Matrices play a large role in the field of computer graphics. The transformation pipelines in several computer graphics systems utilize what are known as "affine transformations", hence their name. In a nutshell, they involve the product of a vector and a matrix or two matrices.

An important detail of matrices that can be confusing is the notation or organization used to represent them and their individual elements. These two notations are "Row-Major Order" and "Column-Major Order".

Different programming languages and APIs may require you to use one order or the other, so having full command of matrix notation is essential. For example, C/C++ uses a Row-Major Order to represent multi-dimensional arrays (matrices), while OpenGL uses a Column-Major Order representation. In fact, no matter what graphics software you're using, traversing the transformation pipeline to get to different coordinate spaces via matrices is useful in several situations e.g. writing shaders.

This can be thought of as: Row-Major Order is similar to how we normally read, starting at the top-left corner, first reading from left to right in the same row, then proceeding to each new row downwards. Column-Major is read starting at the top-left, except each column is read from the top down, from left to right. With Row-Major Order, you are "reading" the rows. With Column-Major Order, you are reading the columns.

Column-Major Order Example:

$$A = \begin{bmatrix} a_{00} & a_{10} & a_{20} \\ a_{01} & a_{11} & a_{21} \\ a_{02} & a_{12} & a_{22} \end{bmatrix} B = \begin{bmatrix} b_{00} & b_{10} & b_{20} \\ b_{01} & b_{11} & b_{21} \\ b_{02} & b_{12} & b_{22} \end{bmatrix}$$

Row-Major Order Example:

$$A = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} B = \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix}$$

The first numbers in each elements' subscripts represent which row or column you are referencing, while the second numbers represent which element in that row or column you are referencing. One way I read this aloud to myself is "Element 0 of row 2", "Element 1 of column 0", "Row 2, element 1", "Column 0, element 2", etc.

Personally, I find that acquiring the product of two matrices can be extremely laborious, so I wanted to write software to help me make sure I understand this well and to make the task easier for myself.

Here is the technique I used:

$$X = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}; Y = \begin{bmatrix} j & m & p \\ k & n & q \\ l & o & r \end{bmatrix}$$
$$Z = XY$$

$$Z = \begin{bmatrix} aj + dk + gl & am + dn + go & ap + dq + gr \\ bj + ek + hl & bm + en + ho & bp + eq + hr \\ cj + fk + il & cm + fn + io & cp + fq + ir \end{bmatrix}$$

In the above example, Z is the product of X and Y, and all matrices are in Column-Major Order. If you were to follow the Row-Major Order, this operation would instead look like:

$$X = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}; Y = \begin{bmatrix} j & k & l \\ m & n & o \\ p & q & r \end{bmatrix}$$

$$Z = XY$$

$$Z = \begin{bmatrix} aj + bm + cp & dj + em + fp & gj + hm + ip \\ ak + bn + cq & dk + en + fq & gk + hn + iq \\ al + bo + cr & dl + eo + fr & gl + ho + ir \end{bmatrix}$$

The C++ code I've come up with to multiply 2 matrices in Column-Major Order uses triple-nested loops. It prints the loop indices for debugging and clarity purposes; it wasn't really until I did so that I could see how everything worked.