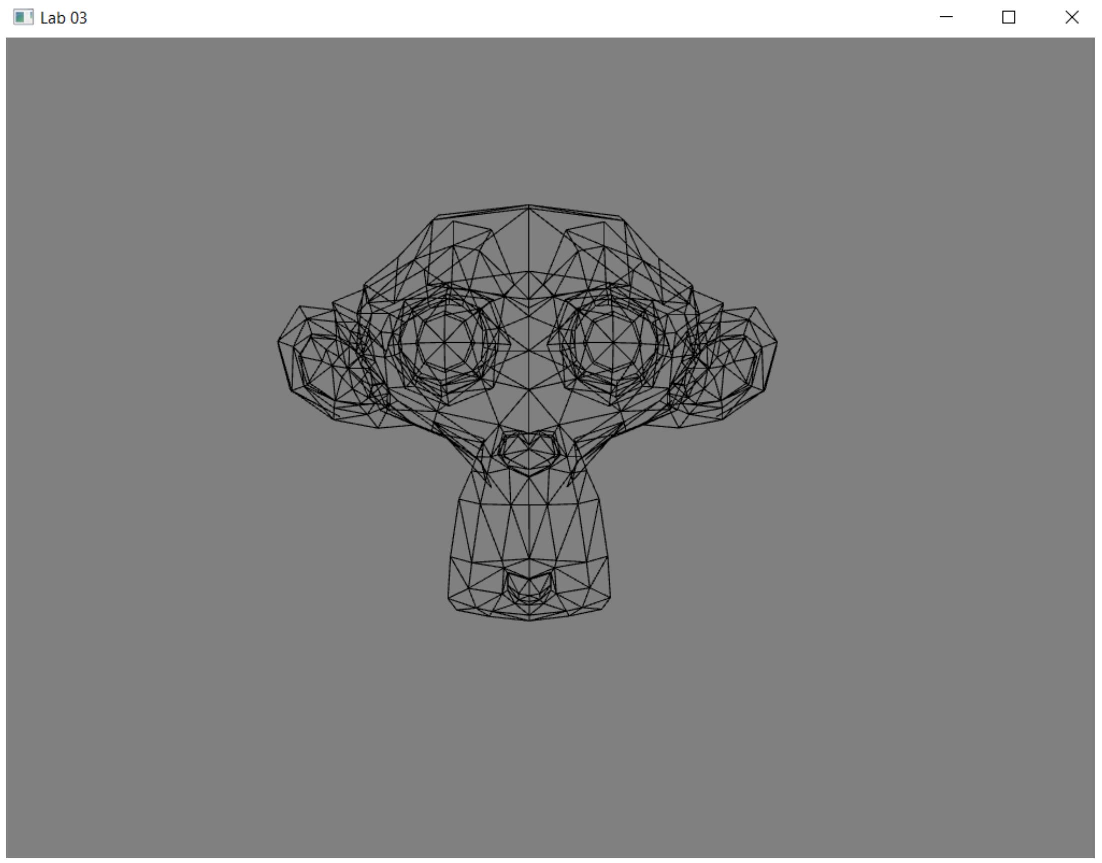
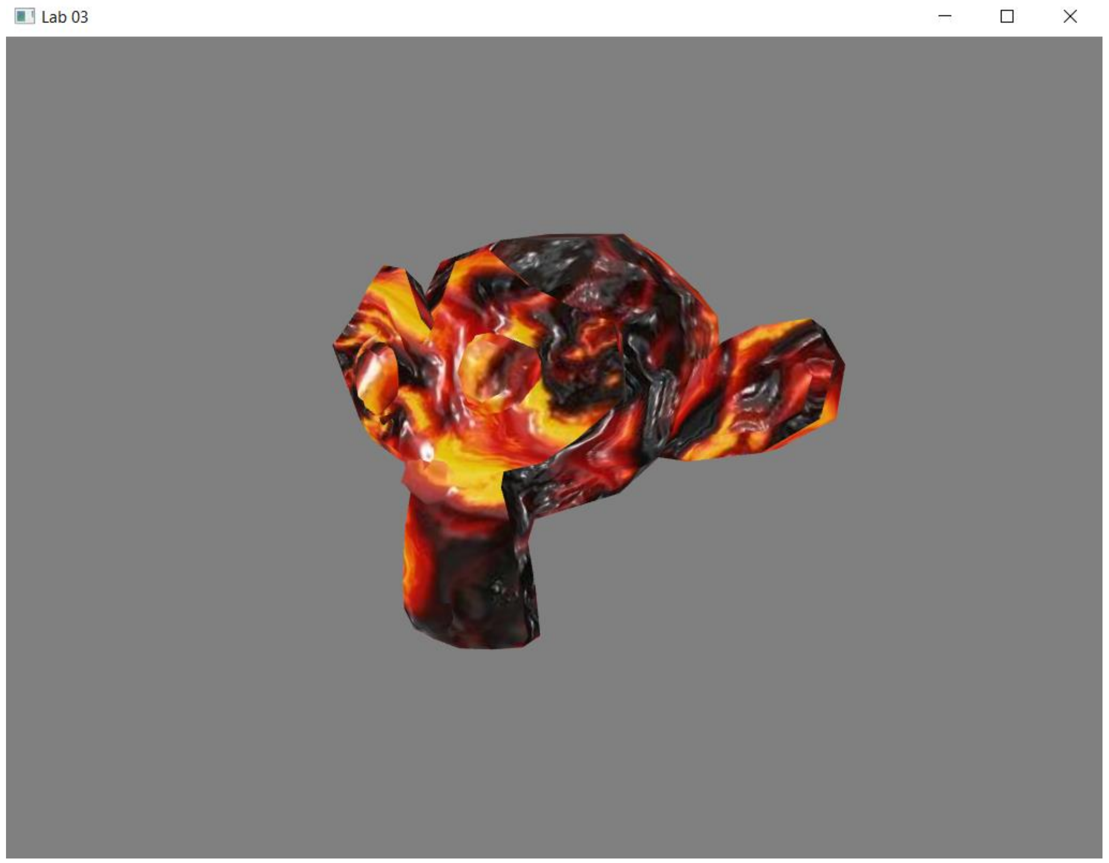




Lab 3: Camera & Texture Mapping

3D model file formats (obj, glb, fbx, …)

- In the previous lab, we defined the vertices of a cube manually.
- Not practical in real applications. Instead, we use files that save all the needed information to represent a 3D model.
- The models are designed in specialized software (Blender, Maya etc.) and the corresponded data is exported automatically.

Obj Format

```
# cube
v 1.000000 -1.000000 -1.000000
v 1.000000 -1.000000 1.000000
v -1.000000 -1.000000 1.000000
v -1.000000 -1.000000 -1.000000
v 1.000000 1.000000 -1.000000
v 1.000000 1.000000 1.000001
v -1.000000 1.000000 1.000000
v -1.000000 1.000000 -1.000000
```

```
vt 0.748573 0.750412
vt 0.749279 0.501284
vt 0.999110 0.501077
vt 0.999455 0.750380
vt 0.250471 0.500702
vt 0.249682 0.749677
vt 0.001085 0.750380
vt 0.001517 0.499994
vt 0.499422 0.500239
vt 0.500149 0.750166
vt 0.748355 0.998230
vt 0.500193 0.998728
vt 0.498993 0.250415
vt 0.748953 0.250920
```

```
vn 0.000000 0.000000 -1.000000
vn -1.000000 -0.000000 -0.000000
vn -0.000000 -0.000000 1.000000
vn -0.000001 0.000000 1.000000
vn 1.000000 -0.000000 0.000000
vn 1.000000 0.000000 0.000001
vn 0.000000 1.000000 -0.000000
vn -0.000000 -1.000000 0.000000
```

```
f 5/1/1 1/2/1 4/3/1
f 5/1/1 4/3/1 8/4/1
f 3/5/2 7/6/2 8/7/2
f 3/5/2 8/7/2 4/8/2
f 2/9/3 6/10/3 3/5/3
f 6/10/4 7/6/4 3/5/4
f 1/2/5 5/1/5 2/9/5
f 5/1/6 6/10/6 2/9/6
f 5/1/7 8/11/7 6/10/7
f 8/11/7 7/12/7 6/10/7
f 1/2/8 2/9/8 3/13/8
f 1/2/8 3/13/8 4/14/8
```

Q: τι είναι κάθε μία από τις παραμέτρους? v, vt, vn, f

Obj Format

```
# cube
v 1.000000 -1.000000 -1.000000
v 1.000000 -1.000000 1.000000
v -1.000000 -1.000000 1.000000
v -1.000000 -1.000000 -1.000000
v 1.000000 1.000000 -1.000000
v 1.000000 1.000000 1.000001
v -1.000000 1.000000 1.000000
v -1.000000 1.000000 -1.000000
```

```
vt 0.748573 0.750412
vt 0.749279 0.501284
vt 0.999110 0.501077
vt 0.999455 0.750380
vt 0.250471 0.500702
vt 0.249682 0.749677
vt 0.001085 0.750380
vt 0.001517 0.499994
vt 0.499422 0.500239
vt 0.500149 0.750166
vt 0.748355 0.998230
vt 0.500193 0.998728
vt 0.498993 0.250415
vt 0.748953 0.250920
```

```
vn 0.000000 0.000000 -1.000000
vn -1.000000 -0.000000 -0.000000
vn -0.000000 -0.000000 1.000000
vn -0.000001 0.000000 1.000000
vn 1.000000 -0.000000 0.000000
vn 1.000000 0.000000 0.000001
vn 0.000000 1.000000 -0.000000
vn -0.000000 -1.000000 0.000000
```

```
f 5/1/1 1/2/1 4/3/1
f 5/1/1 4/3/1 8/4/1
f 3/5/2 7/6/2 8/7/2
f 3/5/2 8/7/2 4/8/2
f 2/9/3 6/10/3 3/5/3
f 6/10/4 7/6/4 3/5/4
f 1/2/5 5/1/5 2/9/5
f 5/1/6 6/10/6 2/9/6
f 5/1/7 8/11/7 6/10/7
f 8/11/7 7/12/7 6/10/7
f 1/2/8 2/9/8 3/13/8
f 1/2/8 3/13/8 4/14/8
```

v: vertex

vt: texture coordinates (UVs)

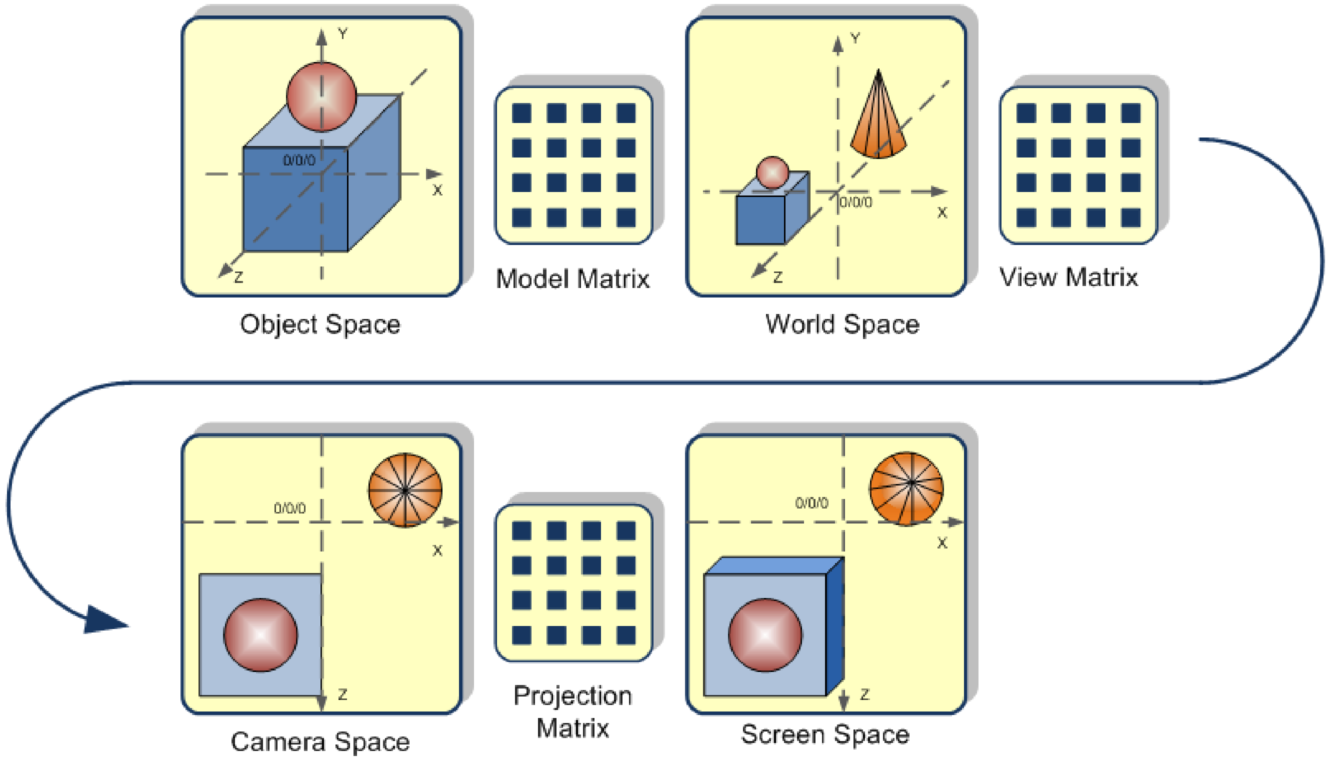
vn: normal vertices

f: faces (v1/vt1/vn1, v2/vt2/vn2, v3/vt3/vn3)

Camera Tasks

- Task 1: Move the camera with WASD
- Task 2: Perspective Projection
- Task 3: Orthographic Projection
- Task 4: Rotate the camera using the mouse

Task 2: Perspective Projection



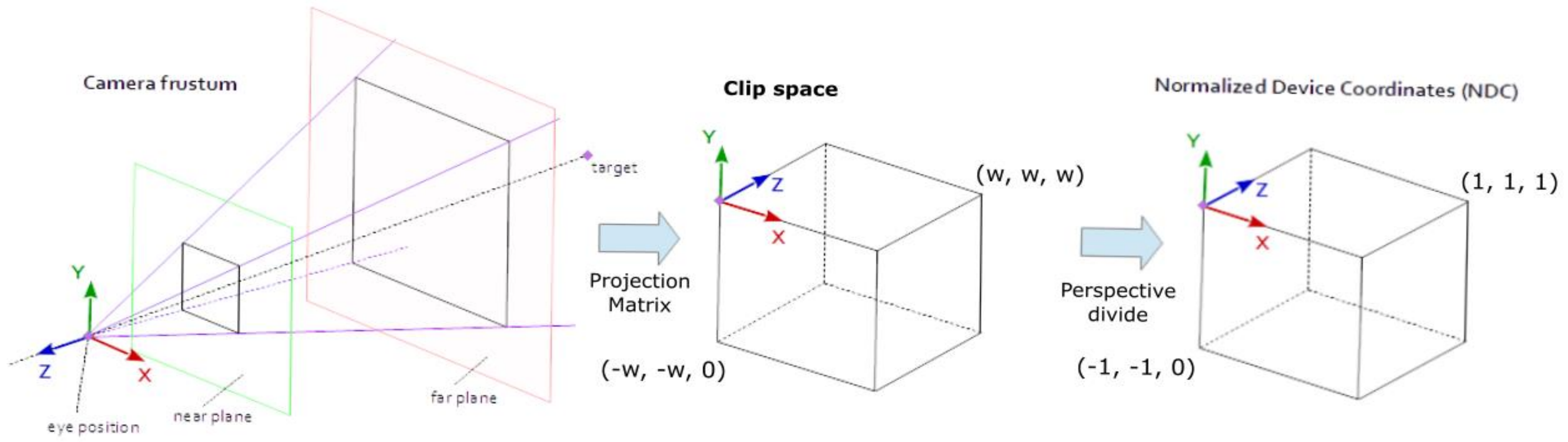
Source: https://www.c-jump.com/bcc/common/Talk3/Math/Matrices/W01_0190_coordinate_spaces_co.htm

$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Subtask 2.1: Translate Suzanne's Model Matrix 10 units away

(alongside z axis).

Task 2: Perspective Projection



$$P^x_{screen} = \frac{near \cdot P^x_{cam}}{-P^z_{cam}}$$

$$P^y_{screen} = \frac{near * P^y_{cam}}{-P^z_{cam}}$$

$$P^x_{ndc} = \frac{2 \cdot P^x_{\text{screen}}}{r\ - l} - \frac{r + l}{r - l}$$

$$P^y_{ndc} = \frac{2 \cdot P^y_{\text{screen}}}{t - b} - \frac{t + b}{t - b}$$

Task 3: Orthographic Projection

$$\begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

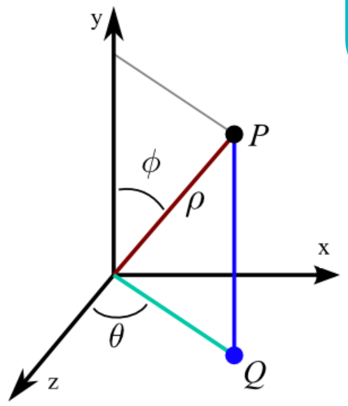
Subtask 3.1: Do the same as before. Notice anthing?

Task 4: Camera Rotation!

Hint:

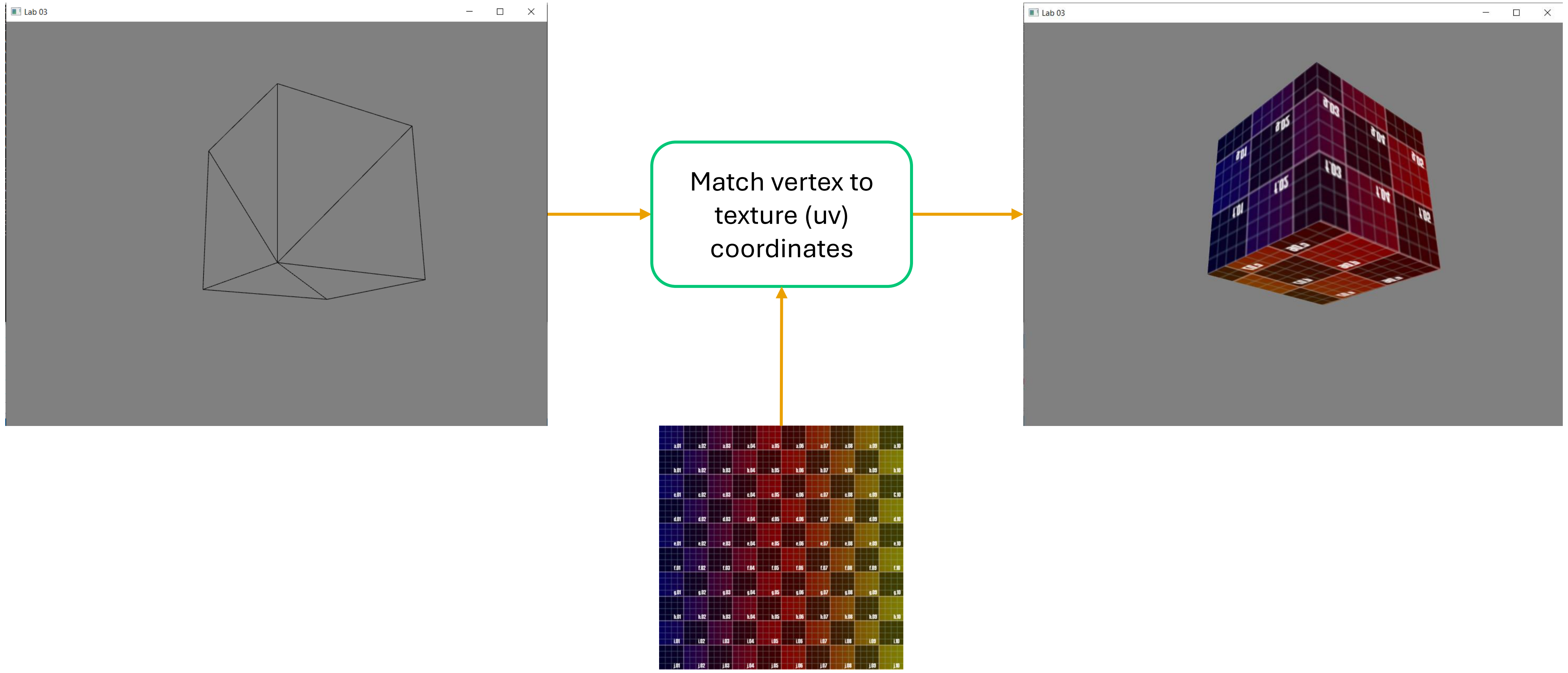
$$\begin{tikzcd}\text{Spherical} \arrow[r] & \text{Cartesian}\end{tikzcd}$$

$$x = \cos\varphi \sin\theta$$
$$y = \sin\varphi$$
$$z = -\cos\varphi \cos\theta$$

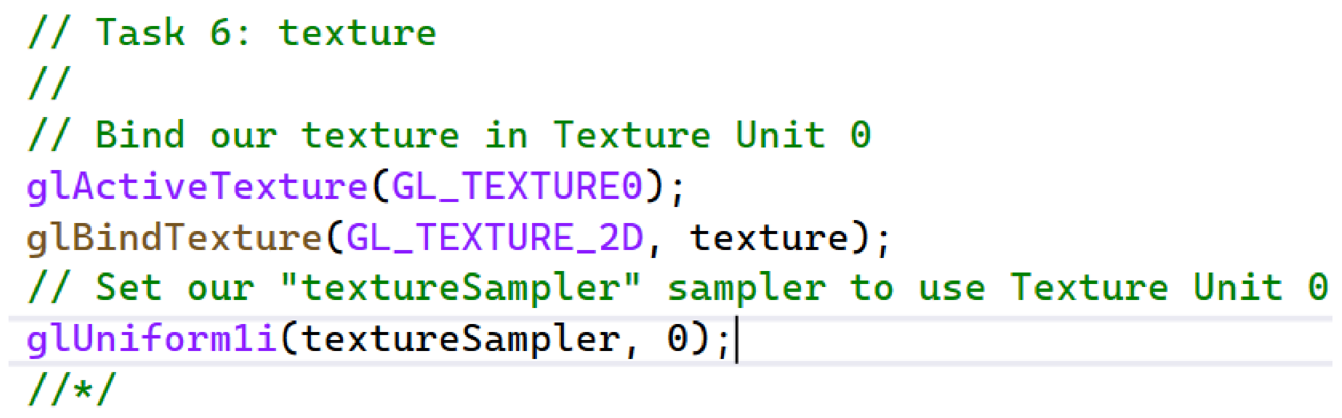


```
// Set up the projection matrix
// Perspective projection with field of view (FoV), aspect ratio, near and far clipping planes
projectionMatrix = perspective(radians(FoV), 4.0f / 3.0f, 0.1f, 100.0f);

// Alternatively, uncomment the line below for orthographic projection
//projectionMatrix = ortho(-5.0f, 5.0f, -5.0f, 5.0f, 0.0f, 15.0f);

// Set up the view matrix to orient and position the camera
// Looking from 'position' towards 'position - forward direction' with 'up' direction as (0,1,0)
viewMatrix = lookAt(position, position + direction , vec3(0.0f, 1.0f, 0.0f));
```

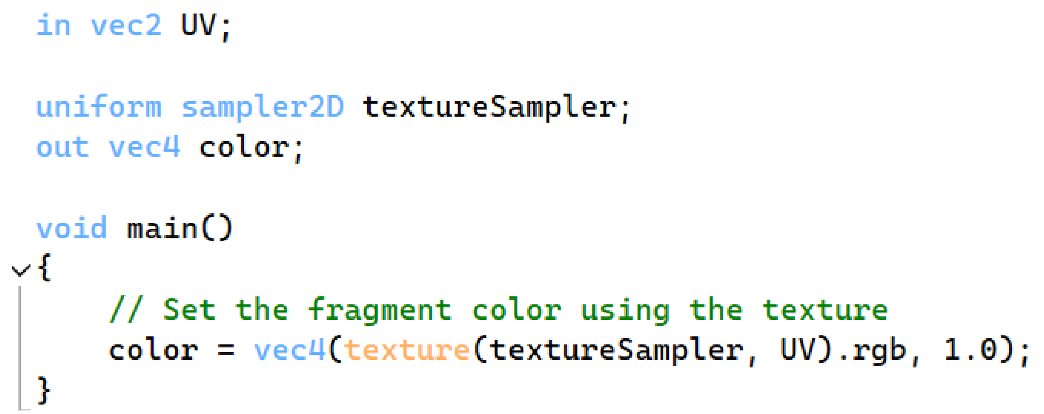
Texture Mapping!



Task 6: Assign a Texture!



```
// Task 6: texture
//
// Bind our texture in Texture Unit 0
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, texture);
// Set our "textureSampler" sampler to use Texture Unit 0
glUniform1i(textureSampler, 0);
//*/
```



```
in vec2 UV;

uniform sampler2D textureSampler;
out vec4 color;

void main()
{
    // Set the fragment color using the texture
    color = vec4(texture(textureSampler, UV).rgb, 1.0);
}
```

- ActivateTexture: Tell OpenGL that we want to use a texture unit.
- BindTexture: Bind to the currently active texture unit (in this case GL_TEXTURE0).
- Uniform1i: Set out “textureSampler” uniform variable inside the fragment shader to use whatever is bound to texture unit “0” (GL_TEXTURE0).

- The texture sampler in the fragment shader is a uniform variable that lets you read pixel data from a bound texture using UV coordinates.
- The texture function retrieves texels from a texture using the UV coordinates and the sampler

Subtask 6.1: Enable texture blending for transparency

Subtask 6.2: Cull triangles with normals not facing towards the camera

Task 7: moving texture & overlay

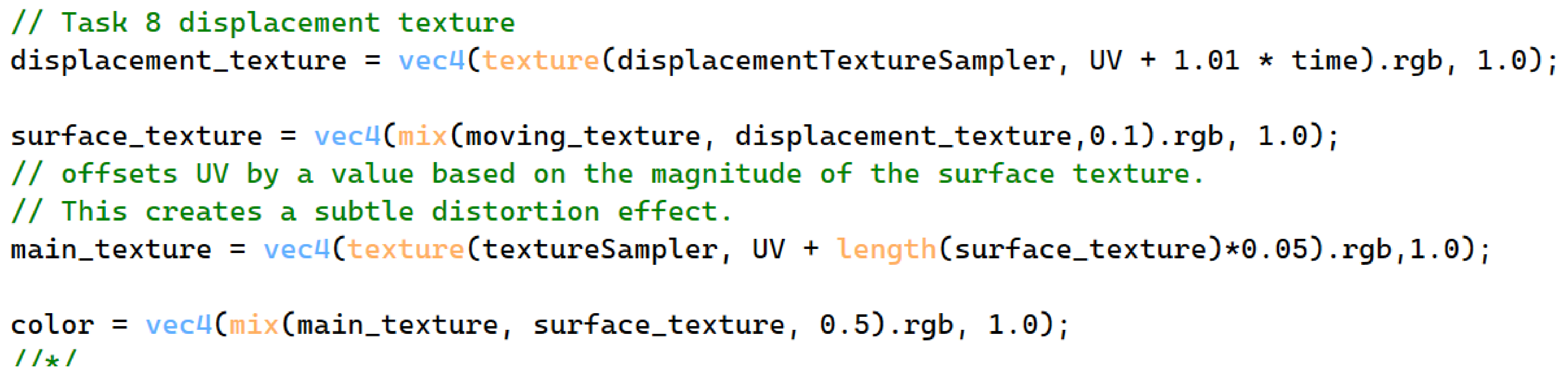
Q: How could I create a moving texture? (e.g. water)

Task 7: moving texture & overlay

Q: How could I create a moving texture? (e.g. water)

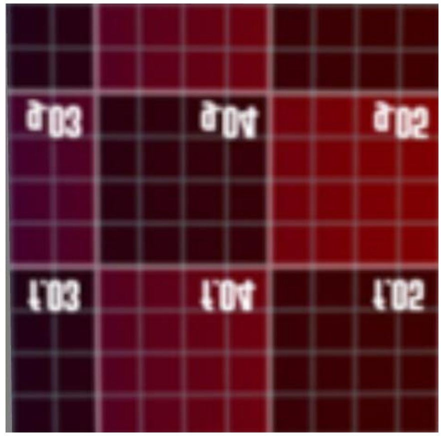
A: Move UV coordinates with respect to time [glfwGetTime( )]

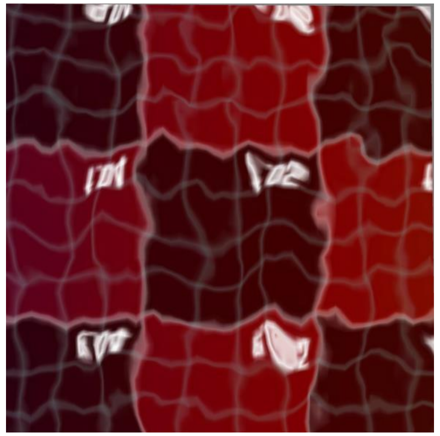
Task 8: displacement & distortion



```
// Task 8 displacement texture
displacement_texture = vec4(texture(displacementTextureSampler, UV + 1.01 * time).rgb, 1.0);

surface_texture = vec4(mix(moving_texture, displacement_texture,0.1).rgb, 1.0);
// offsets UV by a value based on the magnitude of the surface texture.
// This creates a subtle distortion effect.
main_texture = vec4(texture(textureSampler, UV + length(surface_texture)*0.05).rgb,1.0);

color = vec4(mix(main_texture, surface_texture, 0.5).rgb, 1.0);
//*/
```

Distortion effect





Homework!

Camera

- HW0: Sum up what we did today in the lab!
- HW1: Use arrow keys to zoom in/out with camera (FoV)
- HW2: Use Q and E to tilt left & right
- HW3: Use space and Ctrl for character moving up & down!

Textures

- HW4: Use a blending or/and distortion technique you found online! Compare!

Suggestions: DuDv maps, perlin noise, model normals, …

Πχ. DuDv maps

dU -> horizontal distortion
dV -> vertical distortion