사용자 입력에 반응하는 그래픽스 프로그래밍 (심화)

김준호

Abstract

GLFW에서 윈도우 이벤트를 처리를 위해 콜백함수(callback function)를 등록하는 방법을 학습한다. 사용자 입력에 반응하는 그래픽스 프로그램을 작성하는 기법을 익힌다. 사용자 입력을 통해 물체의 색깔이나 위치를 바꾸는 방법을 이해한다.

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 GLFW에서의 이벤트 처리

GLFW는 운영체제와 무관하게 OpenGL 기반 윈도우 프로그래밍이 가능하도록 설계한 라이브러리이다. 따라서 대부분의 GUI 기반 운영체제에서 발생하는 이벤트를 다음과 같이 일반화하였다.

- 키보드 입력 이벤트
- 마우스 버튼 이벤트
- 마우스 움직임 이벤트
- 마우스 휠 이벤트
- 화면 크기 변경 이벤트

GLFW로 프로그래밍된 윈도우는 무한루프를 돌며 운영체제가 현재 윈도우에 전달하는 이벤트를 듣게 된다. GLFW는 윈도우가 전달하는 각종 이벤트에 반응할 수 있도록 설계되어 있지만, 기본적인 GLFW프로그램은 각 이벤트에 대해 아무런 반응을 보이지 않는다.

만일 프로그래머가 특정 이벤트에 대해 반응할 수 있는 GLFW프로그램을 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 GLFW에서 '화면 크기 변경 이벤트'를 처리하려면 다음과 같이 하면 된다.

```
// 2-1) 키보드 입력 이벤트를 처리할 콜백함수 key_callback 함수 선언
void key_callback(GLFWwindow* window, int key, int scancode, int action, int mods);

int main(int argc, char* argv[])
{
  // ...
  // 1) 키보드 입력 이벤트에 대한 콜백함수 key_callback 등록 (반드시 while 루프가 불리기 전에 등록!)
  glfwSetKeyCallback(window, key_callback);
  // ...
  while (!glfwWindowShouldClose(window))
  {

  }
  glfwTerminate();
  return  0;
}

// 2-2) 키보드 입력 이벤트를 처리할 콜백함수 key_callback 함수 구현
```

국민대학교 소프트웨어학부, junho@kookmin.ac.kr

```
void key_callback(GLFWwindow* window, int key, int scancode, int action, int mods)
{

}
```

3 GLFW에서 이벤트 처리를 위한 콜백함수

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

- 키보드 입력 이벤트 처리를 위한 콜백함수 등록: glfwSetKeyCallback(...)
- 마우스 버튼 이벤트 처리를 위한 콜백함수 등록: glfwSetMouseButtonCallback(...)
- 마우스 움직임 이벤트 처리를 위한 콜백함수 등록: glfwSetCursorPosCallback(...)
- 마우스 휠 이벤트 처리를 위한 콜백함수 등록: glfwSetScrollCallback(...)
- 화면 크기 변경 이벤트 처리를 위한 콜백함수 등록: glfwSetFramebufferSizeCallback(...)

콜백함수를 이용하는 자세한 방법은 GLFW input reference를 참고하도록 한다. 여기서는 주요 이벤트를 어떻게 콜백함수로 다루는지 개괄적으로 살펴보기로 한다.

3.1 키보드 입력 이벤트 처리

키보드 입력 이벤트 처리를 위한 콜백함수는 다음과 같이 정의되는 glfwSetKeyCallback함수를 이용하여 등록한다.

```
glfwSetKeyCallback(
  GLFWwindow* window,
  void(* GLFWkeyfun) (GLFWwindow* window, int key, int scancode, int action, int mods)
);
```

glfwSetKeyCallback함수에 등록된 콜백함수는 다음 정보를 매개변수로 건내받아 프로그래머의 목적에 맞게 사용할 수 있다.

- window: 이벤트를 받은 window 인스턴스
- key: 눌려진 혹은 때어진 키보드 키 값
- scancode: 키보드 키에 대한 시스템 별 스캔코드 값
- action: 키보드 키가 눌려진 상황(GLFW_PRESS)인지, 때어진 상황(GLFW_RELEASE)인지, 누르고 있는 상황(GLFW_REPEAT)인지에 대한 값
- mods: Shfit, Ctrl, Alt 키와 같은 모디파이어 키가 눌려졌는지 여부를 확인할 수 있는 값

3.2 마우스 버튼 이벤트 처리

마우스 버튼 이벤트 처리를 위한 콜백함수는 다음과 같이 정의되는 glfwSetMouseButtonCallback함수를 이용하여 등록한다.

```
glfwSetMouseButtonCallback(
  GLFWwindow* window,
  void(* GLFWmousebuttonfun) (GLFWwindow* window, int button, int action, int mods)
);
```

glfwSetMouseButtonCallback함수에 등록된 콜백함수는 다음 정보를 매개변수로 건내받아 프로그래머의 목적에 맞게 사용할 수 있다.

- window: 이벤트를 받은 window 인스턴스
- button: 눌려진 혹은 때어진 마우스 버튼 값
- action: 마우스 버튼이 눌려진 상황(GLFW_PRESS)인지, 때어진 상황(GLFW_RELEASE)인지에 대한 값
- mods: Shfit, Ctrl, Alt 키와 같은 모디파이어 키가 눌려졌는지 여부를 확인할 수 있는 값

3.3 마우스 움직임 이벤트 처리

마우스 움직임 이벤트 처리를 위한 콜백함수는 다음과 같이 정의되는 glfwSetCursorPosCallback함수를 이용하여 등록한다.

```
glfwSetCursorPosCallback(
  GLFWwindow* window,
  void(* GLFWcursorposfun) (GLFWwindow* window, double xpos, double ypos)
);
```

glfwSetCursorPosCallback함수에 등록된 콜백함수는 다음 정보를 매개변수로 건내받아 프로그래머의 목적에 맞게 사용할 수 있다.

- window: 이벤트를 받은 window 인스턴스
- xpos, ypos: 마우스 움직임 이벤트가 발생했을 당시 마우스 커서의 위치. 마우스 커서의 위치를 좌상단이 원점인 스크린 좌표계를 사용함.

3.4 마우스 휠 이벤트 처리

마우스 휠 이벤트 처리를 위한 콜백함수는 다음과 같이 정의되는 glfwSetScrollCallback함수를 이용하여 등록한다.

```
glfwSetScrollCallback(
  GLFWwindow* window,
  void(* GLFWscrollfun) (GLFWwindow* window, double xoffset, double yoffset)
);
```

glfwSetScrollCallback함수에 등록된 콜백함수는 다음 정보를 매개변수로 건내받아 프로그래머의 목적에 맞게 사용할 수 있다.

- window: 이벤트를 받은 window 인스턴스
- xoffset, yoffset: 마우스 휠 이벤트가 발생했을 당시 마우스 휠의 스크롤 오프셋

3.5 화면 크기 변경 이벤트 처리

화면 크기 변경 이벤트 처리를 위한 콜백함수는 다음과 같이 정의되는 glfwSetFramebufferSizeCallback함수를 이용하여 등록한다.

```
glfwSetFramebufferSizeCallback(
  GLFWwindow* window,
  void(* GLFWframebuffersizefun) (GLFWwindow* window, int width, int height)
);
```

여기서 눈여겨 봐야할 점은 glfwSetFramebufferSizeCallback함수의 입력이 void(∗ GLFWframebuffersizefun) (GLFWwindow∗ window, int width, int height)로 정의된 함수포인터라는 점이다. 즉, glfwSetFramebufferSizeCallback의해 콜백함수로 등록될 수 있는 함수는 반드시 입력으로 GLFWwindow∗형 1개와 int형 2개를 입력으로 받고 출력은 void여야 한다. 따라서 다음과 같은 framebuffer_size_callback 함수는 glutReshapeFunc에 의해 콜백함수로 등록될 수 있다.

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

void framebuffer_size_callback(GLFWwindow* window, int width, int height);  // 콜백함수 선언

int main(int argc, char* argv[])
{
  // main 함수 안...
  glfwSetFramebufferSizeCallback(framebuffer_size_callback);            // 콜백함수 등록
  // ...
}

void framebuffer_size_callback(GLFWwindow* window, 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
  // ...

}
```

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 화면 크기가 바뀌었기 때문에 OpenGL이 화면을 다시 그리도록 만들어야 한다. GLFW에서는 이러한 일이 자동으로 처리되기 때문에, 이 부분은 특별히 신경쓰지 않아도 된다.

4 연습문제

소스파일 hello_world.cpp를 수정해서 프로그램을 다음과 같이 변형해 보자.

1) a, s, d, w 키에 대한 키보드 이벤트를 처리할 수 있도록 키보드 처리 콜백함수의 내용을 바꿔보자.
2) 전역변수로 float형 변수 fovy를 새롭게 만들고, 마우스 휠 이벤트에 따라 fovy 값을 높이거나 낮추도록 하자.