3D RENDERING FOR MY GRANDMA

From the model to the world

Let:

$$x_v^m \in \mathbb{R}^3$$

be the vertex v in the model coordinates, these may even be hard-coded for simple geometry like a cube or a thetrahedron. Then we may want to:

1. enlarge the model, so multiply all the vertices by the same scalar;
2. rotate the model with its own attitude;
   This is all obtained using matrices. The first step can be made introducing a shear matrix of the kind:

$$Z_0 = \mathrm{diag}(z_0, z_1, z_2)$$

it is diagonal because it only shears the object. If we want to simply change the scale of it, the matrix can be substituted by a scalar:

$$Z_0 = \mathcal{I}_3 z_0 \quad \text{with } z_0 \in \mathbb{R}^3$$

Now before of moving the vertex we may want to rotate it. So we introduce rotation matrices and the vector of *Euler angles*:

$$\theta = (\theta_x, \theta_y, \theta_z)^T$$

these angles will be fed to our rotation matrices, for instance:

$$R_x(\theta_x) \in \mathbb{R}^{3\times 3}$$

the resulting rotation matrix will be:

$$M^m(\theta) = R_z(\theta_z) R_y(\theta_y) R_x(\theta_x)$$

Obviously after that we would like to move the object around our world, and this is obtained simply adding a translation vector:

$$x_{cm}^w \in \mathbb{R}^3$$

where by cm I mean *center of mass*. This is because an object, as we'll see, is a collection of vertices so, in the hypothesis of rigidity it makes sense to employ only one point to locate the object in the world.
In the modeling above the vertex **v** in the world coordinates will be written as:

$$x_v^w = M^m(\theta) Z_0 x_v^m + x_{cm}^w$$

So notice that in order to render a vertex so far we need:

$$3 \text{ attitude angles } \theta$$

$$\text{a shear matrix } Z_0$$

$$\text{the position of the center of mass } x^w_{cm}$$

Regarding the objects

I find it convenient to think about an object as a collection of vertices, that is a discrete (in this case read finite) subset of the world, or in math terms:

$$\mathcal{O} \subset \mathbb{R}^3$$

and more specifically:

$$\mathcal{O} := \{x_{v_i} \in \mathbb{R}^3 \text{ with } i = 1, \dots, N, N < \infty\}$$

It can be observed that the actual value of the *xs* of an object depends on the frame of reference, for instance we can specialize the object in the world frame of reference referring all the vector to that particular base:

$$\mathcal{O} = \{x^w_{v_i} \in \mathbb{R}^3 \text{ with } i = 1, \dots, N, N < \infty\}$$

...so an object is more general that all of its specializations on the different basis.

The camera

Now, what we see of our world depends on where the cam is and how it is oriented. So the first step is to try to understand how a point in the world frame transforms in the cam frame.
The position of the cam will be:

$$x_{cam}$$

and in the world frame:

$$x^w_{cam}$$

Now let us define the relative position of a vertex with respect to the cam (yet to be rotated) the vector:

$$r^w_v := x^w_v - x^w_{cam}$$

this is almost the needed vector, the last step is to rotate the camera...or better we want to traslate from the world language to the cam language the relative position vector. This is done again with a rotation matrix:

$$M(\theta_{cam}) = R_z(\theta_{z,cam})R_y(\theta_{y,cam})R_x(\theta_{x,cam})$$

It may be notice that this matrix translates from the cam *language* to the world language, that is the opposite transformation that we need, so we invert this map (we are sure that it is invertibile for almost all the Euler triplets). In math terms:

$$\text{Translator(W to C)} = M^{-1}(\theta_{cam}) = R^T_x(\theta_{x,cam})R^T_y(\theta_{y,cam})R^T_z(\theta_{z,cam})$$

exploiting the orthogonality of the rotation matrices.
Putting everything toghether the vertex in the cam frame is:

$$x^c_v = M^T(\theta_{cam})(x^w_v - x^w_{cam})$$

Then in order to write this transformation we need:

the angles of the camera θ_{cam}

the vertices in the world frame x_v^w

the position of the cam in the world frame x_{cam}^w