3D Graphics Programming
Lab 7: Input and Camera Movement

Karsten Pedersen
Department of Creative Technology

October 16, 2018

In this lab we are going to look at implementing a 1st person camera to move around a scene. Many of these same principles work for moving not just the camera around the scene but any objects. The first step it to handle keyboard input. You should remember this from your first year with SDL

```
while(SDL_PollEvent(&event))
{
  ...
  else if(event.type == SDL_KEYDOWN)
  {
    // event.key.keysym.sym
    // SDLK_LEFT
    // SDLK_RIGHT
    // SDLK_UP
    // SDLK_DOWN
  }
  else if(event.type == SDL_KEYUP)
  {
    ...
  }
}
```

Store a glm::vec3 position and glm::vec3 rotation that will refer to the cameras position and rotation respectively. Based on the input, attempt to move the camera position forward relative to the direction it is facing.

Note:
You are not moving the actual camera, just the glm::vec3 which refers to the position. You shouldn't see anything change on screen just yet.

The following code should help you find the forward vector.

```
// create arbitary matrix
glm::mat4 t(1.0f);

// rotate it by angle (the camera's Y rotation)
t = glm::rotate(t, glm::radians(angle), glm::vec3(0, 1, 0));

// move forward 1 unit
t = glm::translate(t, glm::vec3(0, 0, 1));

// apply to an initial position
glm::vec3 fwd = t * glm::vec4(0.0f, 0.0f, 0.0f, 1.0f);

// normalize to get the unit vector
fwd = glm::normalize(fwd);
```

Once you have this forward vector, just add it on to your camera position vector. You may like to multiply it by a magnitude (and preferably the delta time) so that the movement is faster or slower depending on what you need.
With this position in place, we simply upload create a "model" matrix from the position and rotation and upload it to the shader as the view matrix by simply inversing it.

Note:
Remember, we inverse the camera's "model" matrix before uploading to the shader as a view matrix because it isn't the actual camera that moves, it is actually everything we are drawing moves the opposite way to simulate a moving camera. You may have encountered this before in a simpler way with your 2D games.

```
glm::mat4 model(1.0f);
model = glm::translate(model, cameraPosition);
model = glm::rotate(model, glm::radians(angle), glm::vec3(0, 1, 0));
shader->setUniform("in_View", glm::inverse(model));
```

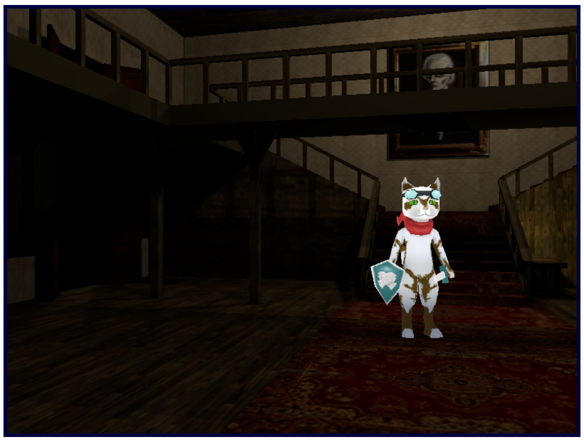

Figure 1: *You should now be able to move the camera and explore the scene*

Note:
Now try to implement smooth movement from the keyboard (no keyboard key repeat delay) and mouse input. Perhaps also try to make the cat run around in a circle.