Homework #4: Camera

김준호

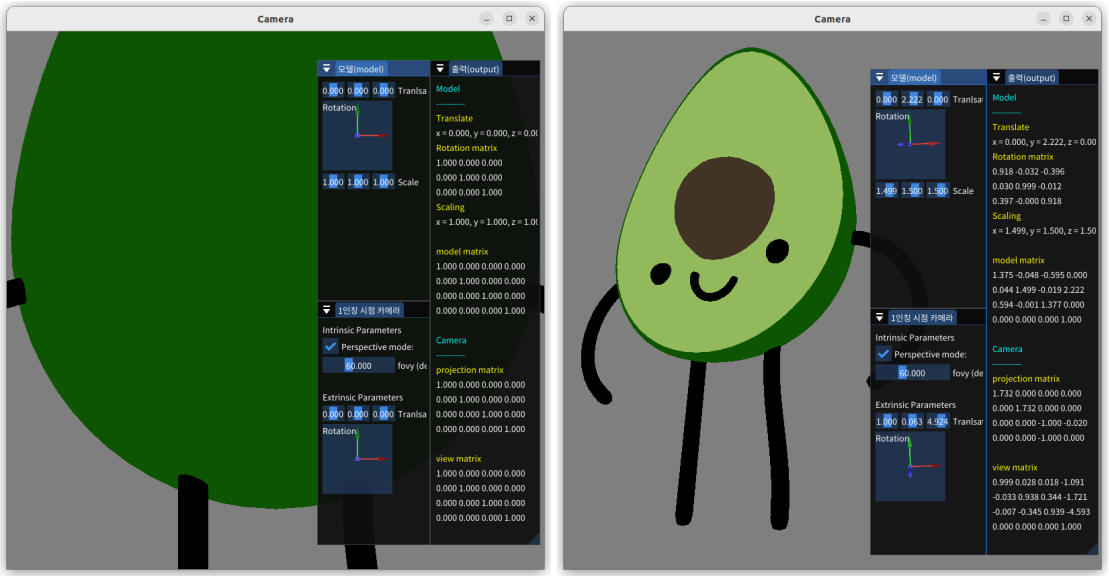
Abstract

본 과제에서는 사용자의 입력에 반응하여 카메라를 조작하는 그래픽스 프로그램을 작성한다. GLFW의 콜백 함수와 ImGui를 사용하여 사용자 입력을 통해 카메라의 위치, 회전, 시야각을 바꾸는 방법을 익힌다.

1 과제 소개

과제로 주어진 `skeleton` 코드를 수정하여, 사용자 입력에 따라 카메라 이동(translation), 회전(rotation), 시야각(field of view)을 변경할 수 있는 프로그램을 완성한다. GLFW의 콜백 함수를 사용하여 키보드 입력과 마우스 입력을 처리하고, 이에 더해 ImGui로 사용자 인터페이스 (User Interface)를 구성하는 방법을 실습한다.

2 과제 가이드



(a) 프로그램 처음 실행 시 (b) 완성된 프로그램

Fig. 1: 프로그램 예시

과제로 주어진 `skeleton` 코드를 컴파일하여 실행하면 Fig. 1(a)와 같은 프로그램을 마주할 수 있다. 현재 프로그램은 model matrix를 조작하는 기능만 포함되어 있다. 본 과제에서는 주어진 소스 코드를 수정하여 사용자 입력에 따라 view matrix와 projection matrix가 적절히 조작되는 Fig. 1(b)와 같은 프로그램을 완성하는 것이 목표이다. 완성된 프로그램의 구동 방식은 유튜브 영상(link)을 참고한다.

`main.cpp`와 `Camera.cpp`의 *`//TODO`* 부분을 완성하여 다음 기능들이 구현될 수 있도 프로그램을 완성한다. 키보드와 마우스를 이용한 카메라 조작은 FPS 게임의 카메라 조작 방식과 유사하게 구현된다.

- 키보드 `D/A` 키를 통한 카메라 $\pm x$축 방향 이동 조작
- 키보드 `S/W` 키를 통한 카메라 $\pm z$축 방향 이동 조작
- imGuIZMO.quat 위젯을 통한 카메라 회전 조작
- ImGui 위젯을 통한 카메라 perspective / orthographic 모드 토글링
- 창 크기를 바꿀 때, 종횡비(aspect ratio)를 반영한 장면 렌더링 (추가 점수)
- 마우스 scroll을 통한 카메라 zoom in/out 콘트롤 (추가 점수)

국민대학교 소프트웨어학부, junho@kookmin.ac.kr

`main.cpp`의 `init_scene()` 함수 내부를 보면 초기 카메라는 $(0, 0, 5)$ 지점에서 $+y$ 방향을 up 방향으로 가진 상태에서 $-z$ 방향을 바라보도록 설정되어 있다. 하지만 `main.cpp`와 `Camera.cpp`의 *//TODO* 부분들이 완성되어 있지 않기 때문에 Fig. 1(a)과 같이 카메라의 설정값이 반영되지 않은 상태로 장면이 렌더링 되고 있다.

키보드 입력 혹은 ImGui 입력에 맞게 카메라의 멤버 변수가 적절히 수정되도록 `main.cpp`와 `Camera.cpp`의 *//TODO*를 부분을 완성하여 Fig. 1(b)와 같은 프로그램을 작성하도록 한다.

과제와 관련된 GLM 함수들은 다음과 같다. 보다 자세한 방법은 LearnOpenGL의 카메라 튜토리얼 (link)을 참고한다.

```
template<typename T >
detail::tmat4x4< T > lookAt (detail::tvec3< T > const &eye,
                             detail::tvec3< T > const &center,
                             detail::tvec3< T > const &up)
template<typename T >
detail::tmat4x4< T > ortho (T const &left, T const &right,
                            T const &bottom, T const &top,
                            T const &zNear, T const &zFar)
template<typename T >
detail::tmat4x4< T > perspective (T const &fovy, T const &aspect, T const &near, T const &far)
```

추가 구현

GLFW의 콜백함수 기능을 통해 다음을 구현하면 추가 점수를 부여한다.

종횡비를 반영한 장면 렌더링

창의 크기가 바뀔 때마다 종횡비를 반영한 장면 렌더링이 되도록 프로그래밍 한다.

- 콜백함수에서 변경된 종횡비를 설정하는 코드를 작성한다.
- `glfwSetFramebufferSizeCallback`를 이용해 창 크기가 바뀔 때마다 호출될 콜백함수를 등록한다.

마우스 스크롤을 이용한 **zoom in/out**

마우스 스크롤에 따라 카메라가 zoom in/out 되도록 프로그래밍한다.

- 콜백함수에서 zoom in/out 정도를 설정하는 코드를 작성한다.
- `glfwSetScrollCallback`를 이용해 창 크기가 바뀔 때마다 호출될 콜백함수를 등록한다.

코드 조각

다음은 콜백함수 작성와 등록에 참고할 코드 조각이다.

```
// declaration of callback functions
void framebuffer_size_callback(GLFWwindow* window, int width, int height);
void scroll_callback(GLFWwindow* window, double xoffset, double yoffset);

// implementation of my callback functions
void framebuffer_size_callback(GLFWwindow* window, int width, int height)
{
  // TODO
}
void scroll_callback(GLFWwindow* window, double xoffset, double yoffset)
{
  // TODO
}

int main(void)
{
  // TODO: register my callback functions
  // glfwSetFramebufferSizeCallback(...);
  // glfwSetScrollCallback(...);

  // Loop until the user closes the window
  while (!glfwWindowShouldClose(window))
  {
    // ...
  }
}
```

3 과제 제출방법(매우 중요!!)

- 본 과제는 개인과제이며, 각자 자신의 코드를 완성하도록 한다.
- 공지된 마감 시간까지 과제 코드를 가상대학에 업로드하도록 한다.
- 과제 코드는 **Ubuntu LTS 18.04+에서 make 명령으로 컴파일 가능**하도록 작성한다.
- 과제 코드는 다음의 파일들을 하나의 압축파일로 묶어 **tar.gz** 파일 형식이나 표준 **zip**파일 형식으로만 제출하도록 한다. 이때, 압축파일의 이름은 반드시 'OOOOOOOO_HW04.tar.gz (OOOOOOOO은 자신의 학번)'과 같이 자신의 학번이 드러나도록 제출한다.

    1) 소스코드 및 리소스 파일들
    2) Makefile

- 과제에 관한 질문은 슬랙의 과제 Q&A 채널을 활용하도록 한다.