Computer Graphics

5 - Lab - 3D Transformations, Vertex Processing 1

Yoonsang Lee
Hanyang University

Spring 2023

Outline

- 3D Affine Transformations
- 3D Affine Transformations with PyGLM Functions
- Vertex Processing in OpenGL
- glm.lookAt()

3D Affine Transformations

[Code] 1-affine-transform-3D

- Vertex shader

```
#version 330 core

layout (location = 0) in vec3 vin_pos;
layout (location = 1) in vec3 vin_color;

out vec4 vout_color;

uniform mat4 M;

void main()
{
    // 3D points in homogeneous coordinates
    vec4 p3D_in_hcoord = vec4(vin_pos.xyz, 1.0);

    gl_Position = M * p3D_in_hcoord;

    vout_color = vec4(vin_color, 1.);
}
```

* The full source code can be found at https://github.com/yssl/CSE4020

[Code] 1-affine-transform-3D

```
while not glfwWindowShouldClose(window):
    ...
    glUseProgram(shader_program)

    # current frame: I (world frame)
    I = np.identity(4)
    glUniformMatrix4fv(M_loc, 1, GL_TRUE, I)

    # draw current frame
    glBindVertexArray(vao_frame)
    glDrawArrays(GL_LINES, 0, 6)

    t = glfwGetTime()

    # rotation
    th = np.radians(t*90)
    R = np.array([[np.cos(th), -np.sin(th),0.,0.],
                  [np.sin(th),  np.cos(th),0.,0.],
                  [0.,          0.,        0.,1.],
                  [0.,          0.,        0.,1.]])

    # tranlation
    T = np.array([[1., 0., 0., np.sin(t)],
                  [0., 1., 0., .2],
                  [0., 0., 1., 0.],
                  [0., 0., 0., 1.]])
```

```
    # scaling
    S = np.array([[np.sin(t), 0., 0., 0.],
                  [0., np.sin(t), 0., 0.],
                  [0., 0., np.sin(t), 0.],
                  [0., 0., 0., 1.]])

    # shearing
    H = np.array([[1., np.sin(t), 0., 0.],
                  [0., 1., 0., 0.],
                  [0., 0., 0., 0.],
                  [0., 0., 0., 1.]])

    M = R
    # M = T
    # M = S
    # M = H
    # M = R @ T
    # M = T @ R

    # current frame: M
    glUniformMatrix4fv(M_loc, 1, GL_TRUE, M)

    # draw triangle w.r.t. the current frame
    glBindVertexArray(vao_triangle)
    glDrawArrays(GL_TRIANGLES, 0, 3)

    # draw current frame
    glBindVertexArray(vao_frame)
    glDrawArrays(GL_LINES, 0, 6)
    ...
```

3D Affine Transformations with PyGLM Functions

PyGLM Transformation Functions

- PyGLM provides a number of matrix transformation functions such as
  - rotate(), scale(), translate(), ...

- Refer to:
  - https://github.com/Zuzu-Typ/PyGLM/blob/master/wiki/function-reference/stable_extensions/matrix_transform.md

[Code] 2-affine-transform-3D-pyglm

```
while not glfwWindowShouldClose(window):
    ...
    glUseProgram(shader_program)

    # current frame: I (world frame)
    I = glm.mat4()
    glUniformMatrix4fv(M_loc, 1, GL_FALSE,
glm.value_ptr(I))

    # draw current frame
    glBindVertexArray(vao_frame)
    glDrawArrays(GL_LINES, 0, 6)

    t = glfwGetTime()

    # rotation
    th = np.radians(t*90)
    R = glm.rotate(th, glm.vec3(0,0,1))

    # tranlation
    T = glm.translate(glm.vec3(np.sin(t),
.2, 0.))
```

```
    # scaling
    S = glm.scale(glm.vec3(np.sin(t),
np.sin(t), np.sin(t)))

    M = R
    # M = T
    # M = S
    # M = R @ T
    # M = T @ R

    # current frame: M
    glUniformMatrix4fv(M_loc, 1,
GL_FALSE, glm.value_ptr(M))

    # draw triangle w.r.t. the current
frame
    glBindVertexArray(vao_triangle)
    glDrawArrays(GL_TRIANGLES, 0, 3)

    # draw current frame
    glBindVertexArray(vao_frame)
    glDrawArrays(GL_LINES, 0, 6)
    ...
```

* The full source code can be found at https://github.com/yssl/CSE4020

Quiz 2

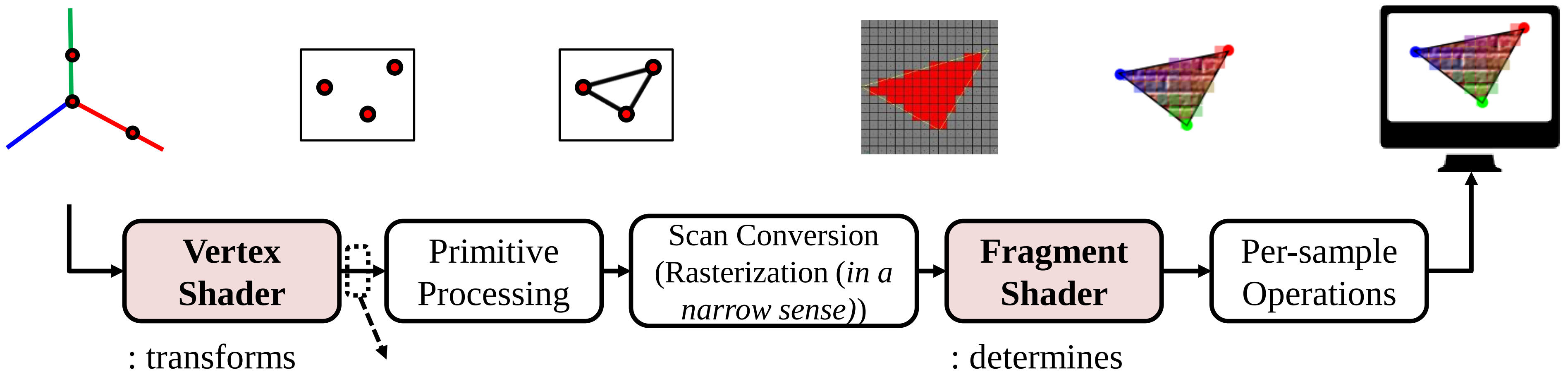
- Go to https://www.slido.com/
- Join #**cg-ys**
- Click "Polls"

- Submit your answer in the following format:
  - **Student ID: Your answer**
  - **e.g. 2021123456: 4.0**

- Note that your quiz answer must be submitted **in the above format** to receive a quiz score!

Vertex Processing in OpenGL

Recall: OpenGL Rendering Pipeline

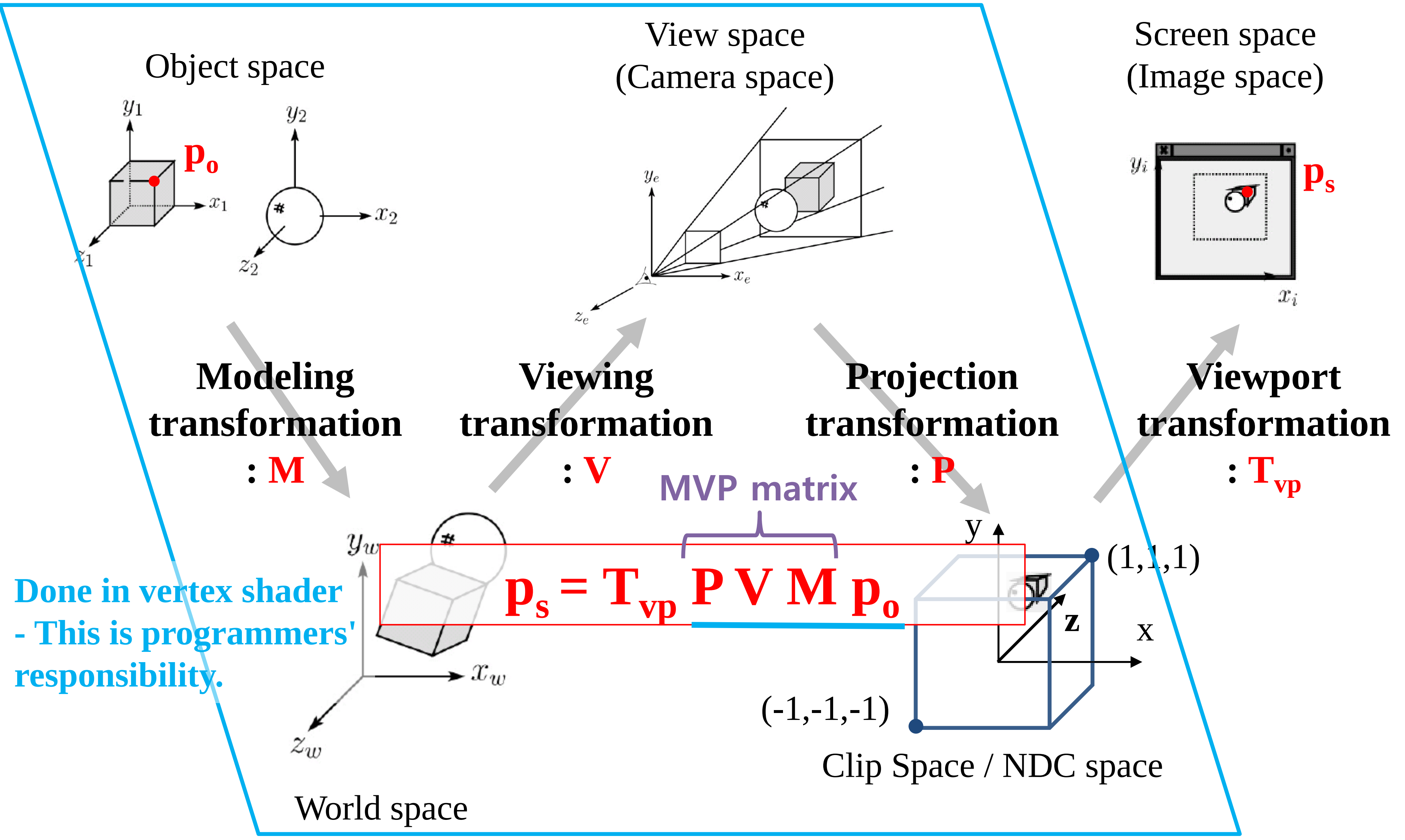


: transforms vertices to ***clip space***

* Vertices in ***clip space*** are transformed to ***screen space*** in ***Vertex Post-Processing***, omitted in this figure. We'll see this part in later lectures.

: determines color of each fragment with light & texture

Vertex Processing in OpenGL

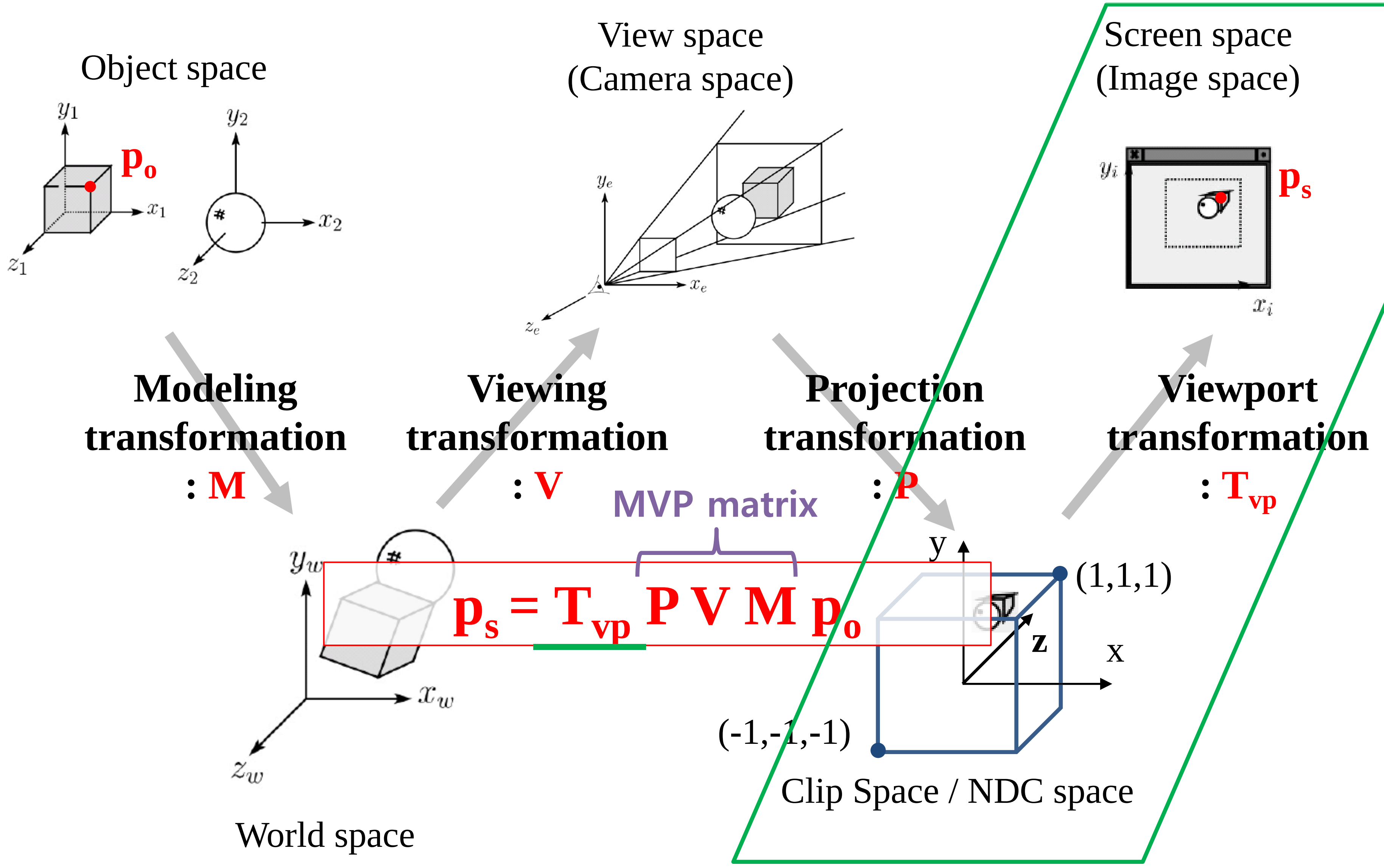

Object space
p_o
y_1
x_1
z_1
y_2
x_2
z_2
View space (Camera space)
y_e
x_e
z_e
Screen space (Image space)
y_i
x_i
p_s
Modeling transformation : M
Viewing transformation : V
Projection transformation : P
Viewport transformation : T_{vp}
MVP matrix
y_w
x_w
z_w
World space
y
x
z
(1,1,1)
(-1,-1,-1)
Clip Space / NDC space

$$p_s = T_{vp}\,\underbrace{\overbrace{P\,V\,M}^{\text{MVP matrix}}\,p_o}$$

Done in vertex shader
- This is programmers' responsibility.

Vertex Processing in OpenGL

Done internally - Programmers just configure this with glViewPort()



$$\mathbf{p_s} = \mathbf{T_{vp}}\,\underbrace{\mathbf{P\,V\,M}}_{\text{MVP matrix}}\,\mathbf{p_o}$$

glm.lookAt()

glm.lookAt()

- PyGLM provides the "lookat" function.
  - https://github.com/Zuzu-Typ/PyGLM/blob/master/wiki/function-reference/stable_extensions/matrix_transform.md

glm.lookAt(eye: *vec3*, **center**: *vec3*, **up**: *vec3*) **->** ***mat4x4***

Build a look at view matrix based on the default handedness.

[Code] 3-lookat

- Vertex shader

```
#version 330 core

layout (location = 0) in vec3 vin_pos;
layout (location = 1) in vec3 vin_color;

out vec4 vout_color;

uniform mat4 MVP; // just rename

void main()
{
    // 3D points in homogeneous coordinates
    vec4 p3D_in_hcoord = vec4(vin_pos.xyz, 1.0);

    gl_Position = MVP * p3D_in_hcoord;

    vout_color = vec4(vin_color, 1.);
}
```

* The full source code can be found at https://github.com/yssl/CSE4020

[Code] 3-lookat

```
...
g_cam_ang = 0.
g_cam_height = .1
...
def key_callback(window, key, scancode, action, mods):
    global g_cam_ang, g_cam_height
    if key==GLFW_KEY_ESCAPE and action==GLFW_PRESS:
        glfwSetWindowShouldClose(window, GLFW_TRUE);
    else:
        if action==GLFW_PRESS or action==GLFW_REPEAT:
            if key==GLFW_KEY_1:
                g_cam_ang += np.radians(-10)
            elif key==GLFW_KEY_3:
                g_cam_ang += np.radians(10)
            elif key==GLFW_KEY_2:
                g_cam_height += .1
            elif key==GLFW_KEY_W:
                g_cam_height += -.1
```

* The full source code can be found at https://github.com/yssl/CSE4020

[Code] 3-lookat

```
def main():
    ...
    while not glfwWindowShouldClose(window):
        # enable depth test (we'll see details later)
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        glEnable(GL_DEPTH_TEST)

        # projection matrix
        # use orthogonal projection (we'll see details later)
        P = glm.ortho(-1,1,-1,1,-1,1)

        # view matrix
        # rotate camera position with g_cam_ang / move camera up & down with 
g_cam_height
        V = 
glm.lookAt(glm.vec3(.1*np.sin(g_cam_ang),g_cam_height,.1*np.cos(g_cam_ang)), 
glm.vec3(0,0,0), glm.vec3(0,1,0))

        # current frame: P*V*I (now this is the world frame)
        I = glm.mat4()
        MVP = P*V*I
        glUniformMatrix4fv(MVP_loc, 1, GL_FALSE, glm.value_ptr(MVP))

        # draw current frame
        glBindVertexArray(vao_frame)
        glDrawArrays(GL_LINES, 0, 6)
```

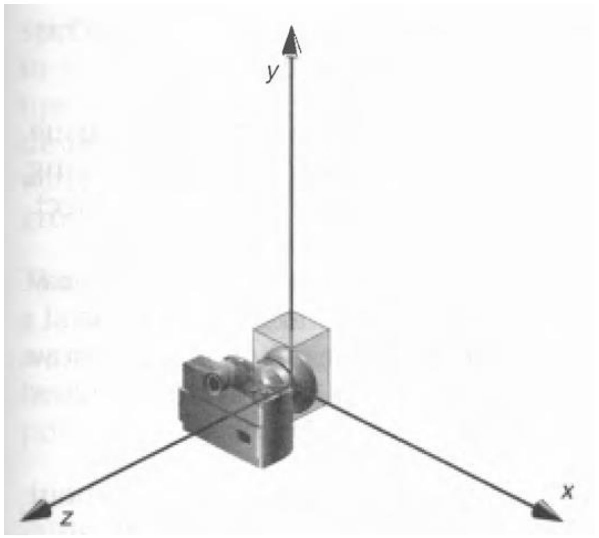
[Code] 3-lookat

```
...

M = R
# M = T
# M = S
# M = R @ T
# M = T @ R

# current frame: P*V*M
MVP = P*V*M
glUniformMatrix4fv(MVP_loc, 1, GL_FALSE, glm.value_ptr(MVP))

# draw triangle w.r.t. the current frame
glBindVertexArray(vao_triangle)
glDrawArrays(GL_TRIANGLES, 0, 3)

# draw current frame
glBindVertexArray(vao_frame)
glDrawArrays(GL_LINES, 0, 6)
```

[Code] 3-lookat

- Try changing those modeling transformations so that they are "really" applied in 3D space.
  - e.g. rotation about x-axis, ...

- The **default OpenGL camera** is:
- located at the **origin**
- looking in **negative z direction**
- its up direction is **positive y**



Quiz 3

- Go to https://www.slido.com/
- Join #**cg-ys**
- Click "Polls"

- Submit your answer in the following format:
  - **Student ID: Your answer**
  - **e.g. 2021123456: 4.0**

- Note that your quiz answer must be submitted **in the above format** to receive a quiz score!

Time for Assignment

- Project 1
  - Due: 23:59, April 16, 2023 (NO SCORE for late submissions!)

- Let's start today's assignment.
- TA will guide you.