Hello Window

GLFW를 실행할 수 있는지 확인

|  |
| --- |
| #include <glad/glad.h>  #include <GLFW/glfw3.h> |

-#include할때 glad가 GLFW앞에 나와야함

-다른 OpenGL 헤더파일보다 먼저 GLAD를 include해야 하는데, GLAD안에 OpenGL헤터가 포함되어 있기 때문임

main함수 만들기 – GLFW창을 위한 인스턴스화 함수

|  |
| --- |
| int main()  {  glfwInit();  glfwWindowHint(GLFW\_CONTEXT\_VERSION\_MAJOR, 3);  glfwWindowHint(GLFW\_CONTEXT\_VERSION\_MINOR, 3);  glfwWindowHint(GLFW\_OPENGL\_PROFILE, GLFW\_OPENGL\_CORE\_PROFILE);  //glfwWindowHint(GLFW\_OPENGL\_FORWARD\_COMPAT, GL\_TRUE);    return 0;  } |

-glfwInit() : GLFW의 초기화, initialize

-glfwWindowHint() : 이 함수를 통해 GLFW를 구성할 수 있음

-[https://www.glfw.org/docs/latest/window.html#window\_hints](https://www.glfw.org/docs/latest/window.html/#window_hints)

-glfwWindowHint에 들어갈 옵션은 여기서 확인

-GLFW\_CONTEXT\_VERSION\_MAJOR : API의 유효한 주 버전

-GLFW\_CONTEXT\_VERSION\_MINOR : API의 유효한 부 버전

-GLFW\_OPENGL\_PROFILE : 프로파일 설정, 뒤에 CORE로 설정함

-GLFW\_OPENGL\_FORWARD\_COMPAT, GL\_TRUE) : Mac OS로 사용할 때에 사용해야할 함수

윈도우 객체 생성

|  |
| --- |
| GLFWwindow\* window = glfwCreateWindow(800, 600, "LearnOpenGL", NULL, NULL);  if (window == NULL)  {  std::cout << "Failed to create GLFW window" << std::endl;  glfwTerminate();  return -1;  }  glfwMakeContextCurrent(window); |

-glfwCreateWindow : 창 반환

-1, 2 매개변수 : 너비, 높이가 들어감

-3 매개변수 : 창 이름

-glfwMakeContextCurrent(window); : window를 현재 스레드의 기본 context로 사용

GLAD 초기화 (GLAD가 OpenGL함수를 포함하고 있기 때문에 먼저 초기화 해야함)

|  |
| --- |
| if (!gladLoadGLLoader((GLADloadproc)glfwGetProcAddress))  {  std::cout << "Failed to initialize GLAD" << std::endl;  return -1;  } |

Viewport

-OpenGL에게 랜더링 창의 크기를 알려줘야함

|  |
| --- |
| glViewport(0, 0, 800, 600); |

-1, 2 매개변수 : 왼쪽 하단 모서리의 위치

-3, 4 매개변수 : 너비, 높이 크기(픽셀단위), window크기와 같게 설정함

-사용자가 창의 크기를 바꾸면 Viewport도 조정되야함

|  |
| --- |
| void framebuffer\_size\_callback(GLFWwindow\* window, int width, int height)  {  glViewport(0, 0, width, height);  } |
| glfwSetFramebufferSizeCallback(window, framebuffer\_size\_callback); |

-glfwSetFramebufferSizeCallback : 창의 크기가 바뀌면 콜백되는 함수

Ready your engines

-화면이 바로 종료되는 것을 방지하기 위해 명시적으로 중지하는 명령을 받을 때 까지 이미지를 계속 그리고, 사용자 입력을 기다림

|  |
| --- |
| while(!glfwWindowShouldClose(window))  {  glfwSwapBuffers(window);  glfwPollEvents();  } |

-창을 닫으라는 명령을 받았는지 while시작에서 확인

-true라면 루프 실행이 중지, 프로그램이 닫아짐

-glfwPollEvents() : 이벤트(키보드 입력, 마우스 입력 등)가 트리거되는지 확인, window상태를 업데이트, 해당 함수를 콜백하는 함수

One last thing

-루프가 종료하자마자 할당된 GLFW 리소스를 정리하기 위한 함수

|  |
| --- |
| glfwTerminate();  return 0; |

Input

-여러 입력 이벤트에 관해 설명

-어떤 키(여기에선 escape버튼)가 눌렸는지 확인

|  |
| --- |
| void processInput(GLFWwindow \*window)  {  if(glfwGetKey(window, GLFW\_KEY\_ESCAPE) == GLFW\_PRESS)  glfwSetWindowShouldClose(window, true);  } |

-escape키를 누르면 WindowShoudClose를 true로 변경, GLFW를 닫음 -> while이 false로될 것임

-while문에 추가

|  |
| --- |
| while (!glfwWindowShouldClose(window))  {  processInput(window); //추가된 코드  glfwSwapBuffers(window);  glfwPollEvents();  } |

Rendering

-while문에서 렌더링 명령을 실행하기 위해 배치하려는 위치

|  |
| --- |
| // render loop  while(!glfwWindowShouldClose(window))  {  // input  processInput(window);  // rendering commands here  ...  // check and call events and swap the buffers  glfwPollEvents();  glfwSwapBuffers(window);  } |

-간단한 예시로 원하는 색을 넣으려고 함

-GL\_COLOR\_BUFFER\_BIT, GL\_DEPTH\_BUFFER\_BIT, GL\_STENCIL\_BUFFER\_BIT가 있으나, 색만 바꿀것이라 color만 사용할 것임

|  |
| --- |
| // render loop  while(!glfwWindowShouldClose(window))  {  // input  processInput(window);  // rendering commands here  glClearColor(0.2f, 0.3f, 0.3f, 1.0f);  glClear(GL\_COLOR\_BUFFER\_BIT);  // check and call events and swap the buffers  glfwPollEvents();  glfwSwapBuffers(window);  } |

-glClearColor(0.2f, 0.3f, 0.3f, 1.0f); : 색은 R, G, B, A 순으로 되어있음