

Input and Interaction

514780
2018년 가을학기
9/27/2018
단국대학교 박경신

Overview

- 입력장치 (Input device)
 - 물리적 입력 장치(Physical input devices)
 - Mouse, Keyboard, Trackball
 - 논리적 장치
 - String, Locator, Pick, Choice, Valuators, Stroke device
- 입력 모드
 - Request mode, Sample mode, Event mode
- GLUT Devices & Event-driven programming
 - 마우스, 키보드, 메뉴, 조이스틱, 테블릿, ..

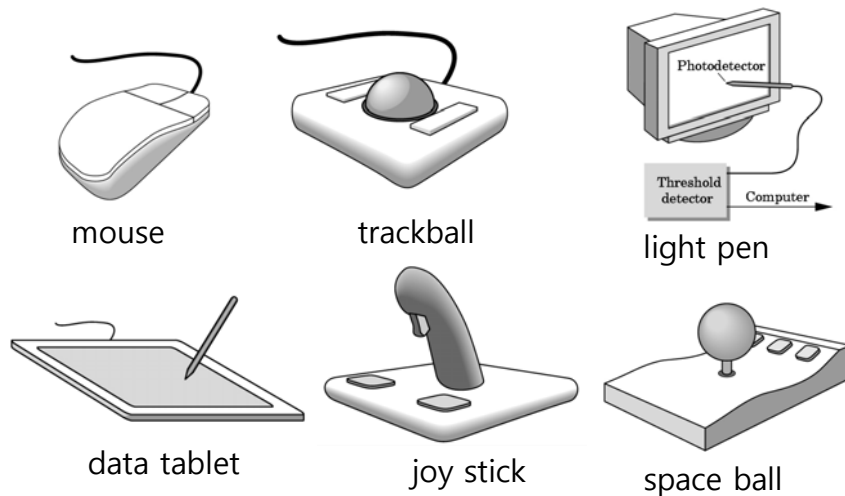
Interaction

- 컴퓨터 기술의 중요한 발전 가운데 하나는 사용자가 컴퓨터 화면을 이용하여 상호작용을 할 수 있다는 것이다.
- 상호작용 (Interaction)
 - 사용자는 마우스와 같은 대화식 장치를 통하여 행동을 취한다.
 - 컴퓨터가 사용자의 입력을 감지한다.
 - 프로그램은 이 입력에 반응하여 상태를 바꾼다.
 - 프로그램은 이 새로운 상태를 디스플레이한다.
 - 사용자는 바뀐 디스플레이를 본다.
 - 사용자가 이 변화에 다시 반응하는 과정들이 반복된다.

Graphical Input

- 입력장치를 두 가지 다른 방법으로 생각할 수 있다
 - 물리적 장치
 - Mouse, Keyboard, Trackball
 - 논리적 장치
 - 물리적 특성이 아닌 사용자 프로그램과의 상위 인터페이스에 의하여 특정지어짐
- 입력 모드
 - 입력장치가 응용프로그램에게 입력을 제공하는 방식은 측정과정(Measure)과 장치 트리거(Trigger)로 설명될 수 있다
 - 요구 모드 (Request mode)
 - 샘플 모드 (Sample mode)
 - 이벤트 모드 (Event mode)

Physical Input Devices



Physical Input Devices

- 물리적 입력장치 (Physical input devices)
 - 지시 장치 (Pointing devices)
 - 사용자가 화면상의 한 위치를 지시할 수 있게 함
 - 대부분의 경우 사용자가 컴퓨터에 신호 즉, 인터럽트를 보내기 위해 1 이상의 버튼을 가지고 있다.
 - Mouse, trackball, tablet, lightpen, joystick, spaceball
 - 키보드 장치 (Keyboard devices)
 - 프로그램에 문자 코드를 반환하는 장치
 - Keyboard

Relative Positioning Device

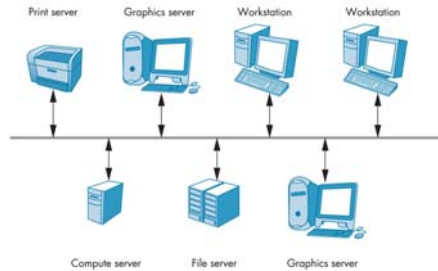
- 데이터 테블렛 (data tablet) 같은 장치는 절대위치를 제공한다.
- 마우스, 트랙볼, 조이스틱 같은 장치는 변화하는 속도를 적분하여 위치정보가 계산된다.
 - Rotation of cylinders in mouse
 - Roll of trackball
 - 정확한 절대위치를 구하기 어려움
 - 장치를 가변-감도 입력으로 사용할 수 있음

Logical Input Devices

- 문자열 장치 (String device) - keyboard
 - 사용자 프로그램에 ASCII문자열(strings of characters)을 제공
- 위치 장치 (Locator device) – mouse, trackball
 - 사용자 프로그램에 실세계 좌표의 위치(position)를 제공
- 지적 장치 (Pick device) – mouse button, gun
 - 사용자 프로그램에 객체의 식별자(ID)를 반환
- 선택 장치 (Choice device) – widgets, function keys, mouse button
 - 사용자가 선택사항들 (menu)중 하나를 선택하도록 함
- 벨류에이터 장치 (Valuators) – slide bars, joystick, dial
 - 사용자 프로그램에 아날로그 입력(range of value)을 제공
- 획 장치 (Stroke) – mouse drag
 - 위치들의 배열(array of positions)을 반환

X Window Input

- X Window System 모델은 클라이언트와 서버 모델 (client-server model) 네트워크로 구성.
 - 그래픽스 서버(Graphics Server)는 래스터 디스플레이, 키보드, 마우스를 가진 워크스테이션이다. 이 서버는 디스플레이에 출력 서비스를 키보드와 마우스를 통한 입력 서비스를 제공한다.
 - 클라이언트 (Client) - OpenGL 프로그램은 그래픽스 서버를 사용하는 클라이언트이다.



Input Modes

- 입력장치는 OS에 신호를 보내는 트리거(trigger)를 가지고 있다.
 - 마우스 "버튼"
 - 키보드에서 "리턴/엔터"키
- 입력장치는 트리거 시 컴퓨터에게 측정치(measure)를 넘겨준다. 신호를 주는데 사용할 수 있는 장치의 물리적 입력이다.
 - 마우스는 "위치정보"를 준다.
 - 키보드는 "ASCII code"를 준다.

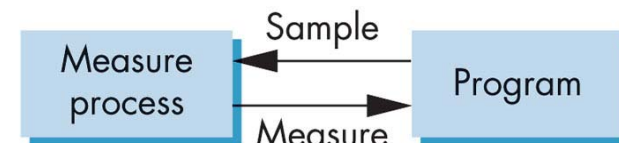
Request Mode

- 요구모드 (Request mode)에서는 장치가 트리거 될 때까지 측정치가 프로그램으로 반환되지 않는다.
- 문자입력을 요구하는 전형적인 비그래픽스적인 응용 프로그램의 표준
 - 예를 들어 C 프로그램의 scanf 함수가 사용될 때 프로그램이 단말기에서 문자를 칠 때까지 기다리면서 정지한다. 그리고 엔터키(trigger)를 누르기까지 타이핑과 교정이 가능하다.



Sample Mode

- 샘플모드 (Sample mode)에서는 즉각적인 입력을 제공한다. 사용자 프로그램에서 함수의 호출을 만나면 즉시 측정치가 반환된다. 따라서 트리거가 필요하지 않다.
- 예: C 프로그램의 getch



Event Mode

- 대부분의 시스템은 여러 개의 입력 장치가 있으며 각 장치마다 자신의 트리거로 측정과정을 실행하는 환경으로 구성된다
- 입력 장치가 트리거(trigger)될 때마다 이벤트(event)가 생성되고, 장치의 측정치가 장치 식별자(ID)와 더불어 이벤트 큐(event queue)에 들어간다.
- 답신(callback) 함수를 특정 이벤트와 연결하여 사용한다.



Event Types

- Window – 윈도우 resize, expose, iconify
- Keyboard – 키를 누름(press)와 키에서 땀(release)
- Mouse – 마우스 버튼을 누름
- Motion – 마우스 움직임
- Idle – 이벤트가 없음

Programming Event-Driven Input

- 이벤트-구동(Event-drive) 입력 프로그래밍을 위해 이벤트에 대응하는 답신 함수(callback function)를 정의한다.
- 이벤트가 발생시 해당 답신 함수(callback function)이 동작한다.
- 예를 들어, GLUT에서 마우스 이벤트의 답신 함수는 main 함수 내에서 glutMouseFunc(mouse)를 통해서 지정한다. 그리고 mouse 답신 함수의 형태는 다음과 같다.

```
void mouse(int button, int state, int x, int y)
```

GLUT Devices

- Keyboard
 - "normal" keys
 - "special" keys
- Mouse
 - Position
 - buttons
- Joystick
- Tablet
- Dial/button box
- Spaceball

GLUT Keyboard Functions

- glutKeyboardFunc(func)
 - ASCII 'character' 키가 눌렸을 때 불러짐
- glutSpecialFunc(func)
 - 특수 'special' 키가 눌렸을 때 불러짐
- glutKeyboardUpFunc(func)
 - ASCII 'character' 키가 떼어졌을 때 불러짐
- glutSpecialUpFunc(func)
 - 특수 'special' 키가 떼어졌을 때 불러짐
- glutGetModifiers()
 - 이벤트 발생시 Shift, Control, Alt keys의 상태를 줌
- glutIgnoreKeyRepeat(val)
 - GLUT에게 자동 키보드 반복을 무시하라고 알려줌

GLUT Keyboard Event Callback

- void keyboard(unsigned char key, int x, int y)
 - 키보드 인터랙션에 대한 처리를 지정.
 - key 인자는 ASCII character code로 지정
 - x, y 인자는 키를 눌렀을 때의 마우스의 위치

```
void keyboard(unsigned char key, int x, int y)
{
    switch (key): /* q-key exits the program */
    {
        case 'q':
            exit(0);
    }
}
```

GLUT Special Key

- GLUT 특수키(special key) 지정
 - GLUT_KEY_{F1,F2,..,F12}
 - GLUT_KEY_{UP,DOWN,LEFT,RIGHT} – 방향키
 - GLUT_KEY_{PAGE_UP,PAGE_DOWN,HOME,END,INSERT}

```
void specialkey(int key, int x, int y)
{
    switch(key) {
        case GLUT_KEY_F1:
            red = 1.0; green = 0.0; blue = 0.0; break;
        case GLUT_KEY_F2:
            ...
    }
}
```

GLUT Modifier Key

- CTRL, ALT, SHIFT modifier key가 눌렸는지 확인하려면 int glutGetModifiers(void) 사용.
 - GLUT_ACTIVE_SHIFT – SHIFT 키 (혹은 Caps Lock인 상태)
 - GLUT_ACTIVE_CTRL
 - GLUT_ACTIVE_ALT

```
void keyboard(unsigned char key, int x, int y)
{
    if (key == 27) /* ESC-key exits the program */
        exit(0);
    else if (key == 'r') {
        int mod = glutGetModifier();
        if (mod == GLUT_ACTIVE_CTRL)
            red = 0.0;
        else
            red = 1.0;
    }
}
```

GLUT *Mouse* Functions

- `glutMouseFunc(void(*func)(int button, int state, int x, int y))`
 - 마우스 버튼이 눌러졌을 때 불러짐
- `glutMotionFunc(void(*func)(int x, int y))`
 - 마우스가 버튼이 눌러진 상태에서 움직일 때 불러짐
- `glutPassiveMotionFunc(void (*func)(int x, int y))`
 - 마우스 버튼이 눌러지지 않은 상태에서 움직일 때 불러짐

GLUT *Mouse* Event Callback

- `void mouse(int button, int state, int x, int y)`
 - *button* 인자에는 GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON
 - *state* 인자에는 GLUT_DOWN (마우스 버튼이 눌러졌을 때) GLUT_UP (마우스 버튼이 떼어졌을 때)
 - *x, y* 인자는 마우스 버튼이 눌렀거나 떼어졌을 때 마우스의 위치 (in GLUT window coordinates)

```
void mouse(int button, int state, int x, int y)
{
  ...
}
```

GLUT *Motion* Event Callback

- `void motion(int x, int y)`
 - *x, y* 인자는 마우스의 새로운 위치 (in GLUT window coordinates)

```
void motion(int x, int y)
{
  ...
}
```

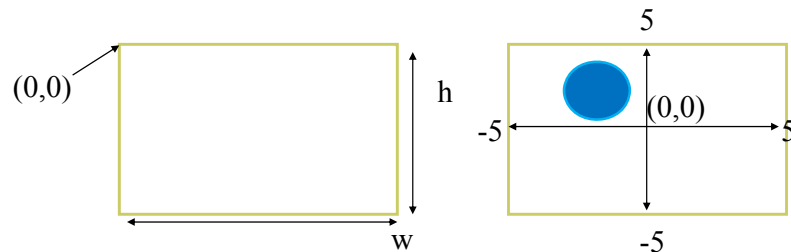
GLUT *Mouse* Enter/Leave the Window

- `void glutEntryFunc(void(*func)(int state));`
- `void entry(int state)`
 - *state* 인자는 GLUT_LEFT, GLUT_ENTERED
- X-window에서는 정확히 윈도우에 마우스가 들어오고 나갈 때의 이벤트를 발생시킴. Windows OS에서는 마우스를 현재 윈도우에 클릭해야 window focus가 바뀌게 되므로 이 함수가 정확히 작동되지 않음.

```
void entry(int state)
{
  ...
}
```

Mouse Positioning

- GLUT screen coordinate은 원점을 왼쪽 위쪽(top-left corner)으로 하고 x+는 오른쪽, y+는 아래쪽으로 1 pixel 단위로 증가한다.
- OpenGL은 2D drawing coordinate은 원점이 왼쪽 아래쪽 (bottom-left corner)으로 하고, x+는 오른쪽, y+는 위쪽으로 증가한다.



Drawing geo at cursor location

```
void mouse(int button, int state, int x, int y) {
    if(button==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
        exit(0);
    if(button==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
        g_mousemove = true;
    else if(button==GLUT_LEFT_BUTTON && state==GLUT_UP)
        g_mousemove = false;
}

void motion(int mx, int my) {
    int w = glGet(GLUT_WINDOW_WIDTH);
    int h = glGet(GLUT_WINDOW_HEIGHT);
    float x = (float) 10 * (mx - w*0.5) / w; // 0~600(x+right) => -5~5(x+right)
    float y = (float) 10 * (h*0.5 - my) / h; // 0~600(y+down) => x -5~5(y+ up)
    if (g_mousemove) {
        geo->setPosition(glm::vec3(x, y, 0));
    }
    glutPostRedisplay();
}
```

If both a mouse button and ALT key are pressed

```
void mouse(int button, int state, int x, int y)
{
    specialKey = glutGetModifiers();
    if((state==GLUT_DOWN)&&(specialKey == GLUT_ACTIVE_ALT))
    {
        if (button == GLUT_LEFT_BUTTON) {
            red = 1.0; green = 0.0, blue = 0.0;
        }
        else if (button = GLUT_MIDDLE_BUTTON){
            red = 0.0; green = 1.0, blue = 0.0;
        }
        ...
    }
}
```

GLUT Device Functions

- glutJoystickFunc(func)
 - 조이스틱의 상태를 매 t milliseconds 단위로 호출
- glutTabletMotionFunc(func)
 - 테블릿 puck이 움직였을 때 호출
- glutTabletButtonFunc(func)
 - 테블릿 버튼이 눌렀거나 떴을 때 호출
- glutDialsFunc(func)
 - 다이얼이 돌려졌을 때 호출
- glutButtonBoxFunc(func)
 - 다이얼/버튼박스의 버튼이 눌렀거나 떴을 때 호출

Widgets

- 대부분의 윈도우 시스템은 그래픽 사용자 인터페이스 (Graphical User Interface) 위젯 (widgets)을 제공
- 그래픽 디스플레이에 입력기능을 제공하는 고수준 인터페이스 (Higher-level interface)
- 물리적 입력장치가 제공하는 대부분의 기능을 제공한다.
- 위젯의 예
 - Menu
 - Push Buttons
 - Radio buttons
 - Sliders
 - Scroll bars
 - Dials

GLUT Menu Functions

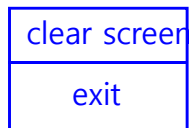
- GLUT는 간단한 팝업메뉴 (pop-up menu) 기능을 제공한다.
- 먼저 메뉴 안에 항목들을 정의한 후 이 팝업메뉴를 특정 마우스 버튼과 연결해야 한다 (이 마우스 버튼을 눌렀을 때 나타날 수 있도록).
- 각 메뉴 항목에 해당하는 답신 함수를 정의해야 함.
- 주 menu 아래 sub-menus를 만들 수 있다.

```
id = glutCreateMenu(func); /* starts defining a new menu */
glutAddMenuEntry(label, value); /* adds an entry to the menu */
glutAddSubMenu(label, id); /* adds a submenu to the menu */
glutAttachMenu(button); /* defines mouse buttons to pop-up the menu */
```
- Menu는 실행 중에 아래와 같은 함수를 써서 바꿀 수도 있다.
 - glutSetMenu, glutRemoveMenuItem, glutChangeToMenuEntry, glutChangeToSubMenu
- Menu는 실행 중에 아래와 같은 함수를 써서 버튼에 attach 혹은 detach가 가능하다.
 - glutDetachMenu

Defining a simple menu

- `int glutCreateMenu(void (*func)(int value));`
- `void glutAddMenuEntry(char *name, int value);`
 - name인자는 메뉴에 나타나는 이름
 - value인자는 메뉴 엔트리가 선택됐을 때 답신함수에 보내지는 값
- `void glutAttachMenu(int button);`
 - button인자에는 GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON

```
menu_id = glutCreateMenu(menu);
glutAddMenuEntry("clear screen", 1);
glutAddMenuEntry("exit", 2);
glutAttachMenu(GLUT_RIGHT_BUTTON);
```



Menu actions

- Menu callback

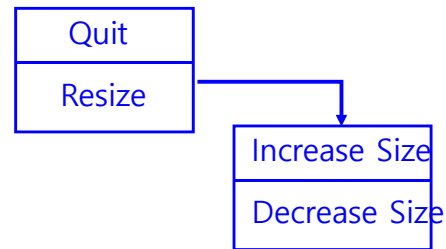
```
void menu(int id)
{
    if(id == 1) glClear();
    if(id == 2) exit(0);
}
```

- 각각의 메뉴는 생성시 ID를 갖는다.
- `glutAddSubMenu(char *submenu_name, submenu id)`를 사용하여 sub-menus 추가

Hierarchical Menus

```
sub_menu = glutCreateMenu(size_menu);
glutAddMenuEntry("Increase Size", 2);
glutAddMenuEntry("Decrease Size", 3);
```

```
glutCreateMenu(top_menu);
glutAddMenuEntry("Quit", 1);
glutAddSubMenu("Resize", sub_menu);
glutAttachMenu(GLUT_RIGHT_BUTTON);
```



Idle Callback

- glutIdleFunc(void (*func)(void))의 답신함수는 다른 이벤트가 없을 때 실행됨.
- Idle은 애니메이션 움직임에 사용함, e.g. rotating square

```
void idle() {
    /* change something */
    t += dt
    glutPostRedisplay();
}
```
- void display() {
glClear();
/* draw something that depends on t */
glutSwapBuffers();
}
- Idle의 default 답신함수(callback function)는 NULL.

The display callback

- 디스플레이 답신함수는 GLUT에서 윈도우가 새로 그리기(window refresh)를 요할 때 불려진다.
 - 윈도우가 처음으로 열렸을 때
 - 윈도우가 재구성이 필요할 때
 - 윈도우가 expose됐을 때
 - 사용자 프로그램에서 디스플레이가 바뀌길 원할 때
- glutDisplayFunc(display)는 모든 GLUT 프로그램에서 반드시 불리는 display callback이다.

glutPostRedisplay

- void glutPostRedisplay() 함수는 윈도우가 새로 그려져야 할 필요가 있는 경우를 표시하는 일을 한다.
- 일반적으로 display callback 함수를 직접 호출하는 것 보다, 이 함수를 사용하면 디스플레이가 다시 그리기 지시 플래그를 GLUT 메인루프 내에 설치해서, 과다하거나 불필요한 화면 그리기를 방지한다.
- glutPostRedisplay를 사용하면 이 프로그램이 이벤트 루프를 처리 할 때마다 디스플레이가 한 번 실행되도록 한다.

Animating a Display

- Display callback 함수 내에서 디스플레이를 새로 그릴 때 `glClear()`를 사용하여 먼저 윈도우를 clear한다.
- 그러나, 프레임 버퍼에서의 그림 정보는 그림 내용의 디스플레이와 분리되어 있다.
- 따라서 반쪽만 그리게 되는 디스플레이를 보게 됨.
- `single_double_buffer.cpp` 참고 – rotating cube 예제

Double Buffering

- 2개의 color buffer를 사용
 - Front Buffer – 화면에 출력
 - Back Buffer – 그래픽스 렌더링으로 래스터 이미지 생성
- Double Buffer
 - 더블 버퍼를 초기화
 - `glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB)`
 - display callback에서 버퍼를 지움
 - `glClear(GL_COLOR_BUFFER_BIT | ...)`
 - display callback에서 마지막으로 스왑 버퍼를 부름
 - `glutSwapBuffers()`

The *Reshape* callback

- `glutReshapeFunc(reshape)`는 재구성 이벤트 답신함수를 호출한다.
- `void reshape(int w, int h)`
 - 윈도우의 넓이 (width)와 높이(height)를 반환한다.
 - 이 답신함수는 자동적으로 `redisplay`를 부른다.
- Reshape callback 함수는 윈도우가 처음 열렸을 때 호출되므로 관측함수를 넣기에 좋은 곳이다.

Example Reshape

```
void reshape(int w, int h)
{
    g_aspectRatio = (float) (w/h);
    g_Projection = glm::perspective(g_fovy, g_aspect, g_near, g_far);

    glViewport(0, 0, w, h);
    glutPostRedisplay();
}
```