Viewing and Modeling
Computer Viewing

Three aspects of viewing process:

• Position camera (model-view matrix)
• Selecting a lens (projection matrix)
• Clipping (view volume)
World and Camera Frames

- Base frame in OpenGL is world frame
- Use view matrix to change world representation to camera representation
- Fixed pipeline OpenGL treated model and view matrices as single (model-view) matrix
OpenGL Camera

Initial representation:

- Object and camera frames are the same (model-view matrix is identity)
- Camera located at origin
- Camera points in negative Z direction
- Default view volume is centered at origin with side lengths of 2 (NDC)
Changing the View

How to handle objects around Z axis?
Moving Camera Frame

Move the camera in the positive Z direction (translate camera frame)
Move objects in the negative Z direction (translate world frame)

…Which is better?
Moving Camera Frame

Move the camera in the positive Z direction (translate camera frame)
Move objects in the negative Z direction (translate world frame)

...they’re equivalent!
View Matrix

• All vertices defined relative to the camera
• Therefore world moves relative to camera

Consider:

```cpp
glm::mat4 ViewMatrix =
    glm::translate(0.f, 0.f, -14.f);
```

What is this doing?
Translation in View Space

\[
\begin{bmatrix}
1 & 0 & 0 & x \\
0 & 1 & 0 & y \\
0 & 0 & 1 & z \\
0 & 0 & 0 & 0
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & -14 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Point at (0, 0, 0) moves to (0, 0, -14)
General Camera Motion

Position camera using translations and rotations
- Move camera to origin (T)
- Rotate camera (R)
- MV = RT
A Better Viewing Matrix

“Look at” Transform:
Construct an affine 4x4 matrix to map world space into camera space

What do we need to know about the camera’s placement in the world to construct this?
glm::lookAt

Defines:
• Camera position
• Camera target
• Camera up

Returns:
• View matrix
lookAt Algorithm

In order to define view coordinate system:

• Z axis (forward vector) = normalize(at - eye)
• X axis (left vector) = normalize(up x Z)
• Y axis (up vector) = normalize(X x Z)

What happens if Z or up are zero length?
What happens if Z and up are coincident?
OpenGL’s Internal lookAt Matrix

\[
\begin{bmatrix}
X_x & X_y & X_z & 0 \\
Y_x & Y_y & Y_z & 0 \\
-Z_x & -Z_y & -Z_z & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Handles camera positioning without explicit translates and rotates
Why Recompute Up?

Given up direction not necessarily perpendicular to forward vector

Actual up vector will be orthogonal to left and forward vectors
Combining Model-View-Projection

\[
glm::mat4 \text{MVPmatrix} = \text{projection} \times \text{view} \times \text{model};
\]

Remember: matrix multiplication is associative but not commutative:

\[
A(BC) = (AB)C \quad ABC \neq CBA
\]
OpenGL Tutorial

Look through:

http://www.opengl-tutorial.org/beginners-tutorials/tutorial-3-matrices/