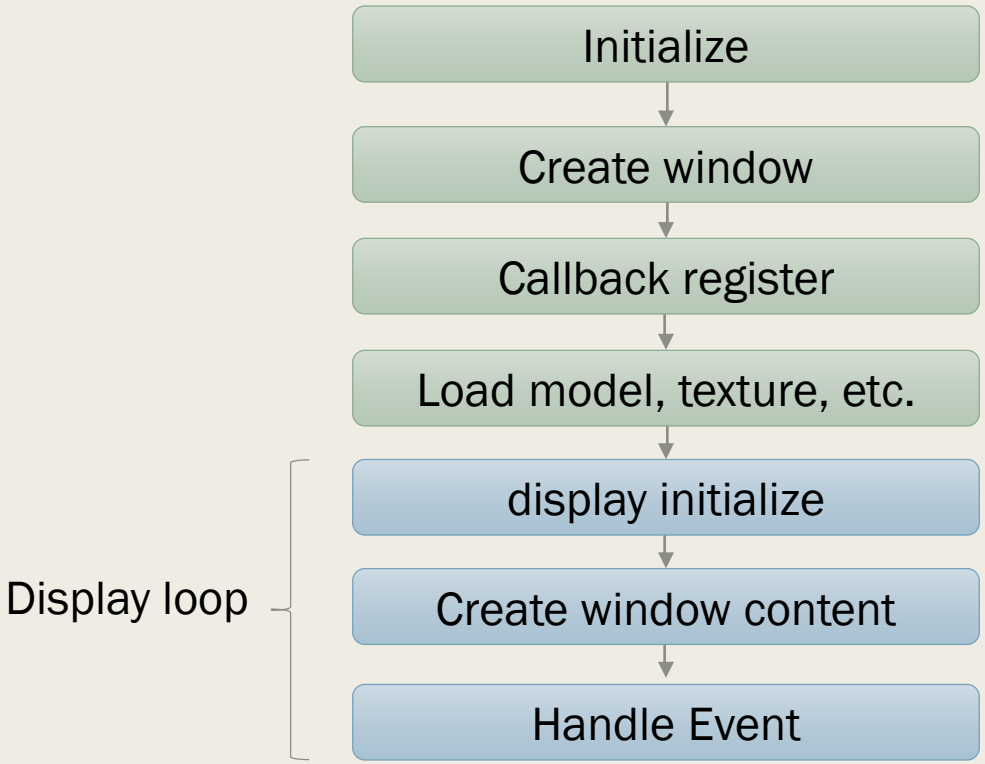
HW1 TUTORIAL

IDE & Kit

- Visual Studio 2019 – Community (download here)
- GLFW (provided in zip)
  - An Open Source, multi-platform library for OpenGL
  - Provides a simple API for creating windows, receiving input and, etc.
- GLAD(provided in zip)
  - An openGL loading library that loads pointers to OpenGL functions at runtime
- GLM(provided in zip)
  - Math library for OpenGL

Architecture



Initialize and window

- int glfwInit()
  - Initialize GLFW
  - Return GLFW_TRUE while succeed, else GLFW_FALSE
- void glfwWindowHint(int hint, int value)
  - Window setting for next window creation
  - In this homework,

    we're using openGL 3.3 core profile

```
glfwInit();
glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
```

Initialize and window

- GLFWwindow* glfwCreateWindow(int width, int height, const char* title,
  GLFWmonitor* monitor, GLFWmonitor* share)
  - Create window with specified width, height and title
  - monitor: monitor used for full screen mode, NULL for window mode
  - share: the window to share resource with, NULL to not share resource
  - Return the handle of the created window, or NULL if an error occurred

```
GLFWwindow* window = glfwCreateWindow(800, 600, "HW1", NULL, NULL);
if (window == NULL)
{
    std::cout << "Failed to create GLFW window" << std::endl;
    glfwTerminate();
    return -1;
}
```

Initialize and window

- void glfwMakeContextCurrent(GLFWwindow* window)
  - Make context current for the calling thread
- GLFWframebuffersizefun glfwSetFramebufferSizeCallback(GLFWwindow* window, GLFWframebuffersizefun cbfun)
- GLFWkeyfun glfwSetKeyCallback(GLFWwindow* window, GLFWkeyfun cbfun)
  - Register callback function for window resize and key event
- void glfwSwapInterval(int interval)
  - set the number of screen updates to wait from the time glfwSwapBuffers() was called before swapping the buffers and returning

```
glfwMakeContextCurrent(window);
glfwSetFramebufferSizeCallback(window, framebuffer_size_callback);
glfwSetKeyCallback(window, keyCallback);
glfwSwapInterval(1);
```

Initialize and window

- gladLoadGLLoader((gladloadproc)glfwgetprocaddress)
  - Initialize GLAD to get OpenGL function pointer

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

Depth test

To prevent occluded faces being rendered, we need to enable depth testing

- void glEnable(GL_DEPTH_TEST)

While depth test enabled, OpenGL tests depth value of each fragment against the content in the depth buffer. If the test passes, the fragment is rendered. If not, the fragment is discarded

- void glDepthFunc(GLenum func)

```
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LEQUAL);
```

Specify how the test is performed.

func: GL_NEVER, GL_LESS, GL_EQUAL, GL_LEQUAL, GL_GREATER, GL_GEQUAL, GL_NOTEQUAL, GL_ALWAYS

GL_LEQUAL: test passes if fragment depth <= depth stored in buffer

Face culling

Face culling reduces the number of faces rendered by discarding face not visible

- void glEnable(GL_CULL_FACE)

```
glEnable(GL_CULL_FACE);
glFrontFace(GL_CCW);
glCullFace(GL_BACK);
```

Tell OpenGL to enable face culling

- void glFrontFace(GLenum mode)

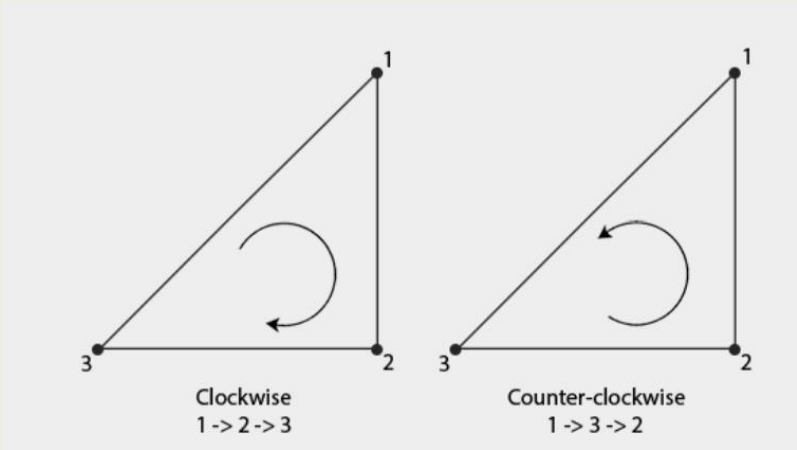
mode: GL_CW, GL_CCW

Faces with specified ordered vertices are defined front

- void glCullFace(GLenum mode)

mode: GL_FRONT, GL_BACK, GL_FRONT_AND_BACK

Cull specified faces



Display loop

Before we start to draw, we first need to clear the color buffer and depth buffer

- void glClearColor(GLfloat red, GLfloat green, GLfloat blue, GLfloat alpha)

Set the color value OpenGL uses to reset color buffer

- void glClear(GLbitfield mask)

Clear specified buffer

mask: GL_COLOR_BUFFER_BIT: clear c

GL_DEPTH_BUFFER_BIT: clear depth buffer

```
glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

Draw a model

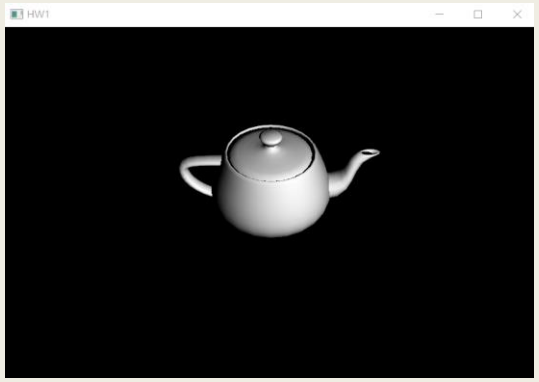
- void DrawModel(const char* target, glm::mat4 M, glm::mat4 V, glm::mat4 P)

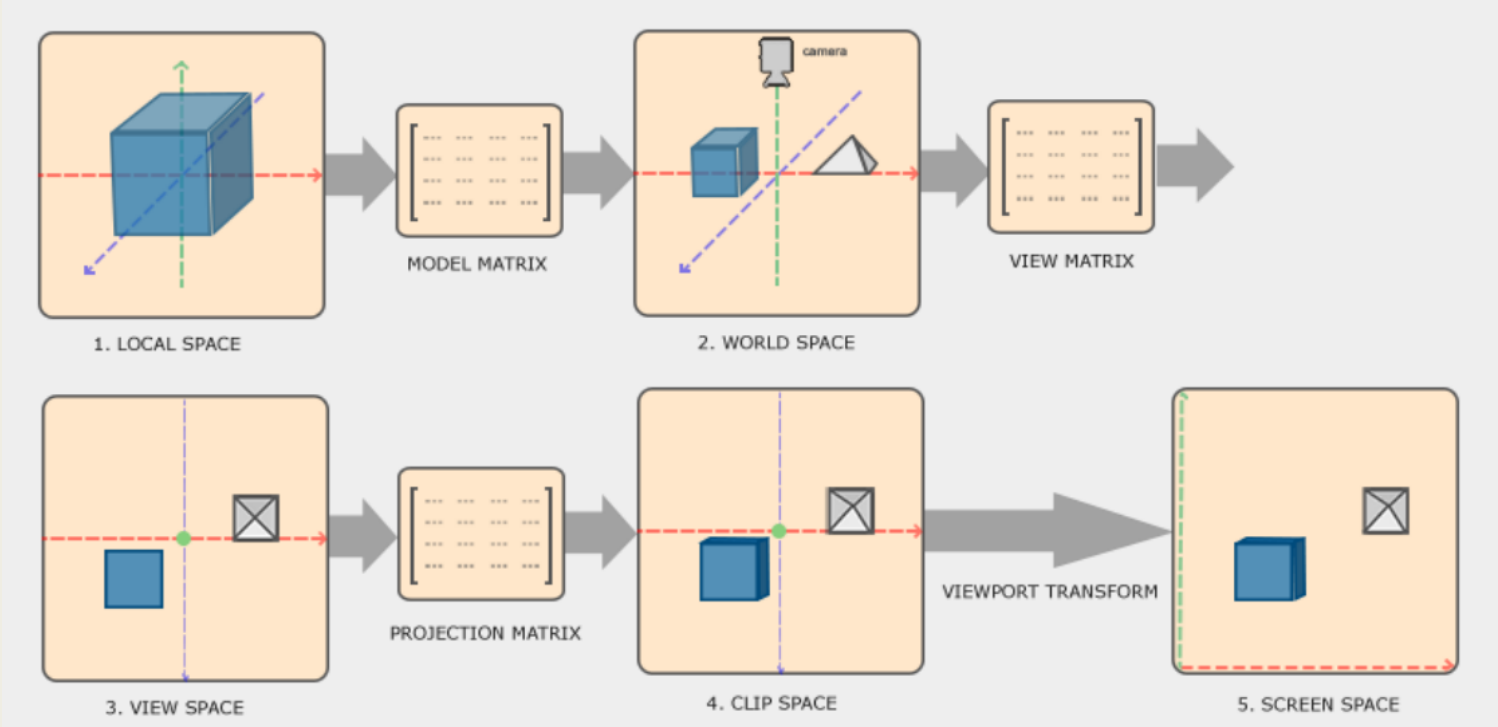
Draw the target model (teapot, base, cat, ball)

M: model matrix.

V: view matrix.

P: projection matrix



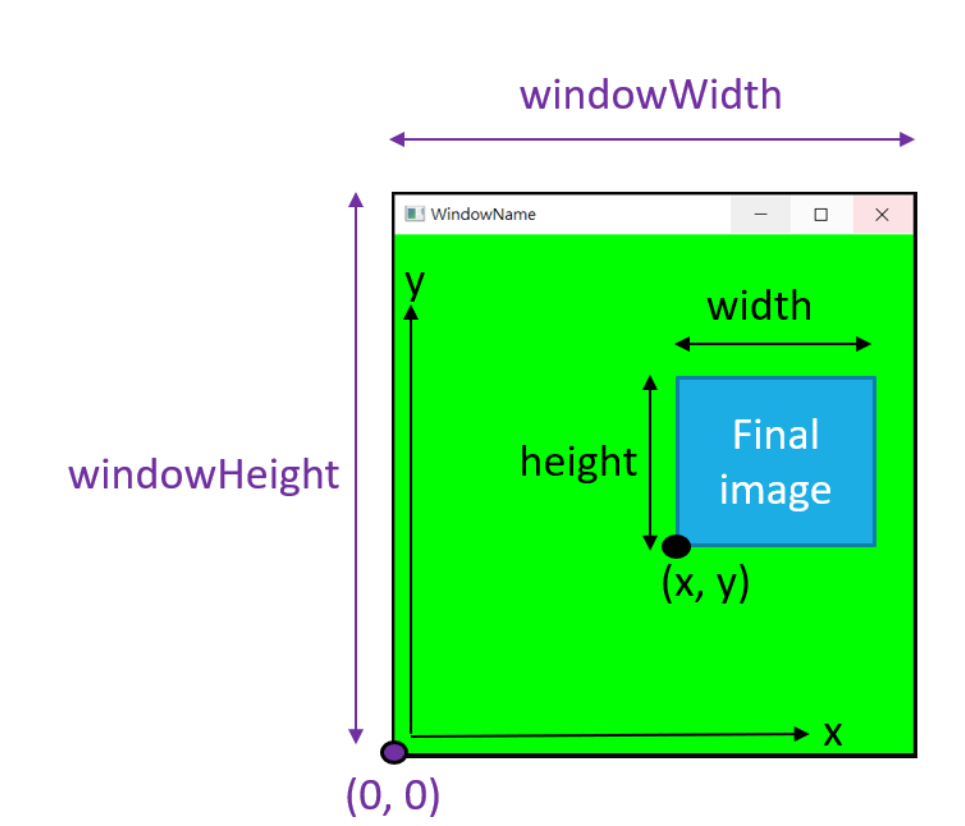


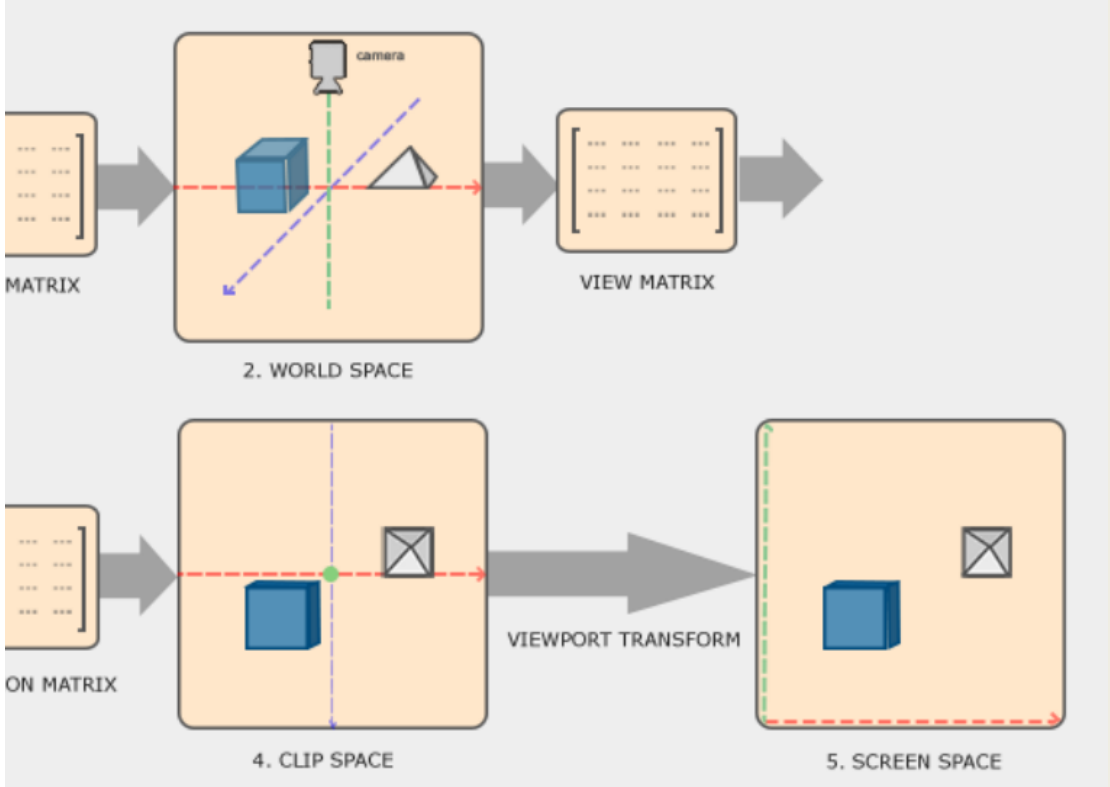

```
DrawModel("teapot", model, view, prespective);
```

Draw a model

- void glViewport(GLint x, GLint y, GLint width, GLint height)

Specify the viewport rectangle



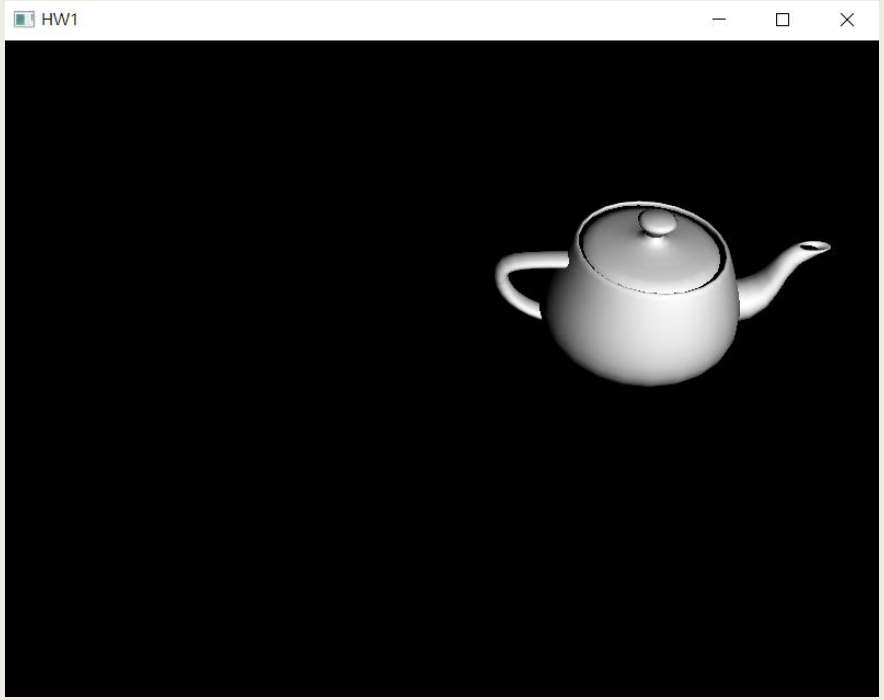


Model matrix

- glm::translate(glm::mat4 M, glm::vec3 translation)

Returns M * (translation matrix)

```
glm::mat4 model = glm::mat4(1.0f);
model = glm::translate(model, glm::vec3(4, 0, 0));
DrawModel("teapot", model, view, prespective);
```

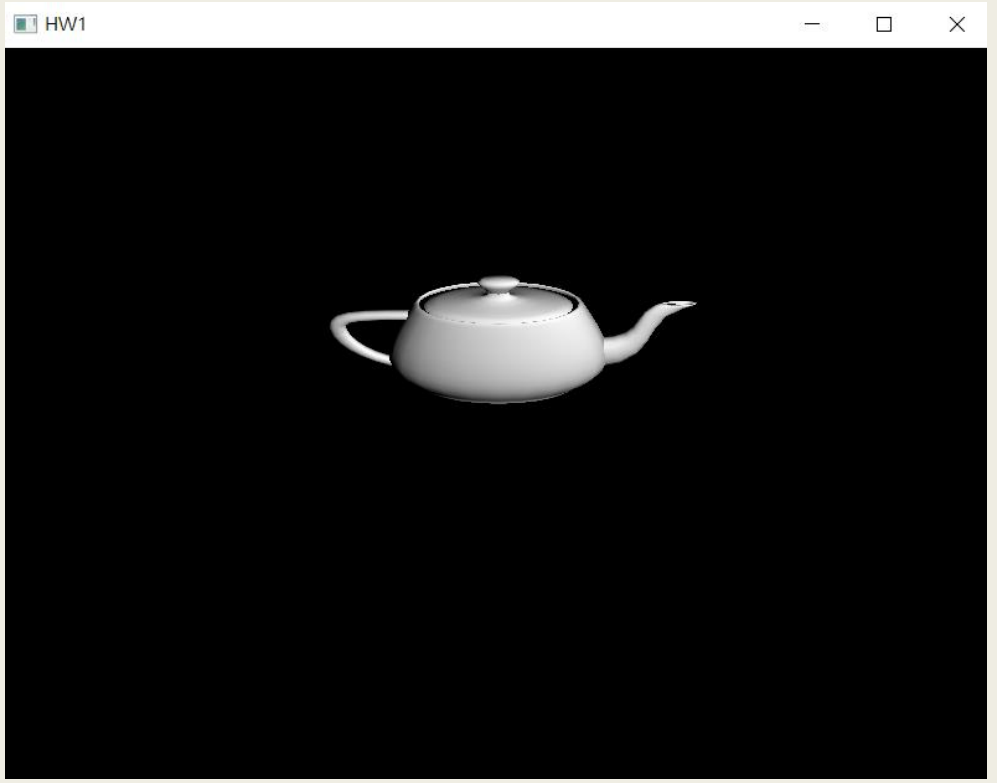


Model matrix

- glm::scale(glm::mat4 M, glm::vec3 scale)

Returns M * (scale matrix)

```
glm::mat4 model = glm::mat4(1.0f);
model = glm::scale(model, glm::vec3(1, 0.8, 0.4));
DrawModel("teapot", model, view, prespective);
```



Model matrix

- glm::rotate(glm::mat4 M, GLfloat angle, glm::vec3 axis)

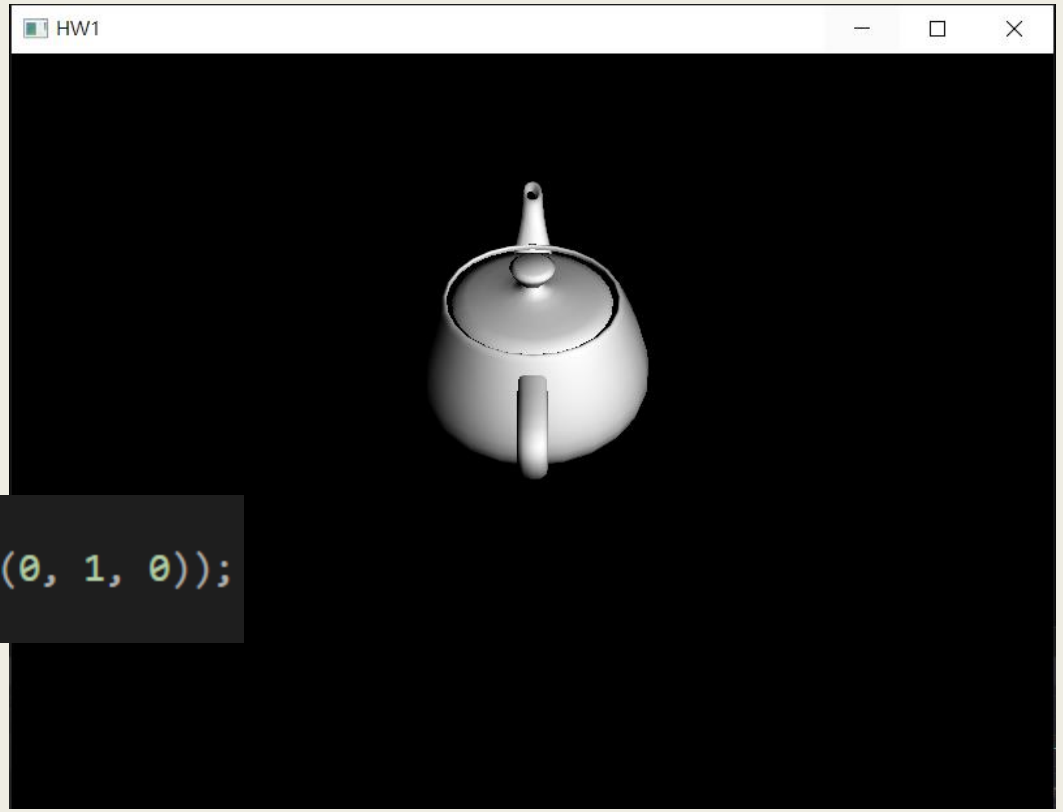
Returns M * (rotation matrix)

The rotation matrix rotate angle about axis in radians

- glm::radians(GLfloat degree)

Transform degree to radian

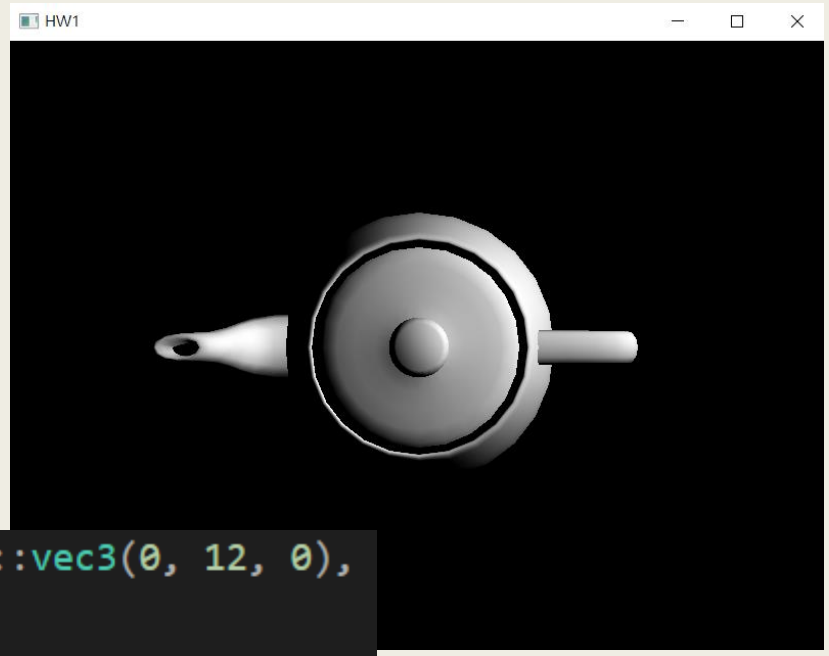
```
glm::mat4 model = glm::mat4(1.0f);
model = glm::rotate(model, glm::radians(90.0f), glm::vec3(0, 1, 0));
DrawModel("teapot", model, view, prespective);
```

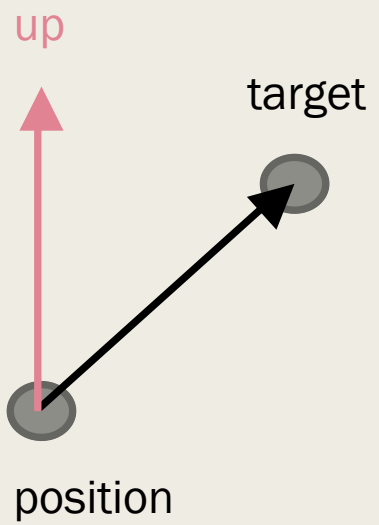


View matrix

- glm::lookAt(glm::vec3 position, glm::vec3 target, glm::vec3 up)

Returns view matrix with camera at position looking at target with up vector






```
glm::mat4 view = glm::lookAt(glm::vec3(0, 12, 0),
    glm::vec3(0, -1, 0),
    glm::vec3(1, 0, 0));
DrawModel("teapot", model, view, prespective);
```

Projection matrix

- glm::perspective(GLfloat fov, GLfloat aspect, GLfloat near, Glfloat far)

Returns perspective projection matrix with above parameters

fov: specify Field of View in radians

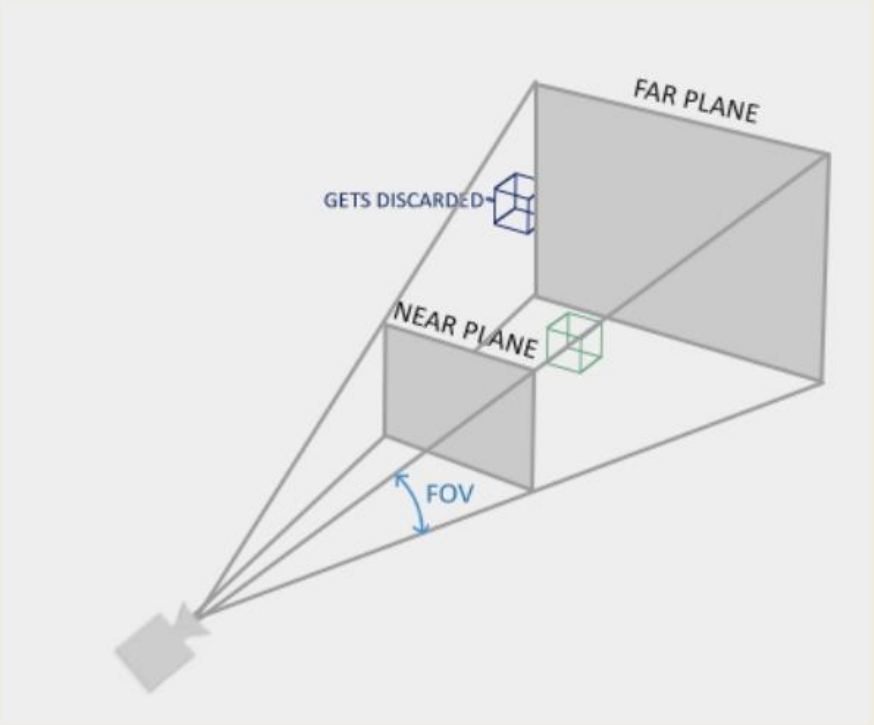
aspect: specify aspect ratio of the scene

near: specify near plane

far: specify far plane

Coordinates in front of near plane or behind far plane will not be drawn

```
glm::mat4 prespective = glm::perspective(
    glm::radians(30.0f),
    (float)windowWidth / (float)windowHeight,
    0.1f, 100.0f);
DrawModel("teapot", model, view, prespective);
```



Display loop

- void glfwSwapBuffers(GLFWwindow* window)

Swap buffer at the end of display loop

- void glfwPollEvent()

Handle the event occurred while rendering the frame, if any

```
glfwSwapBuffers(window);
glfwPollEvents();
```

Key callback

```
void keyCallback(GLFWwindow* window, int key, int scancode, int action, int mods)
{
    if (key == GLFW_KEY_ESCAPE && action == GLFW_PRESS)
        glfwSetWindowShouldClose(window, true);
}
```

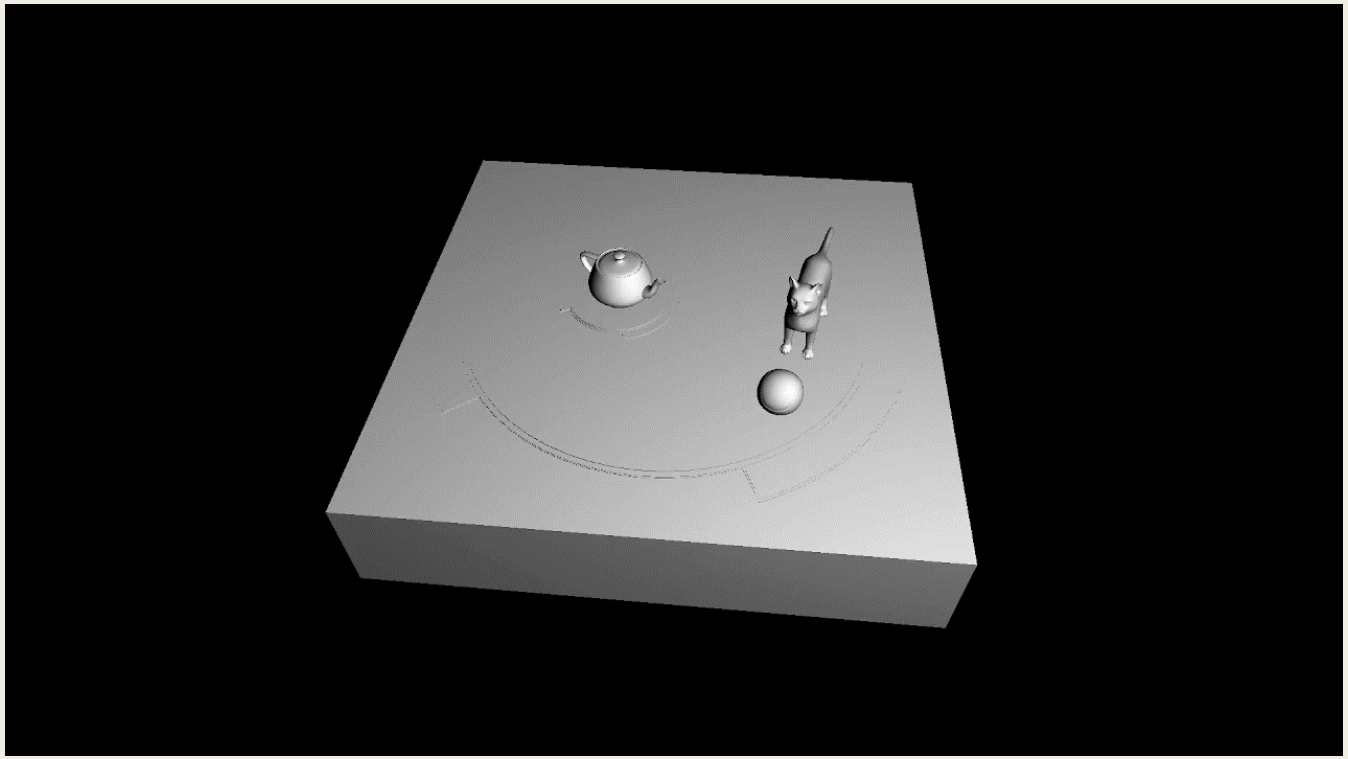
Previously the above function is registered for key callback. We can check for key events and act correspondingly.

The above function set WindowShouldClose to true when escape key is pressed, which exit the display loop.

The full list of key can be found here.

The full list of action and their effects can be found here.

Homework 1 – music box



Homework 1

Camera:
Position: (0, 12, 12)
Target: (0, -1, 0)
Up: (0, 1, 0)
FoV: 45.0
near: 0.1
far: 100.0

Base:
Rotate about +y axis -0.2 degree/frame
Scale (1.25, 1, 1.25)

Teapot disk: (parent: Base)
Rotate about +y axis -1.0 degree/frame
Scale (1, 1, 1) to original size

Disk: (parent: Base)
Rotate about +y axis -0.5 degree/frame
Scale (4, 1, 4) to original size

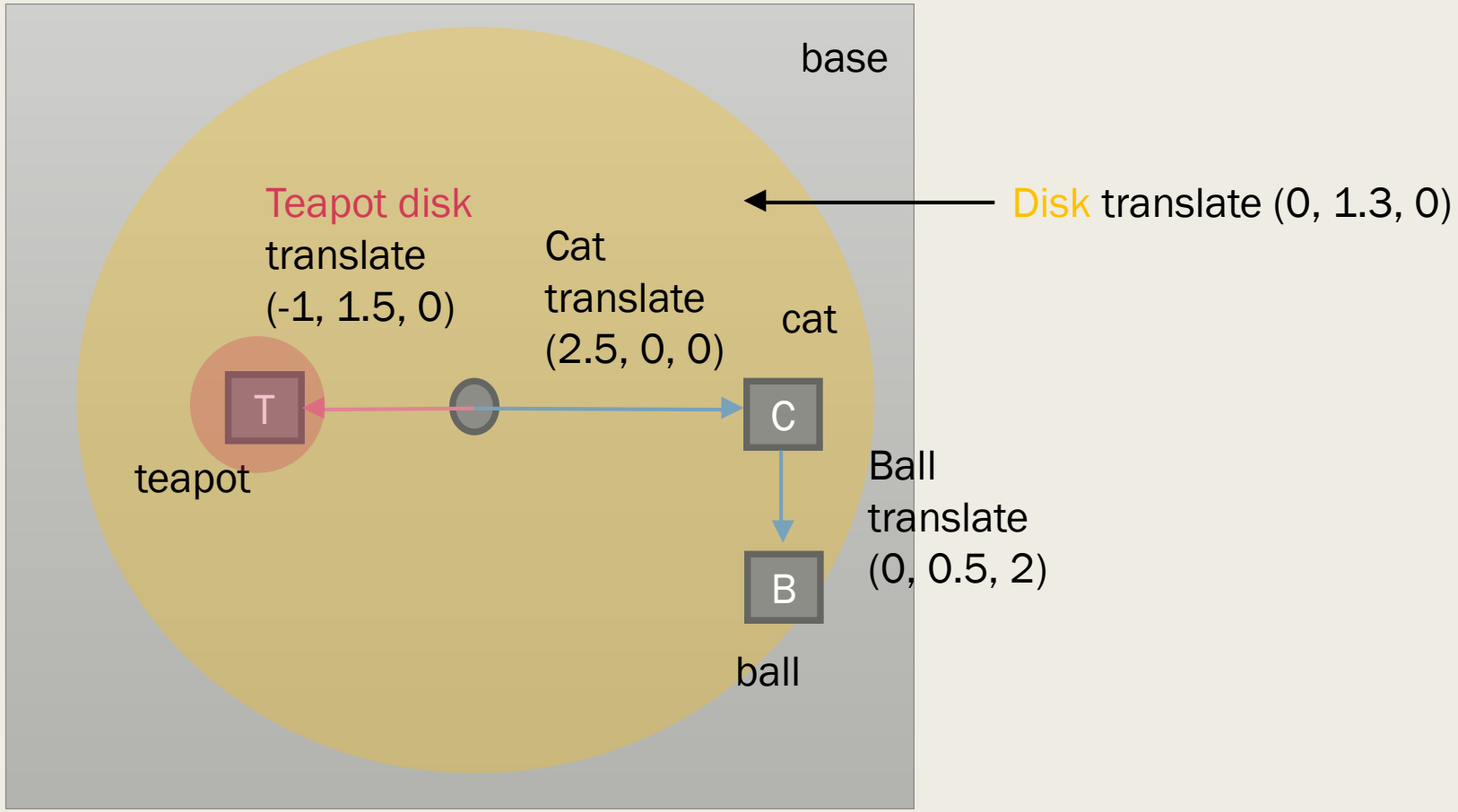
Teapot: (parent: Teapot disk)
Scale (0.3, 0.3, 0.3) to original size

Cat: (parent: Disk)
Scale (1, 1, 1) to original size

ball: (parent: Cat)
Rotate about +x axis 1.2 degree/frame
Scale (1.2, 1.2, 1.2) to original size

Keyboard input:
Press 1 to double the rotation speed of the disk (press twice, 4 times faster)
Press 2 to half the rotation speed of the disk (press twice, 1/4 slower)
This will also affect its children (cat, ball)

Homework 1 – relative position



Homework 1 - score

Depth testing (pass if less or equal)– 5%

Face culling (counter-clockwise as front, cull back)– 5%

Camera and perspective – 5%

Base (all transform must be correct) – 10%

Teapot disk (all transform must be correct) – 5%

Teapot (all transform must be correct) – 10%

Disk (all transform must be correct) – 5%

Cat (all transform must be correct) – 10%

Ball (all transform must be correct) – 10%

All 4 model correct – 20%

Keyboard input– 15%

Homework 1 - submission

- Deadline: 2022/10/18 23:59:59
- 10% penalty for each week late
  - *Final score = original score * 0.9 for less then a week late (10/19 ~ 10/25)*
  - *Final score = original score * 0.8 for one week late (10/25 ~ 10/31)*
  - *So on…*
- Zip and upload visual studio project on E3
- Zip name: studentID_HW1.zip

Reference

- https://learnopengl.com/
- https://www.glfw.org/docs/3.3/index.html