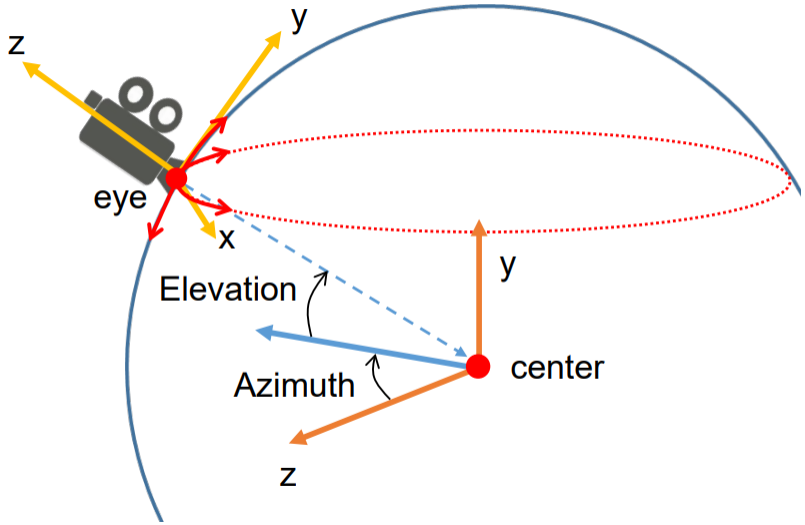
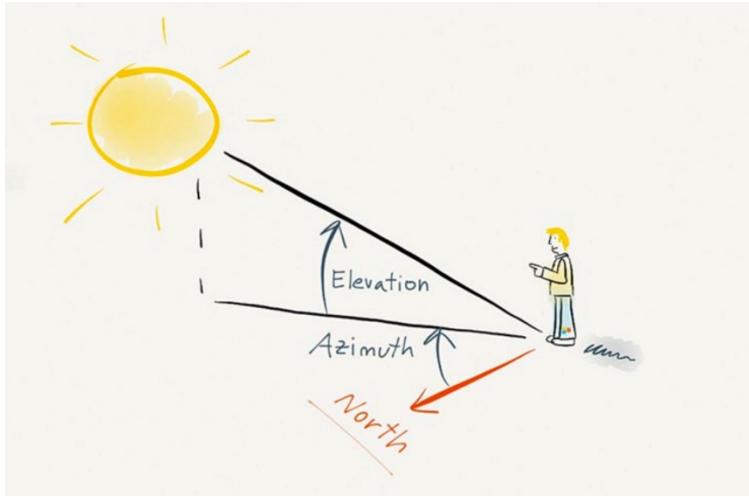
Computer Graphics
- Interactive Program2

Sung Soo Hwang

Camera Control

- Tumble Tool
  - Revolve the camera by varying the **azimuth** and **elevation** in a perspective view.
  - It is normally done by pressing Alt + Left mouse button

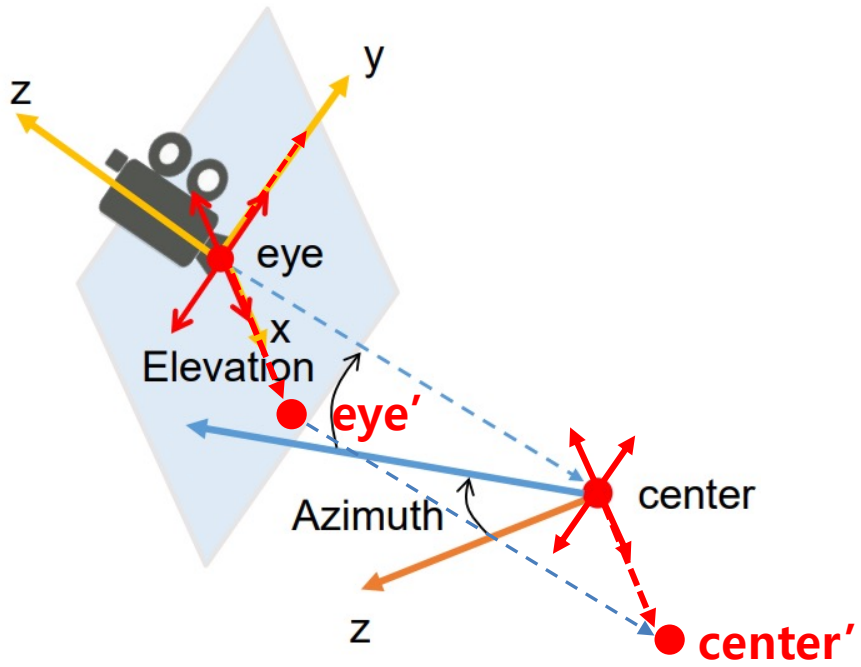




https://www.photopills.com/articles/understanding-azimuth-and-elevation

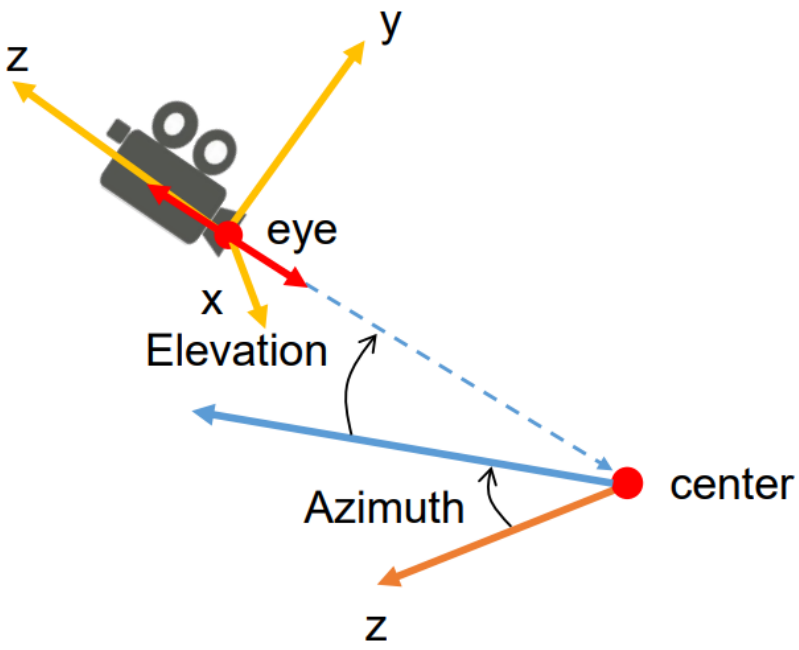
Camera Control

- Track Tool
  - Tracks the camera vertically and horizontally.
  - It is normally done by pressing Alt + Middle mouse button



Camera Control

- Dolly Tool
  - Moves the camera into the view, or backs the camera out of the view. It has an effect only on **perspective view**.
  - It is normally done by rolling mouse wheel.



Review: Callback functions in GLUT

- Mouse Callbacks
  - void cb_mouse(int button, int state, int x, int y)
    - button: represents which button is selected
    - state: represents the button state
    - x, y: mouse position
    - It is registered by glutMouseFunc(cb_mouse).

Constants	Val.	Description
GLUT_LEFT_BUTTON GLUT_MIDDLE_BUTTON GLUT_RIGHT_BUTTON	0 1 2	Left Mouse Button Middle Mouse Button Right Mouse Button
GLUT_DOWN GLUT_UP	0 1	Button pressed Button released

Review: Callback functions in GLUT

- Mouse Callbacks
  - void cb_motion(int x, int y)
  - void cb_passive_motion(int x, int y)
    - Called when the mouse moves within the window while one or more mouse buttons are pressed(cb_motion) or while no mouse buttons are pressed (cb_passive_motion)
    - x, y: mouse position
    - They are registered by glutMotionFunc(cb_motion) and glutPassiveMotionFunc(cb_passive_motion), respectively

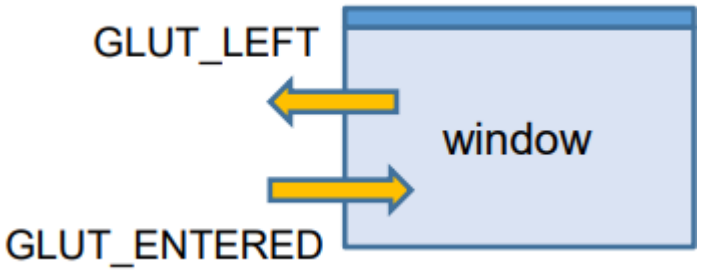
Review: Callback functions in GLUT

- Mouse Callbacks
  - void cb_entry(int state)
    - State: represents the entry status of the mouse pointer

GLUT_LEFT	The mouse pointer has left the window
GLUT_ENTERED	The mouse pointer has entered the window

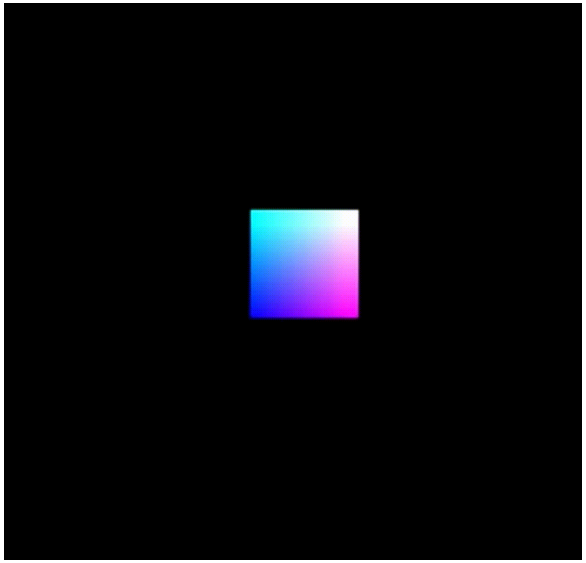
    - It is registered by glutEntryFunc(cb_entry).

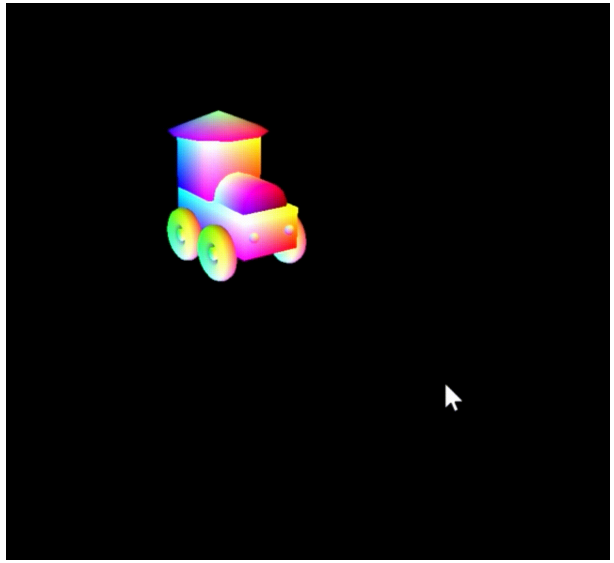
  - void cb_wheel(int wheel, int direction, int x, int y)
    - wheel: wheel number (which seems not well defined)
    - direction: +1 or -1 depending on the direction of the scroll
    - x, y: mouse position
    - It is registered by glutMouseWheelFunc(cb_wheel).



Practice

- According to the mouse interaction, the camera control must be done as follows:
  - **Mouse Left Button + Alt + Drag:**
    Works as the **Tumble tool** does
  - **Scroll up / down + Alt:**
    Works as the **Dolly tool** does





Practice

- How to track mouse movement and button clicks.
  - Global variables:

```
// Things of mouse interaction.
int button_pressed[3] = {GLUT_UP, GLUT_UP, GLUT_UP};
int mouse_pos[2] = { 0, 0 };
```

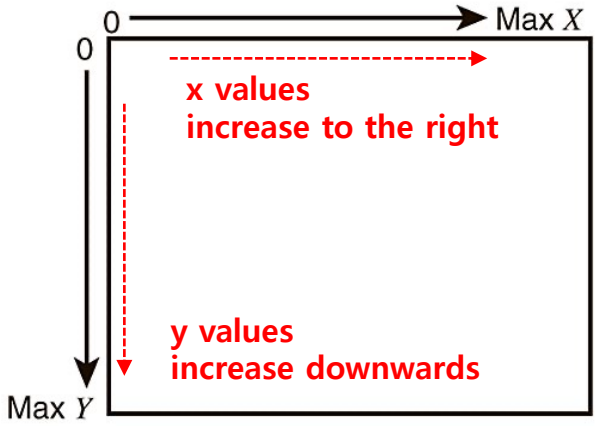
  - Mouse callbacks:

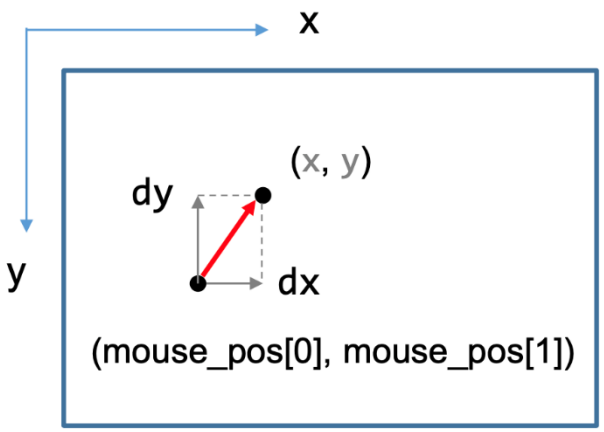
```
// Mouse callbacks
void mouse(int button, int state, int x, int y)
{
    button_pressed[button] = state;
    mouse_pos[0] = x;
    mouse_pos[1] = y;
}
```

Practice

- Motion callbacks:

```
void motion(int x, int y)
{
    using namespace glm;

    int modifiers = glutGetModifiers();
    int is_alt_active = modifiers & GLUT_ACTIVE_ALT;
    int is_ctrl_active = modifiers & GLUT_ACTIVE_CTRL;
    int is_shift_active = modifiers & GLUT_ACTIVE_SHIFT;

    int w = glutGet(GLUT_WINDOW_WIDTH);
    int h = glutGet(GLUT_WINDOW_HEIGHT);
    GLfloat dx = 1.f * (x - mouse_pos[0]) / w;
    GLfloat dy = -1.f * (y - mouse_pos[1]) / h;

    /* Add code here to deal with mouse motion */

    mouse_pos[0] = x;
    mouse_pos[1] = y;

    glutPostRedisplay();
}
```



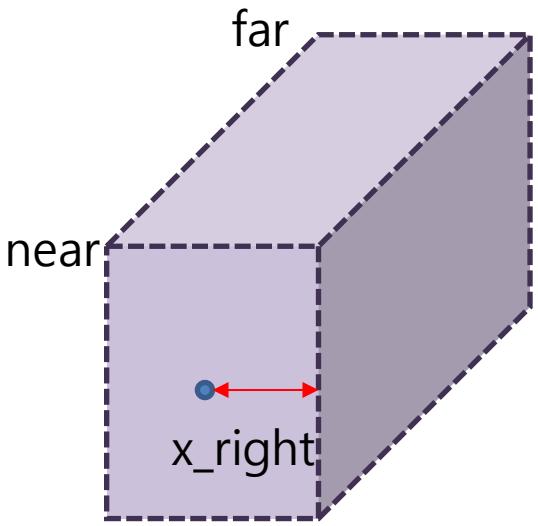


Practice

- Camera structure:

```
// Camera structure
struct Camera
{
    enum { ORTHOGRAPHIC, PERSPECTIVE};
    glm::vec3 eye;
    glm::vec3 center;
    glm::vec3 up;
    int projection_mode;
    float z_near;
    float z_far;
    float fovy;
    float x_right;
```

… See the next slides …



View volume of orthographic projection

Practice

- Camera structure:
  - Constructor

```
// Constructor
Camera() :
    eye(0, 0, 8),
    center(0, 0, 0),
    up(0, 1, 0),
    projection_mode(ORTHOGRAPHIC),
    z_near(0.01f),
    z_far(100.0f),
    fovy((float)(M_PI / 180.0 * (30.0))),
    x_right(1.2f)
{}
```

... See the next slides ...

Practice

- Camera structure:
  - Member functions

```
    // Member functions
    glm::mat4 get_viewing() { return glm::lookAt(eye, center, up); }
    glm::mat4 get_projection(float aspect)
    {
        glm::mat4 P;
        switch (projection_mode)
        {
        case ORTHOGRAPHIC:
            P = parallel(x_right, aspect, z_near, z_far);
            break;

        case PERSPECTIVE:
            P = glm::perspective(fovy, aspect, z_near, z_far);
            break;
        }
        return P;
    }
};
```

Practice

- **Mouse Left Button + Alt + Drag:**
  - Works as the **Tumble tool** does

```
// Tumble tool(Left mouse + Alt + Drag)
if (button_pressed[GLUT_LEFT_BUTTON] == GLUT_DOWN)
{
    if (is_alt_active)
    {
        vec4 disp(camera.eye - camera.center, 1);

        GLfloat alpha = 2.0f;
        mat4 V = camera.get_viewing();
        mat4 Rx = rotate(mat4(1.0f), alpha*dy, vec3(transpose(V)[0]));
        mat4 Ry = rotate(mat4(1.0f), -alpha * dx, vec3(0,1,0));
        mat4 R = Ry * Rx;
        camera.eye = camera.center + vec3(R*disp);
        camera.up = mat3(R) * camera.up;

    }

}
```

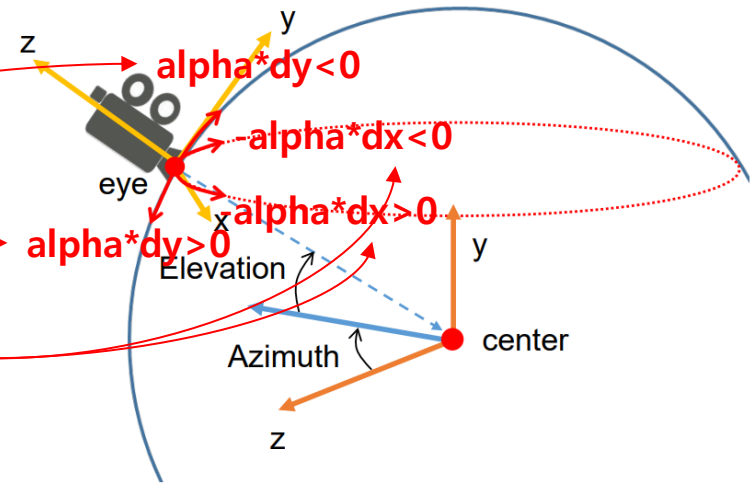
x-axis of the camera coordinate

$$V = \begin{bmatrix} U_x & V_x & W_x & P_{eyex} \\ U_y & V_y & W_y & P_{eyey} \\ U_z & V_z & W_z & P_{eyez} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Practice

- **Mouse Left Button + Alt + Drag:**
  - Works as the **Tumble tool** does

```
// Tumble tool(Left mouse + Alt + Drag)
if (button_pressed[GLUT_LEFT_BUTTON] == GLUT_DOWN)
{
    if (is_alt_active)
    {
        vec4 disp(camera.eye - camera.center, 1);

        GLfloat alpha = 2.0f;
        mat4 V = camera.get_viewing();
        mat4 Rx = rotate(mat4(1.0f), alpha*dy, vec3(transpose(V)[0]));
        mat4 Ry = rotate(mat4(1.0f), -alpha * dx, vec3(0,1,0));
        mat4 R = Ry * Rx;
        camera.eye = camera.center + vec3(R*disp);
        camera.up = mat3(R) * camera.up;

    }

}
```



Practice

- **Scroll up / down + Alt:**
  - Works as the **Dolly tool** does

```
// Mouse Wheel callbacks
void mouse_wheel(int wheel, int dir, int x, int y)
{
    int is_alt_active = glutGetModifiers() & GLUT_ACTIVE_ALT;

    if (is_alt_active) {
        glm::vec3 disp = camera.eye - camera.center;
        if (dir > 0)
            camera.eye = camera.center + 0.95f * disp;
        else
            camera.eye = camera.center + 1.05f * disp;
    }

    glutPostRedisplay();
}
```