FreeGLUT 콜백함수 활용하기

김준호

Abstract

FreeGLUT에서 윈도우 이벤트를 처리하기 위해 콜백함수(callback function)를 등록하고 활용하는 방법을 학습한다.

1 그래픽 유저 인터페이스 환경에서의 이벤트

최근 운영체제는 프로그램을 윈도우 형태로 구동하고 사용자가 마우스나 키보드를 통해 윈도우를 조작할 수 있는 그래픽 유저 인터페이스(Graphical User Interface, 이하 GUI) 환경을 제공한다. 대개 GUI 환경에서 동작하는 윈도우를 프로그래밍하는 방법은 운영체제가 제공하는 다양한 이벤트(event)를 적절히 처리하는 방식으로 코딩된다.

2 FREEGLUT에서의 이벤트 처리

FreeGLUT는 운영체제와 무관하게 OpenGL 기반 윈도우 프로그래밍이 가능하도록 설계한 라이브러리이기 때문에, 대부분의 GUI 기반 운영체제에서 발생하는 이벤트를 다음과 같이 일반화하였다.

- 화면 그리기 이벤트
- 화면 크기변경 이벤트
- 마우스 클릭 이벤트
- 마우스 움직임 이벤트
- 마우스 휠 이벤트
- 키보드 입력 이벤트
- 백그라운드 처리 이벤트

FreeGLUT로 프로그래밍된 윈도우는 glutMainLoop() 함수에서 무한루프를 돌며 운영체제가 현재 윈도우에 전달하는 이벤트를 듣게 된다. FreeGLUT는 윈도우가 전달하는 각종 이벤트에 반응할 수 있도록 설계되어 있지만, 기본적인 FreeGLUT 프로그램은 각 이벤트에 대해 아무런 반응을 보이지 않는다.

만일 프로그래머가 특정 이벤트에 대해 반응할 수 있는 FreeGLUT 프로그램을 작성하려면 크게 다음의 두 단계를 거쳐야 한다.

1) 해당 이벤트를 처리하도록 콜백함수를 등록
2) 그 콜백함수에서 해당 이벤트가 적절히 처리되도록 함수 선언 및 함수 구현

예를 들어 FreeGLUT에서 '화면 그리기 이벤트'를 처리하려면 다음과 같이 하면 된다.

```
// 2-1) 화면 그리기 이벤트를 처리할 콜백함수 mydisplay 함수 선언
void mydisplay();

int main(int argc, char* argv[])
{
  // ...
  // 1) 화면 그리기 이벤트에 대한 콜백함수 mydisplay 등록 (반드시 glutMainLoop 함수가 불리기 전에 등록!)
  glutDisplayFunc(mydisplay);
  // ...
  glutMainLoop();

  return  0;
}

// 2-2) 화면 그리기 이벤트를 처리할 콜백함수 mydisplay 함수 구현
void mydisplay()
{

}
```

국민대학교 소프트웨어학부, junho@kookmin.ac.kr

다음은 FreeGLUT에서 다룰 수 있는 대표적인 이벤트에 대해 콜백함수를 등록하는 함수들이다. FreeGLUT에서는 콜백함수의 등록이 함수포인터(function pointer)를 넘기는 방식으로 동작하기 때문에, 다음 리스트에서 ... 부분은 콜백함수의 이름(즉 함수포인터)로 구성된다.

- 화면 그리기 콜백함수 등록: glutDisplayFunc(...)
- 화면 크기변경 콜백함수 등록: glutReshapeFunc(...)
- 마우스 클릭 콜백함수 등록: glutMouseFunc(...)
- 마우스 움직임 콜백함수 등록: glutMotionFunc(...)
- 마우스 휠 콜백함수 등록: glutMouseWheelFunc(...)
- 키보드 입력 콜백함수 등록: glutKeyboardFunc(...)
- 백그라운드 처리 콜백함수 등록: glutIdleFunc(...) 멍하니 있을 때

콜백함수를 이용하는 자세한 방법은 GLUT API 레퍼런스를 참고하도록 한다. 여기서는 주요 이벤트를 어떻게 콜백함수로 다루는지 개괄적으로 살펴보기로 한다.

2.1 화면 그리기 이벤트 처리

화면 그리기 이벤트를 처리할 콜백함수는 다음과 같이 정의되는 glutDisplayFunc함수를 이용하여 등록하면 된다.

```
void glutDisplayFunc(void (*func)(void));
```

여기서 눈여겨 봐야할 점은 glutDisplayFunc함수의 입력이 void (*func)(void)로 정의된 함수포인터라는 점이다. 즉, glutDisplayFunc에 의해 콜백함수로 등록될 수 있는 함수는 반드시 입력이 void이고 출력이 void여야 한다. 따라서, 다음과 같은 mydisplay 함수는 glutDisplayFunc에 의해 콜백함수로 등록될 수 있다.

```
void mydisplay();                  // 콜백함수 선언

  // main 함수 안...
  glutDisplayFunc(mydisplay);    // 콜백함수 등록
  // ...

void mydisplay()                   // 콜백함수 구현
{
  // ...
}
```

2.2 화면 크기변경 이벤트 처리

화면 크기변경 이벤트를 처리할 콜백함수는 다음과 같이 정의되는 glutReshapeFunc함수를 이용하여 등록하면 된다.

```
void glutReshapeFunc(void (*func)(int width, int height));
```

여기서 눈여겨 봐야할 점은 glutReshapeFunc함수의 입력이 void (*func)(int width, int height)로 정의된 함수포인터라는 점이다. 즉, glutReshapeFunc에 의해 콜백함수로 등록될 수 있는 함수는 반드시 입력으로 두개의 int형을 받고 출력은 void여야 한다. 따라서, 다음과 같은 myreshape 함수는 glutReshapeFunc에 의해 콜백함수로 등록될 수 있다.

```
float g_aspect = 1.0f;     // aspect ratio: width/height

void myreshape(int width, int height);  // 콜백함수 선언

  // main 함수 안...
  glutReshapeFunc(myreshape);           // 콜백함수 등록
  // ...

void myreshape(int width, int height)    // 콜백함수 구현
{
  glViewport(0, 0, width, height);             // 바뀐 창크기에 맞게 viewport 설정
  g_aspect = (float) width / (float) height;  // 바뀐 창크기에 맞게 aspect ratio 업데이트

  // ...

  glutPostRedisplay();
}
```

일반적으로 화면 크기가 바뀌면 OpenGL 프로그램에서는 다음의 일을 수행해야 한다.

- 뷰포트(viewport) 재설정: 보통 창크기에 꽉채워 OpenGL 렌더링 결과물을 디스플레이하는 것이 일반적이기 때문에 뷰포트의 좌측하단은 $(0, 0)$으로 우측상단은 $(width, height)$로 설정할 수 있도록 glViewport함수를 호출한다.
- 종횡비(aspect ratio) 재설정: 창 크기가 바뀌게 되면 $\frac{width}{height}$에 해당하는 종횡비를 재설정 하는 것이 좋다. 종횡비는 추후 카메라에 대한 프로젝션 행렬을 설정할 때 중요하게 활용된다.
- 화면 다시 그리기 요청: 화면 크기변경 이벤트를 처리하는 콜백함수에서는 마지막 순간 glutIdleFunc함수를 부르는 것이 좋다. 그 이유는 화면 크기가 바뀌었기 때문에 화면 내용을 OpenGL로 다시 그려야만 제대로된 화면이 구성되는 경우가 많기 때문이다. 이때 glutDisplayFunc에 등록된 화면 그리기 콜백함수(예, mydisplay)를 직접 호출하지 말고 반드시 glutIdleFunc함수를 호출하도록 하자. glutIdleFunc함수는 운영체제로 하여금 화면 그리기 이벤트 발생시키게 유도하는 방식으로 GUI환경에서 동작하는 윈도우 프로그래밍에서 정석에 해당하는 화면 갱신 방법이다.

2.3 마우스 클릭 이벤트 처리

마우스 버튼 클릭 이벤트를 처리할 콜백함수는 다음과 같이 정의되는 glutMouseFunc함수를 이용하여 등록하면 된다.

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

glutMouseFunc함수의 콜백함수로 등록된 함수는 매개변수를 통해 다음 정보를 받아온다.

- button: GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON 중 하나의 값을 가지며 각각 마우스의 왼쪽, 가운데, 오른쪽 버튼에 해당하는 이벤트가 발생했음을 의미한다.
- state: GLUT_UP 혹은 GLUT_DOWN 중 하나의 값을 가지며, button이 어떤 상태에 놓여 있을 때 이벤트가 발생했는지 알려준다.
- x, y: 마우스 이벤트가 발생했을 당시 마우스의 위치를 의미한다.

2.4 마우스 움직임 이벤트 처리

마우스 움직임 이벤트를 처리할 콜백함수는 다음과 같이 정의되는 glutMouseFunc함수 또는 glutPassiveMotionFunc함수를 이용하여 등록하면 된다. 두 콜백함수 등록의 차이점은 다음과 같다.

- glutMouseFunc에 등록된 콜백함수: 마우스 버튼이 클릭된 채 마우스가 움직이는 경우 호출
- glutPassiveMotionFunc에 등록된 콜백함수: 마우스 버튼이 눌러지지 않은 상태에서 마우스가 움직이는 경우 호출

```
void glutMotionFunc(void (*func)(int x, int y));
void glutPassiveMotionFunc(void (*func)(int x, int y));
```

glutMotionFunc함수 혹은 glutPassiveMotionFunc함수의 콜백함수로 등록된 함수는 매개변수를 통해 다음 정보를 받아온다.

- button: GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON 중 하나의 값을 가지며 각각 마우스의 왼쪽, 가운데, 오른쪽 버튼에 해당하는 이벤트가 발생했음을 의미한다.
- state: GLUT_UP 혹은 GLUT_DOWN 중 하나의 값을 가지며, button이 어떤 상태에 놓여 있을 때 이벤트가 발생했는지 알려준다.
- x, y: 마우스 움직임 이벤트가 발생했을 당시 마우스의 위치를 의미한다.

2.5 마우스 휠 이벤트 처리

마우스 휠 이벤트를 처리할 콜백함수는 다음과 같이 정의되는 glutMouseWheelFunc함수를 이용하여 등록하면 된다.

```
void glutMouseWheelFunc(void (*func)(int wheel, int direction, int x, int y));
```

glutMouseWheelFunc함수의 콜백함수로 등록된 함수는 각 매개변수를 통해 다음과 같은 정보를 받아온다.

- wheel: 휠 버튼이 돌아간 횟수를 알려준다.
- direction: 휠 버튼이 돌아간 방향을 +1 혹은 -1로 알려준다.
- x, y: 마우스 휠 이벤트가 발생했을 당시 마우스의 위치를 의미한다.

2.6 키보드 입력 이벤트 처리

키보드 입력 이벤트를 처리할 콜백함수는 다음과 같이 정의되는 glutKeyboardFunc함수 또는 glutSpecialFunc함수를 이용하여 등록하면 된다. 두 콜백함수 등록의 차이점은 다음과 같다.

- glutKeyboardFunc등록된 콜백함수: 키보드 중 ASCII 문자에 해당하는 버튼을 누른 경우 호출
- glutSpecialFunc등록된 콜백함수: 키보드 중 ASCII 문자가 아닌 특수키(예: F1 키, PageDn 키, Home 키 등)에 해당하는 버튼을 누른 경우 호출

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

glutKeyboardFunc함수 혹은 glutSpecialFunc함수의 콜백함수로 등록된 함수는 매개변수를 통해 다음 정보를 받아온다.

- key: 눌려진 키보드 값을 의미한다. glutKeyboardFunc함수의 경우 ASCII 문자값을 가지며, glutSpecialFunc함수의 경우 키보드 특수키에 매핑된 정수값을 가진다.
- x, y: 키보드 입력 이벤트가 발생했을 당시 마우스의 위치를 의미한다.

2.7 백그라운드 처리 이벤트 처리

특별한 이벤트를 받지 않는 휴식시간(idle time)에 처리할 콜백함수는 다음과 같이 정의되는 glutIdleFunc함수를 이용하여 등록하면 된다. 보통 백그라운드 작업이나 연속적인 애니메이션을 수행하고자 할 때 활용된다.

```
void glutIdleFunc(void (*func)(void));
```