LOOKAT MATRIX

HJALMAR BASILE
DEC 2018

After the MODEL matrix transformed our scene objects from LOCAL to WORLD space, we need the VIEW matrix to transform the coordinates to CAMERA space.

We can see the transformation which describes the camera configuration as a composition of:

1) a rotation R around the origin

2) a translation T to the final camera position

So our transformation is $\Psi = T \circ R$, also the camera is just an abstraction to make life easier to the developers, what we actually do from engine side is transforming the whole scene using the inverse of Ψ.

(Example: moving the camera to the right of 1 unit is equivalent to moving the world of 1 unit to the left)

So our view matrix is $V = \Psi^{-1} = (T \circ R)^{-1} = R^{-1} \circ T^{-1}$.

Let's start by describing the translation first (it's easier)

CAMERA TRANSLATION

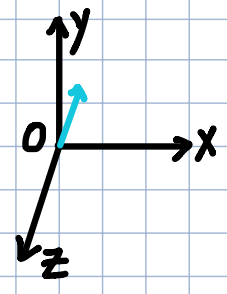
Suppose that our camera is moved to the point $\vec{P}=(P_x, P_y, P_z)$.

Then $T = \begin{pmatrix} 1 & & & P_x \\ & 1 & & P_y \\ & & 1 & P_z \\ & & & 1 \end{pmatrix}$, indeed $T \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x+P_x \\ y+P_y \\ z+P_z \\ 1 \end{pmatrix}$

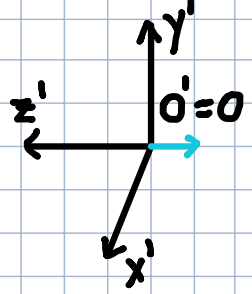
Also an easy observation is that $T^{-1} = \begin{pmatrix} 1 & & & -P_x \\ & 1 & & -P_y \\ & & 1 & -P_z \\ & & & 1 \end{pmatrix}$

CAMERA ROTATION

Let's start with the following example:



(ORIGINAL)



(ROTATED)

(→: where camera is looking at in OpenGL)

Standard basis

$B = \{e_1=\begin{pmatrix}1\\0\\0\end{pmatrix}, e_2=\begin{pmatrix}0\\1\\0\end{pmatrix}, e_3=\begin{pmatrix}0\\0\\1\end{pmatrix}\}$

New basis

$B' = \{v_1=\begin{pmatrix}0\\0\\1\end{pmatrix}, v_2=\begin{pmatrix}0\\1\\0\end{pmatrix}, v_3=\begin{pmatrix}-1\\0\\0\end{pmatrix}\}$

Observe that the matrix $\left(v_1 \middle| v_2 \middle| v_3\right) = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$ will map $\ell_1 \mapsto v_1$, $\ell_2 \mapsto v_2$, $\ell_3 \mapsto v_3$.

This is true for any v_1, v_2, v_3, indeed

$$\left(v_1 \middle| v_2 \middle| v_3\right)\begin{pmatrix}1\\0\\0\end{pmatrix} = v_1, \quad \left(v_1 \middle| v_2 \middle| v_3\right)\begin{pmatrix}0\\1\\0\end{pmatrix} = v_2, \quad \left(v_1 \middle| v_2 \middle| v_3\right)\begin{pmatrix}0\\0\\1\end{pmatrix} = v_3$$

Also if we choose v_1, v_2, v_3 such that they are orthonormal, then $\begin{pmatrix} v_1^T \\ \hline v_2^T \\ \hline v_3^T \end{pmatrix} \cdot \left(v_1 \middle| v_2 \middle| v_3\right) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = I \Rightarrow$

$\left(v_1 \middle| v_2 \middle| v_3\right)$ is invertible with inverse $\begin{pmatrix} v_1^T \\ \hline v_2^T \\ \hline v_3^T \end{pmatrix}$.

Summing up, the rotation matrix we are looking for is

$$R = \left(\begin{array}{ccc|c} & & & 0 \\ v_1 & v_2 & v_3 & \vdots \\ & & & 0 \\ \hline 0 & \cdots & 0 & 1 \end{array}\right)$$

and its inverse,

under the assumption that v_1, v_2, v_3 are orthonormal,

is $R^{-1} = \left(\begin{array}{c|c} \begin{array}{c} v_1^T \\ \hline v_2^T \\ \hline v_3^T \end{array} & \begin{array}{c} 0 \\ \vdots \\ 0 \end{array} \\ \hline 0 \quad \cdots \quad 0 & 1 \end{array}\right)$

VIEW MATRIX

As stated before, $V = R^{-1} \cdot T^{-1}$, so it is equal to

$$V = \left(\begin{array}{ccc|c} v_{1x} & v_{1y} & v_{1z} & \\ v_{2x} & v_{2y} & v_{2z} & 0 \\ v_{3x} & v_{3y} & v_{3z} & \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \cdot \left(\begin{array}{ccc|c} 1 & & & -P_x \\ & 1 & & -P_y \\ & & 1 & -P_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right) =$$

$$= \left(\begin{array}{ccc|c} v_{1x} & v_{1y} & v_{1z} & -\vec{v_1} \cdot \vec{P} \\ v_{2x} & v_{2y} & v_{2z} & -\vec{v_2} \cdot \vec{P} \\ v_{3x} & v_{3y} & v_{3z} & -\vec{v_3} \cdot \vec{P} \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$$

, where $\vec{v_i} \cdot \vec{P}$ is the dot product between the two vectors.

We are now ready to derive the LookAt matrix.

LOOKAT

The LookAt function takes as input:

1.) The position $\vec{E}$ (eye) of the camera (what we called $\vec{P}$ above)

2.) The target $\vec{C}$ (center) the camera must look at ($\vec{C} \neq \vec{E}$)

3.) An up $\vec{U}$ vector, usually (0, 1, 0), NOT aligned with $\vec{C}-\vec{E}$

It will compute an appropriate orthonormal basis for

the camera space and it will output the VIEW MATRIX described before. This is the complete algorithm:

```
98          template<typename T, qualifier Q>
99          GLM_FUNC_QUALIFIER mat<4, 4, T, Q> lookAtRH(vec<3, T, Q> const& eye, vec<3, T, Q> const& center, vec<3, T, Q> const& up)
100         {
101                 vec<3, T, Q> const f(normalize(center - eye));
102                 vec<3, T, Q> const s(normalize(cross(f, up)));
103                 vec<3, T, Q> const u(cross(s, f));
104
105                 mat<4, 4, T, Q> Result(1);
106                 Result[0][0] = s.x;
107                 Result[1][0] = s.y;
108                 Result[2][0] = s.z;
109                 Result[0][1] = u.x;
110                 Result[1][1] = u.y;
111                 Result[2][1] = u.z;
112                 Result[0][2] =-f.x;
113                 Result[1][2] =-f.y;
114                 Result[2][2] =-f.z;
115                 Result[3][0] =-dot(s, eye);
116                 Result[3][1] =-dot(u, eye);
117                 Result[3][2] = dot(f, eye);
118                 return Result;
119         }
120
```

Notice that our orthonormal basis v_1, v_2, v_3 corresponds to $s, u, -f$, this is why the result matrix is

$$\left(\begin{array}{ccc|c} s_x & s_y & s_z & -\vec{s}\cdot\vec{E} \\ u_x & u_y & u_z & -\vec{u}\cdot\vec{E} \\ -f_x & -f_y & -f_z & +\vec{f}\cdot\vec{E} \\ \hline 0 & & & 1 \end{array}\right)$$

N.B. The algorithm is NUMERICALLY UNSTABLE when the input up vector and $\vec{C}-\vec{E}$ are almost aligned, this must be taken into consideration when calling the LookAt function.