Computer Graphics

Input Processing

1. Read the OS messages received via SDL events and process key/mouse inputs. Create a class or struct that allows you to query the input during any given frame (similar to Unity).
   a) Filter the events via their type to get the ones we need using **event.type**; relevant are:
      SDL_EventType::SDL_EVENT_KEY_UP,
      SDL_EventType::SDL_EVENT_KEY_DOWN,
      SDL_EventType::SDL_EVENT_MOUSE_BUTTON_UP,
      SDL_EventType::SDL_EVENT_MOUSE_BUTTON_DOWN,
      SDL_EventType::SDL_EVENT_MOUSE_MOTION
   b) Read the event data via **event.key** (keyboard key), **event.button** (mouse button) and **event.motion** (mouse movement)
   c) Store the input data in a way that it can be retrieved later (and clear it after every frame to prevent reading inputs from last frame!). This may best be achieved via **std::set** (allows storing a set of items) or **std::vector** (C++-style array that can grow). Feel free to use other solutions as well if they work better!
   d) Provide a method for querying inputs, e.g. with an "input" object:
      input.key_released(SDLK_Y); // Y was released (this frame)
      input.key_pressed(SDLK_A); // A key was pressed (this frame)
      input.key_held(SDLK_B); // B key is being held down (multiple frames)
      Mouse button events should be handled similarly, while mouse motion will simply provide x and y positions or deltas.

2. Offset the triangle vertices using keys (wasd or arrow keys) using your input method.
   a) To upload a vector to the GPU at runtime, put **layout (location = 16) uniform vec3 triangle_pos;** into your shader, right below the other location lines. Then you can use the **triangle_pos** variable to offset the vertex position.
   b) In our CPU code, use **glUniform3f(0, x, y, z);** to upload data to the graphics pipeline **AFTER** binding it! Fill x, y and z with your **float** data.