Tutorial 5 – Matrices and Transformations

Basics

In the following, we want to look into the steps of the graphics pipeline dealing with perspective projection, i.e. the matrix multiplications that transform our 3D models and vectors into the 2D representations that are shown on the computer screen. Chapter 7 of the book and the slides from the lecture describe the individual steps involved in this process. It will be handy if you have those around while doing the following exercises. Let's start by taking a closer look into some of the individual steps involved.

Exercise 1.

Our 3D models and scenes are usually expressed in world coordinates: that is, with respect to a general coordinate system (usually a Cartesian coordinate system). In these world coordinates, the location and orientation of our camera can be described by the *eye* position E specifying the camera's location and the view vector $\vec{V}$ specifying the direction in which our camera is looking. To project the 3D scene towards our camera, it would be much easier if the origin was placed at the position of the eye and the z-axis was pointing in the viewing direction (or negative viewing direction). We can do that by moving from world coordinates to camera coordinates where our 3D models and vectors are expressed with respect to a coordinate system centered at the eye vector, and the axes are aligned with the viewing window and view vector.

Let's look at a simple example in 2D to understand the difference between "expressed in world coordinates" versus "expressed in camera coordinates". Assume a 2D world coordinate system with base vectors $\vec{b}_1 = (1, 0)$ and $\vec{b}_2 = (0, 1)$, and a camera placed at position $E = (2, 1)$, looking in direction $\vec{V}=(\frac{1}{2}\sqrt{2}, \frac{1}{2}\sqrt{2})$. Notice that the view vector is already a unit vector, so $\vec{u} = \vec{V}$ and $\vec{v} = (-\frac{1}{2}\sqrt{2}, \frac{1}{2}\sqrt{2})$ gives us a camera coordinate system.

a) Draw an image of this scene including a point $P = (3, 1)$ expressed in world coordinates.
b) If a point P is expressed with respect to a particular coordinate system, e.g. the world coordinates given by $\vec{b}_1, \vec{b}_2$, we often denote this by P_{xy}. Likewise, if this point is expressed in camera coordinates, we denote it P_{uv}.
   Write down the general form of a point P_{xy} in world coordinates (i.e., write down P_{xy} as a linear combination of the base vectors $\vec{b}_1$ and $\vec{b}_2$).
   Write down the general form of a point P_{uv} in camera coordinates (i.e., write down P_{uv} as a linear combination of the base vectors $\vec{u}$ and $\vec{v}$).
   Note: you don't have to fill in and calculate the actual numbers yet.
c) We can transform between these two coordinate systems using matrix multiplication. The procedure is the same as in the case of rotation around an arbitrary vector that we discussed in the lecture about transformations.

Give the matrix that transforms a point given in camera coordinates into one given in world coordinates, i.e. a matrix M with $P_{xy} = M\ P_{uv}$.
Give the matrix that transforms a point given in world coordinates into one given in camera coordinates, i.e. a matrix M with $P_{uv} = M\ P_{xy}$.

d) Fill in the actual numbers and express the point $P_{xy} = (3, 1)$ in camera coordinates.

Exercise 2.

In the simple 2D example in the previous exercise it was easy to create the base vectors for the camera coordinate system. For 3D, this process is a little more complicated because we need three base vectors, but we only have one (the view vector) to do this. Fortunately, we already learned how to generate an orthonormal basis given a single vector when we talked about transformation matrices for rotation around an arbitrary vector in 3D.

a) For the rotation around an arbitrary vector in 3D, we used a non-parallel random vector to create our coordinate system using the cross product. Here, we introduced a so-called view up vector $\overrightarrow{up}$. How is that vector specified, and why do we need it (i.e. why can't we just take a random vector like before)?
b) One of the axis of our camera coordinate system will be a normalized version of either the view vector or the negative view vector. It depends of course if we want to get a left or right handed one. How can we, for example, create a right handed one using the view up vector and the negative view vector?

Exercise 3.

The following matrix M_{orth} maps the orthographic view volume to the canonical view volume:

$$M_{orth} = \begin{pmatrix} \frac{2}{r-l} & 0 & 0 & \frac{l+r}{l-r} \\ 0 & \frac{2}{t-b} & 0 & \frac{b+t}{b-t} \\ 0 & 0 & \frac{2}{n-f} & \frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

a) How is the orthographic view volume defined?
b) How is the canonical view volume defined?
c) Show that the above matrix takes the corners of the orthographic view volume to the corners of the canonical view volume.

Exercise 4.

Similarly to when we introduced homogeneous coordinates in order to be able to do affine transformations, we had to further extend our matrix framework to enable us to do projective transformations.

a) Why?
b) For a 4×4 matrix whose top three rows are arbitrary and whose bottom row is $(0,0,0,1)$, show that points $(x, y, z, 1)$ and (hx, hy, hz, h) transform to the same point after homogenization (for $h \neq 0$).
c) Given our extended framework, the original z-coordinate is mapped to a value $z_s = n + f - \frac{fn}{z}$. Show algebraically that the perspective matrix preserves the order of z values within the view volume.

Now that we have a better understanding of some of the steps involved, let's go through the whole process of projecting a 3D model onto our 2D screen with a concrete example. Notice that the following requires some calculations that are not as "smooth" and easy as the ones we usually have. If you really want to calculate all these matrices (and although we will give you "nicer" ones in the exam it is recommended to do so), you can of course use a calculator here (but not in the exam where, as said, we try to make the numbers easier to calculate).

Exercise 5.

Let's assume that our model has an object centered at the point $(7, 16, 18)$. Now, instead of looking at it from the origin, we want to look at it from behind and above, so we place the origin of our camera at $(10, 20, 30)$. What is the view vector $\vec{V}$ we should specify so that the object is centered in the image?

Exercise 6.

For our camera in the previous problem, we specify an up vector $\overrightarrow{up}$ of $(0, 1, 0)$. We are going to do projection in the way it is explained in Chapter 7 of the textbook and the related lecture. Explain how we compute the matrix M_{cam} that does the transformation from world space to camera space. If you want to compute the actual numbers, beware that they are not really nice (they contain fractions and square roots), so calculating may take a while. But as said, it is instructional to do such calculations at least once.

Exercise 7.

Our camera hasn't been completely specified yet. Amongst other things, we need to set the near and far plane distances. Let's set them at $n = -1$ and $f = -100$, respectively (recall that we view along the negative z-axis). What is the matrix P that does the perspective transform, given these values? (Note that in this case, the numbers are much nicer.)

Exercise 8.

The final matrix that we need to construct is M_{orth}, the one that takes care of the orthographic projection. Let's specify an image where width times height is 1024×768, and the left, right, top and bottom plane parameters are $-4, 4, 4$, and -3, respectively. Determine M_{orth}, given these parameters.

Exercise 9.

If you dare, compute the full matrix $M = M_{\text{orth}}\ P\ M_{\text{cam}}$ (or, if you use the notation from the 2nd edition of the book: $M = M_{\text{o}}M_{\text{p}}M_{\text{v}}$). Onto which pixel is the point $(7, 16, 18)$ projected? Recall that we wanted it to be centered in the image.

The End

(for now)