3D Graphics Programming

T163 - Game Programming

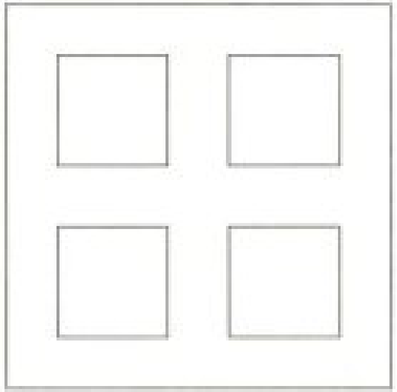
Fall 2020

Week 5

Orientation

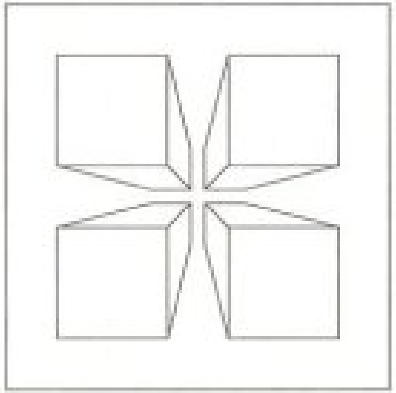
Camera Projections

Orthographic

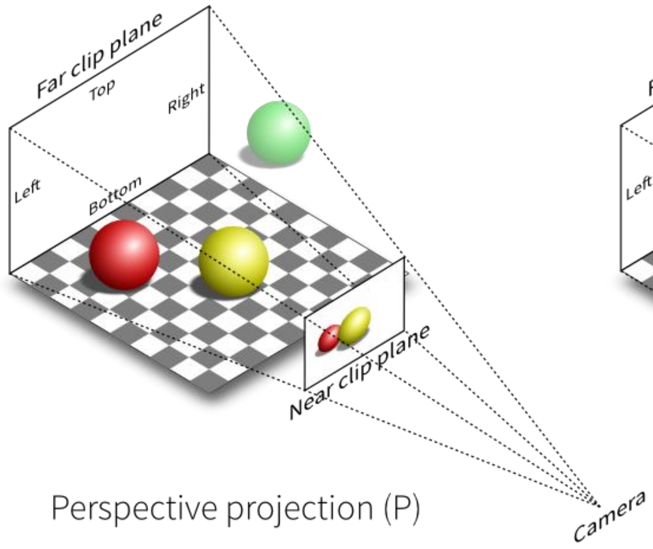


VS

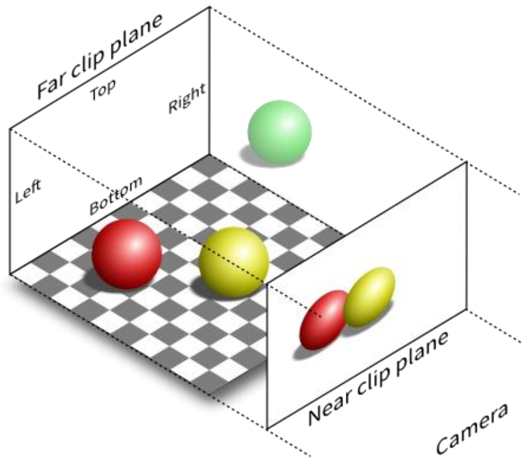
Perspective



Camera Projections



Perspective projection (P)



Orthographic projection (O)

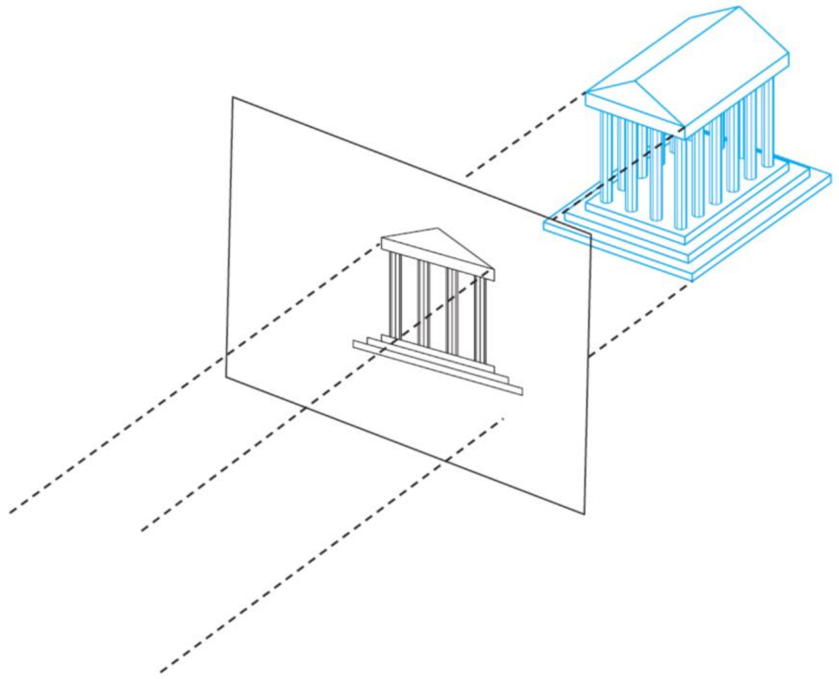
Camera Projections

- For the orientation component of our program, we setup multiple 4x4 matrices:

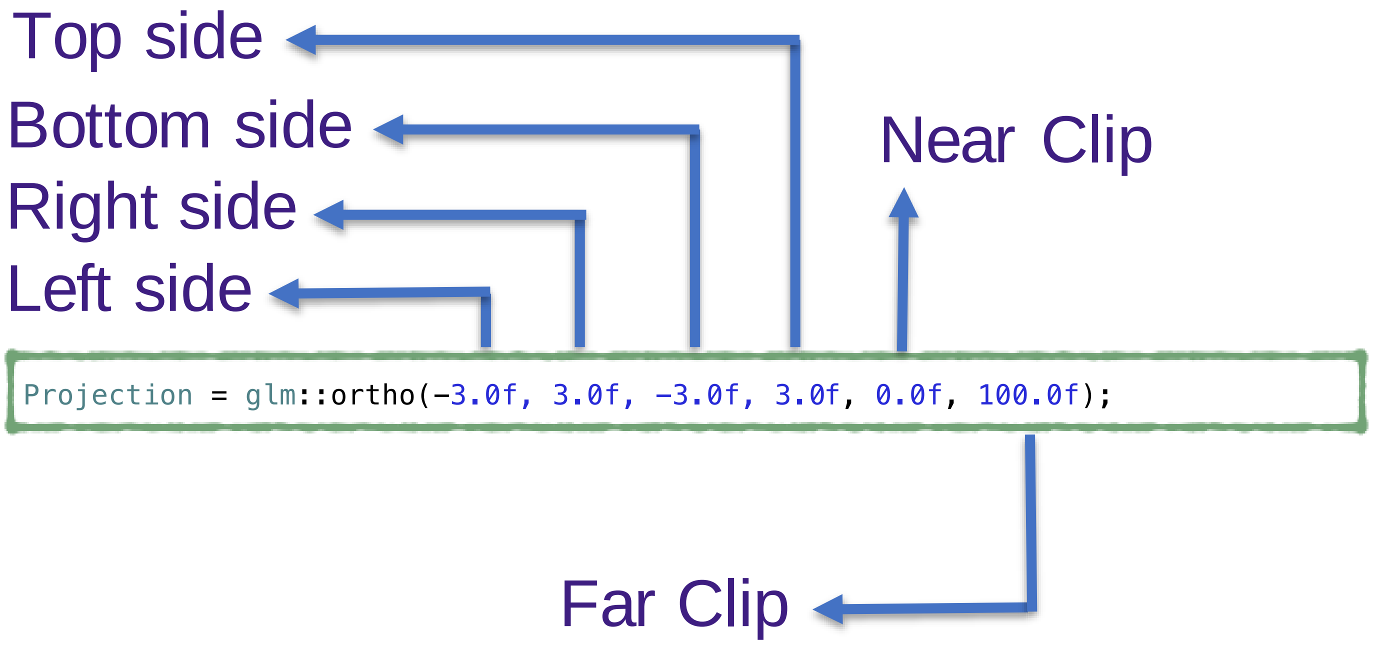
```
glm::mat4 MVP, View, Projection;
```

- Then in the init() function, we set our Projection and View
- The next slides will show the Projection matrix

Camera Projections - Orthographic

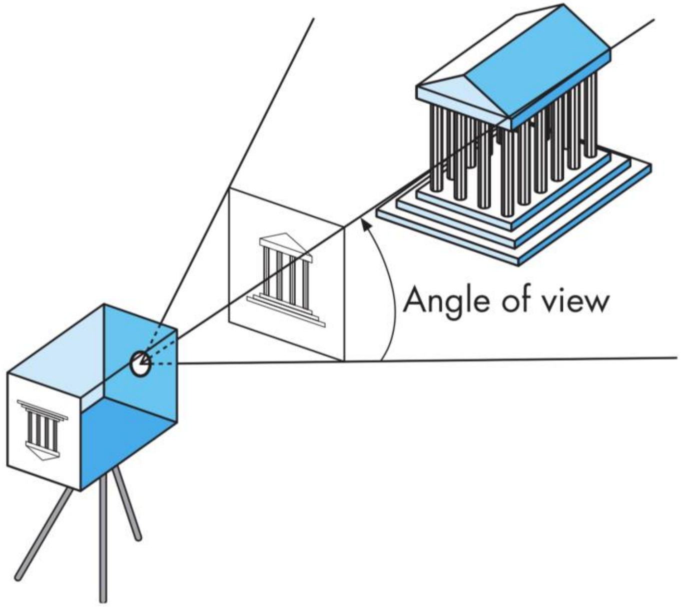


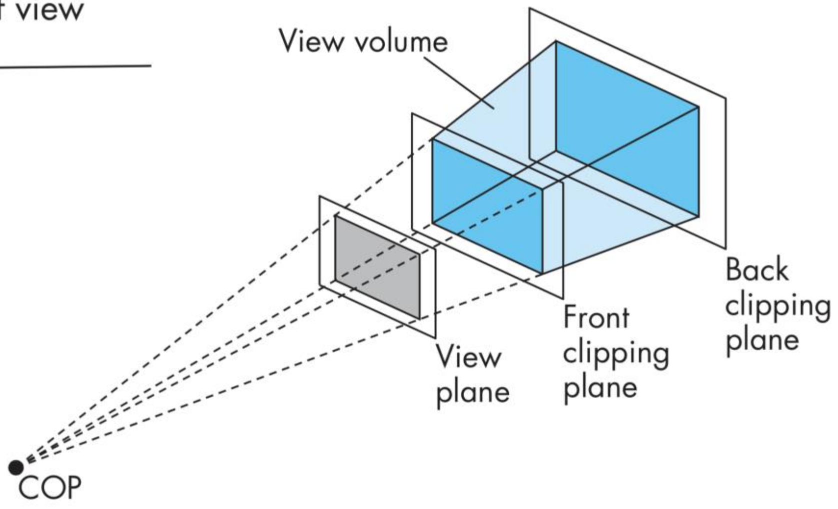
Camera Projections - Orthographic



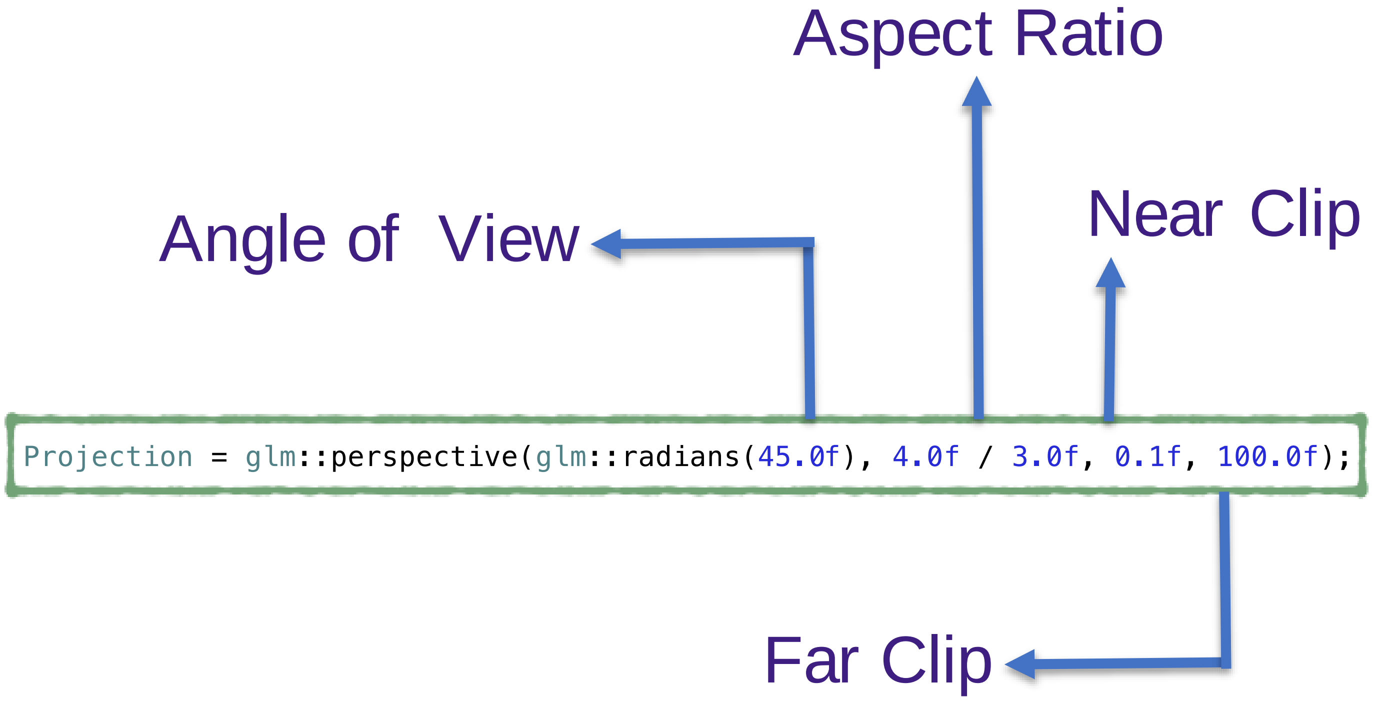
```
Projection = glm::ortho(-3.0f, 3.0f, -3.0f, 3.0f, 0.0f, 100.0f);
```

Camera Projections - Perspective





Camera Projections - Perspective



```
Projection = glm::perspective(glm::radians(45.0f), 4.0f / 3.0f, 0.1f, 100.0f);
```

Camera Projections

- The View matrix is set by invoking the glm::lookAt function that returns a mat4

```
View = glm::lookAt(
    glm::vec3(0, 0, 3.0f), // Origin. Camera is at (0,0,3), in World Space
    glm::vec3(0, 0, 0),   // Look target. Looks at the origin
    glm::vec3(0, 1, 0)   // Up vector. Head is up (set to 0,-1,0 to look upside-down)
);
```

- Later, the Model-View-Projection matrix (MVP) is created by multiplying the Model matrix with the Projection and View matrices, thus:

```
MVP = Projection * View * Model;
glUniformMatrix4fv(mvp_ID, 1, GL_FALSE, &MVP[0][0]);
```

- The glUniformMatrix4fv is an Extension Wrangler function that specifies the value of a uniform variable declared in the vertex shader
- A uniform variable holds a value that is the same for all vertices

Week 5

Lab Activities

Week 5 Lab

- For the lab, see Hooman's materials
- OpenGL examples covered:
  - More Projections
    - Orthographic and perspective
    - Square of walls
    - Sphere
    - Shadow

Week 5

End