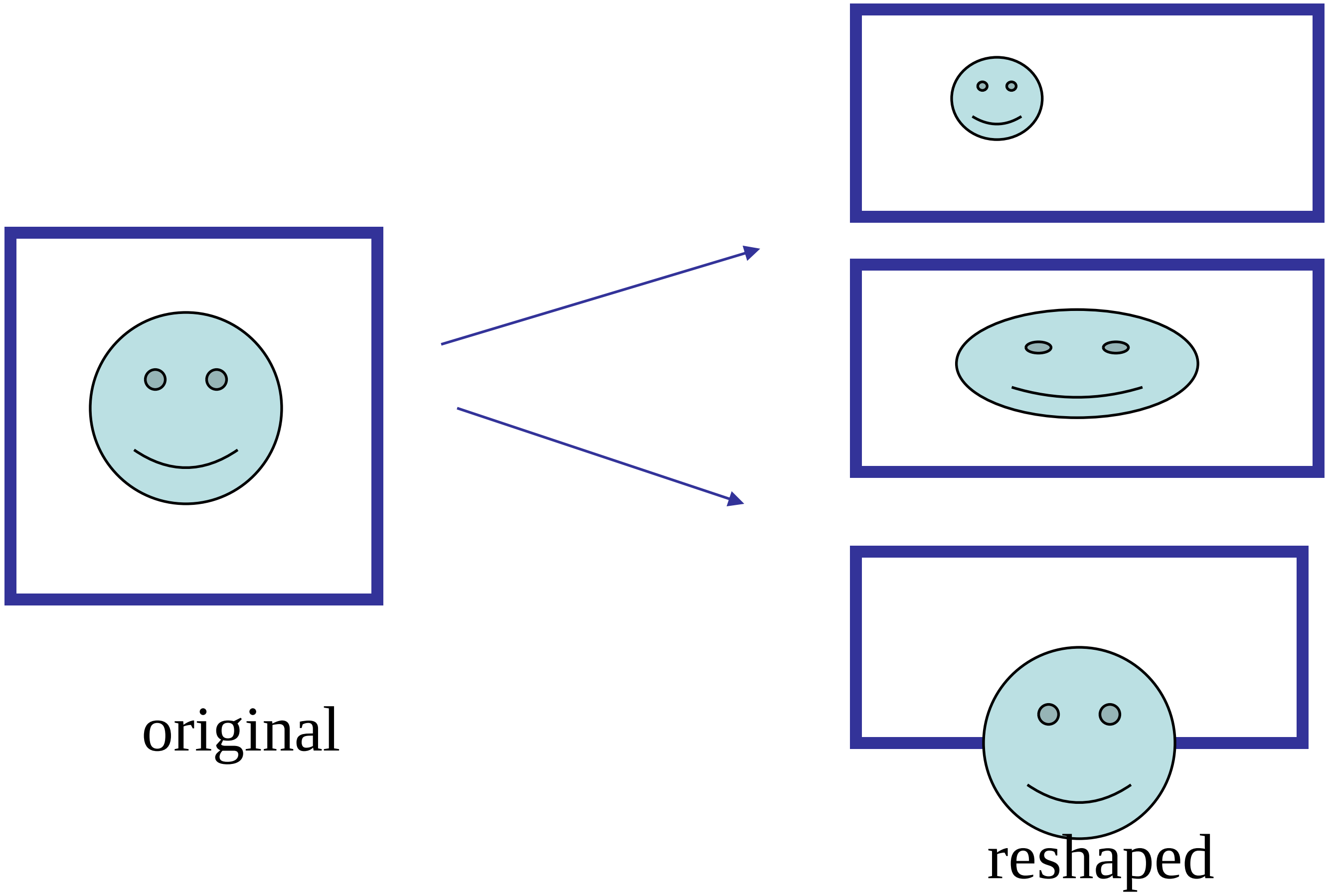
$l = A.0 + B$

Mouse Event

- Mouse Move
  - glutMotionFunc: mouse buttons are pressed
  - glutPassiveMotionFunc: no mouse buttons are pressed
  - Callback Function Prototype

```
void myMoveMouse(int x, int y)
```

Reshape Event

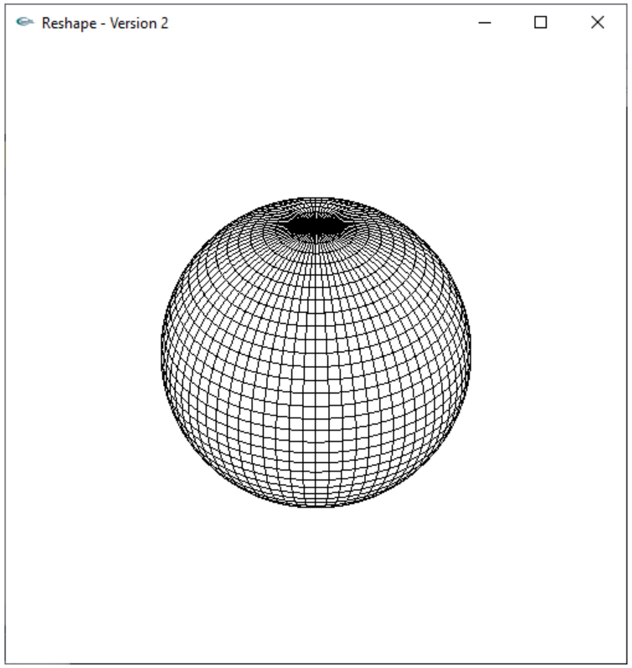


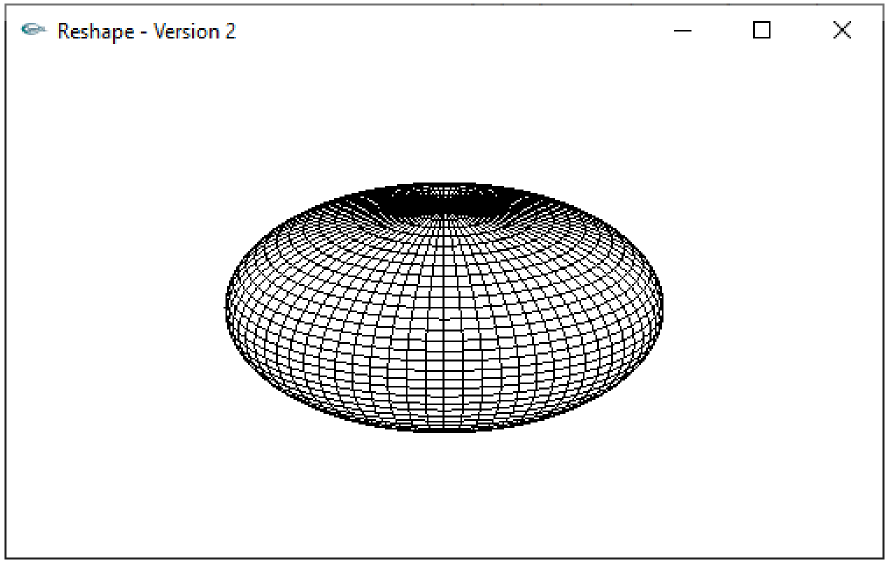
Reshape Event

- The reshape callback
  - **glutReshapeFunc(myreshape);**
  - **void myReshape(int w, int h)**
    - Returns width and height of new window (in pixels)
  - A redisplay is posted automatically at end of execution of the callback
  - GLUT has a default reshape callback but you probably want to define your own
  - The reshape callback is good place to put viewing functions because it is invoked when the window is first opened

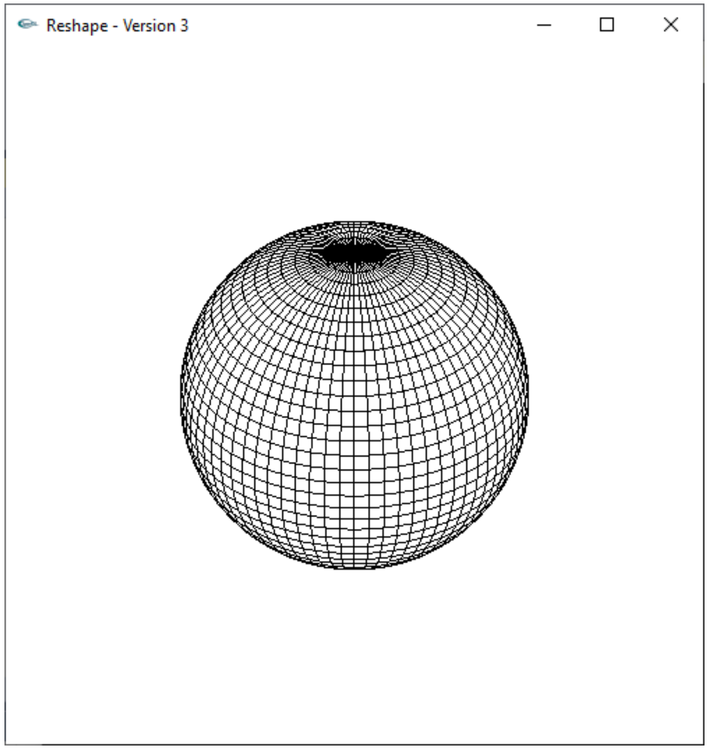
Reshape Event

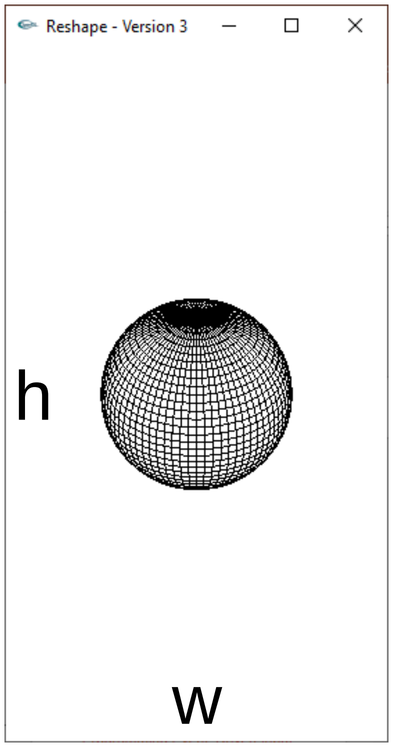
```
void myReshape(int w, int h){
        glViewport(0, 0, w, h);
}
```

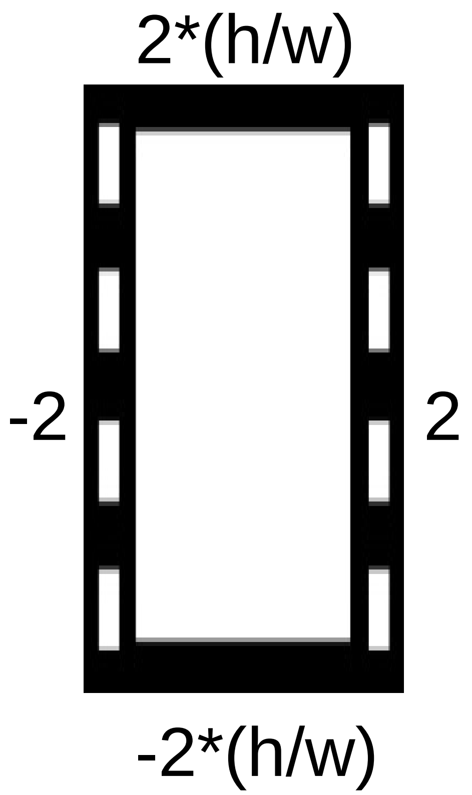




Reshape Event



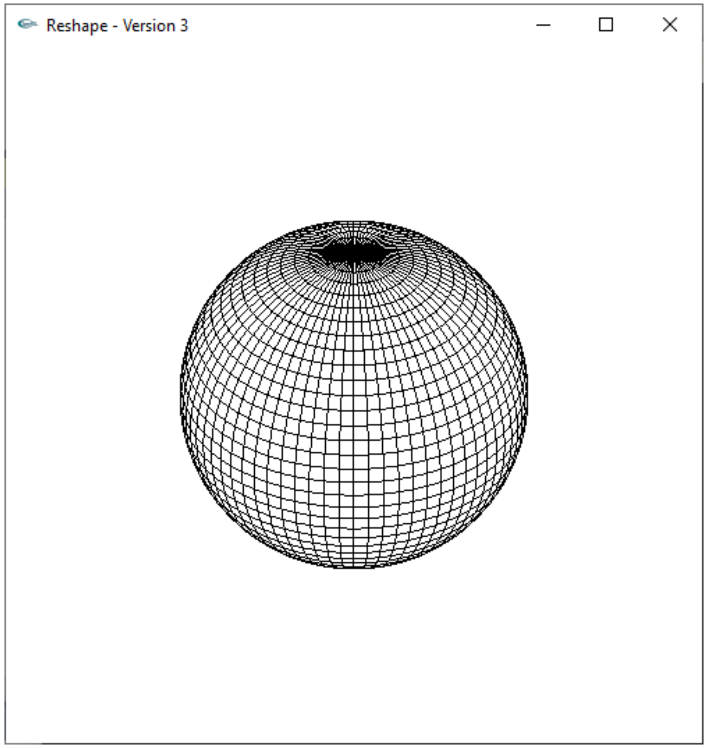


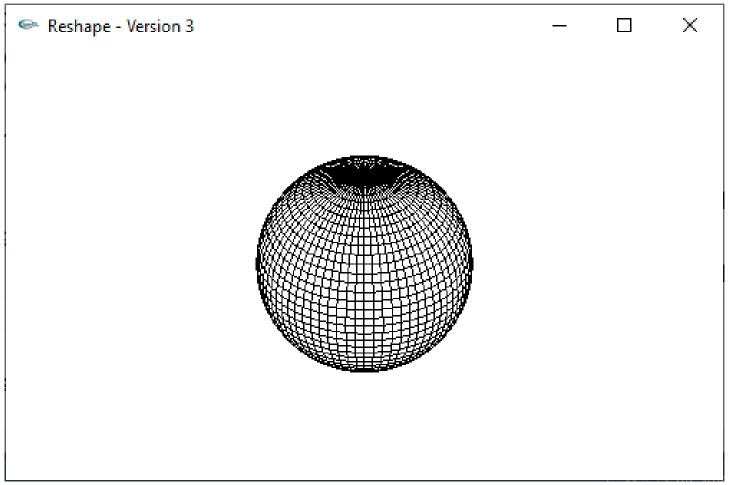


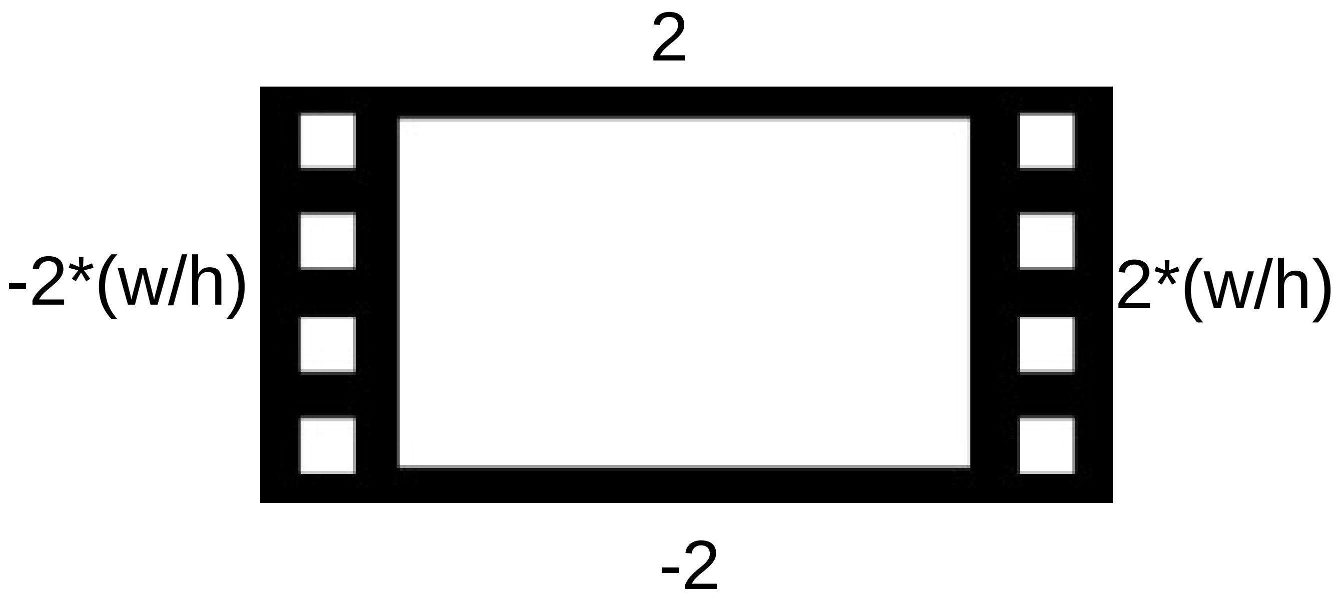
Reshape Event

```
void myReshape(int w, int h){
        float factor = 2;
        glViewport(0, 0, w, h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        if (w <= h)
                glOrtho(-factor, factor, -factor * h / w,
                        factor * h / w, -10.0, 10.0);
        else
                glOrtho(-factor * w / h, factor * w / h,
                        -factor, factor, -10.0, 10.0);
}
```

Reshape Event







Reshape Event

```
void myReshape(int w, int h){
        float factor = 2;
        glViewport(0, 0, w, h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        if (w <= h)
                glOrtho(-factor, factor, -factor * h / w,
                        factor * h / w, -10.0, 10.0);
        else
                glOrtho(-factor * w / h, factor * w / h,
                        -factor, factor, -10.0, 10.0);
}
```

Idle Event

- The idle callback is executed whenever there are no events in the event queue
  - glutIdleFunc(myidle)
  - **Useful for animations**

```
void myidle() {
  /* change something */
  t += dt
  glutPostRedisplay();
}
void mydisplay() {
  glClear();
  /* draw something that depends on t */
  glutSwapBuffers();
}
```

Double Buffer

- Double Buffering
  - Instead of one color buffer, we use two
    - Front Buffer**: one that is displayed but not written to**
    - Back Buffer**: one that is written to but not displayed**
  - Program then requests a double buffer in main.c
    - glutInitDisplayMode(GL_RGB | GL_DOUBLE)
    - At the end of the display callback buffers are swapped
    - void mydisplay() {

```
      glClear(GL_COLOR_BUFFER_BIT|….)
      /* draw graphics here */
      glutSwapBuffers()
    }
```

Toolkits and Widgets

- Most window systems provide a toolkit or library of functions for building user interfaces that use special types of windows called *widgets*
- Widget sets include tools such as
  - **Menus**
  - **Slidebars**
  - **Dials**
  - **Input boxes**
- But toolkits tend to be platform dependent
- GLUT provides a few widgets including menus

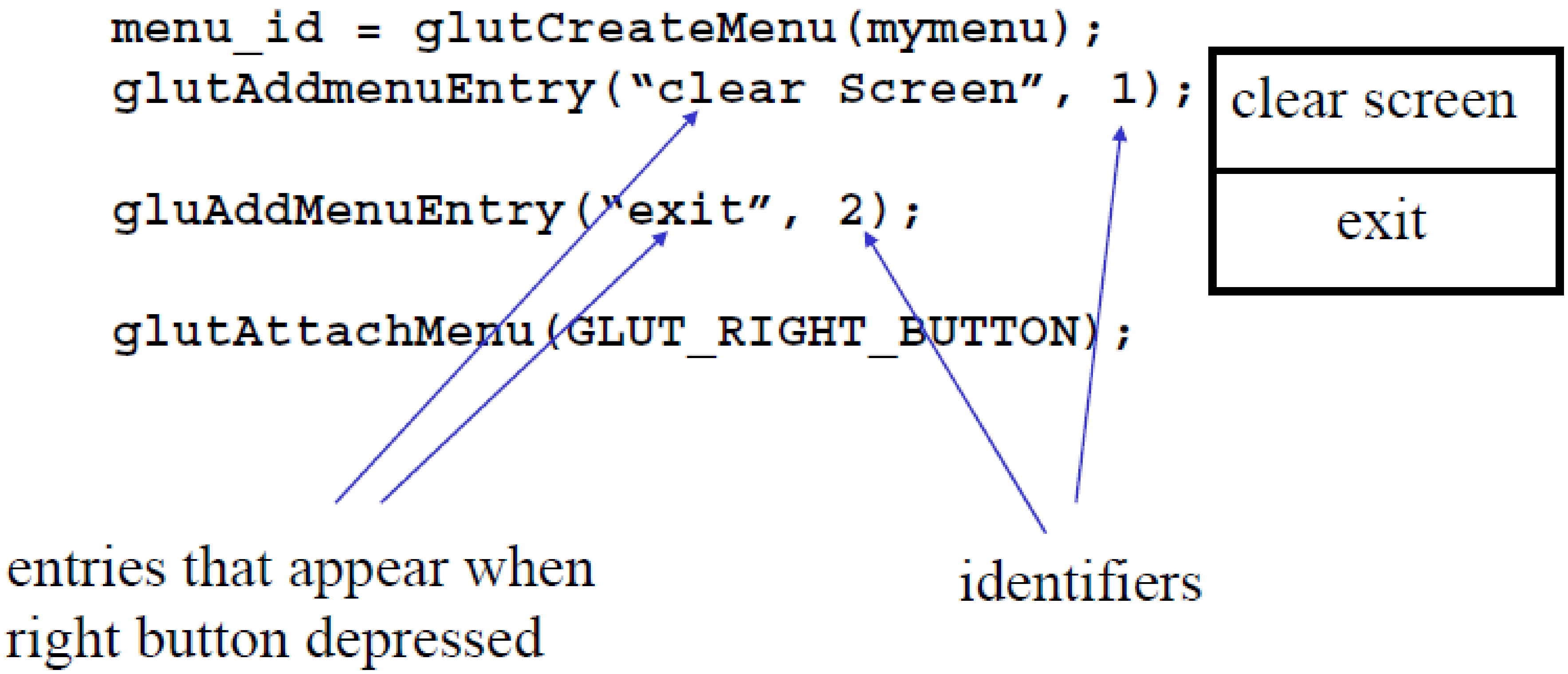
Toolkits and Widgets

- Menus
  - GLUT supports pop-up menus
    - **A menu can have submenus**
  - Three steps
    - **Define entries for the menu**
    - **Define action for each menu item**

      →Action carried out if entry selected
    - **Attach menu to a mouse button**

Toolkits and Widgets

- Menus



```
menu_id = glutCreateMenu(mymenu);
glutAddmenuEntry("clear Screen", 1);

gluAddMenuEntry("exit", 2);

glutAttachMenu(GLUT_RIGHT_BUTTON);
```

Toolkits and Widgets

- Menu
  - **Menu callback**

```
void mymenu(int id) {
        if(id == 1) glClear();
        if(id == 2) exit(0);
}
```

  - **Note each menu has an id that is returned when it is created**
  - **Add submenus by**
    - glutAddSubMenu(char *submenu_name,
                submenu id)