TORQUE MATH #3: MATRICES

Exercise Files
Starter – "engine/exercises/chapter10/exer_009.cc"
Answers – "engine/answers/chapter10/exer_009.cc"

Exercise Mission
n/a

Special Steps
Please remember, when you modify the engine and compile, you must copy the new executable over to your Kit/ directory before you can run it and see the changes in the Kit (as instructed below).

Synopsis

In this exercise, we will explore some features of the MatrixF class, learning how to create our own by hand, and examining the effects of transforms applied in various orders.

Prerequisites
1. *ch1_001.pdf "Using The Kit"*

Exercises

1 Constructing Matrices

Goal: Create our own matrices by hand, having them match the results of built-in translation (position), scaling, and rotation methods.

Starter Code: You are provided with two console function definitions, ch10_exer_009a(), and ch10_exer_009b().

ch10_exer_009a() uses the built in functions to generate a translation (position), a rotation, and a scaling matrix.

```
ConsoleFunction(ch10_exer_009a, void, 1, 1, "")
{
   MatrixF tMatrix; // Translation Matrix
   MatrixF rMatrix; // Rotation Matrix
   MatrixF sMatrix; // Scaling Matrix

   tMatrix.identity();
   rMatrix.identity();
   sMatrix.identity();

   Point3F tVec( 3.0f , 4.0f, 5.0f );
   Point3F rVec( mDegToRad( 70.0f ), 0.0f, mDegToRad( 32.1f ) );
   Point3F sVec( 3.0f , 4.0f, 5.0f );

   tMatrix.setPosition( tVec ); // Set translation matrix
   rMatrix.set( rVec );         // Set rotation matrix
   sMatrix.scale( sVec );       // Set scaling matrix

   dumpMatrix( "   Translation Matrix", tMatrix );
   dumpMatrix( "      Rotation Matrix", rMatrix );
   dumpMatrix( "       Scaling Matrix", sMatrix );
}
```

The second function (ch10_exer_009b()) is nearly empty. It is your job to fill this function body in with hand-written code that will effectively reproduce the code from ch10_exer_009b(), but without using the built in translation, rotation, and scaling matrix initialization methods.

TORQUE MATH #3: MATRICIES

```
ConsoleFunction(ch10_exer_009b, void, 1, 1, "")
{
   MatrixF tMatrix; // Translation Matrix
   MatrixF rMatrix; // Rotation Matrix
   MatrixF sMatrix; // Scaling Matrix

   tMatrix.identity();
   rMatrix.identity();
   sMatrix.identity();

   Point3F tVec( 3.0f , 4.0f, 5.0f );              // Translation vector
   Point3F rVec( 0.0f, 0.0f, mDegToRad( 32.1f ) ); // Rotation vector
   Point3F sVec( 3.0f , 4.0f, 5.0f );              // Scaling vector

   // 1 Set up translation matrix manually
   // Add code here (several lines)

   // 2 Set up rotation matrix manually
   // Add code here (several lines)


   // 3 Set up scaling matrix manually
   // Add code here (several lines)

   dumpMatrix( "Translation Matrix", tMatrix );
   dumpMatrix( "   Rotation Matrix", rMatrix );
   dumpMatrix( "    Scaling Matrix", sMatrix );
}
```

Steps:

1. Write code to manually set up the translation matrix using the same values as those used in ch10_exer_009a().
2. Write code to manually set up the rotation matrix using the same values as those used in ch10_exer_009a().
3. Write code to manually set up the scaling matrix using the same values as those used in ch10_exer_009a().

TORQUE MATH #3: MATRICIES

Output Goal:

When you successfully complete this exercise and rebuild Torque, you will we able to start the kit and to open the console(~). Then, when you run either ch10_exer_009a() or ch10_exer_009b(), you will get the following output.

```
==>ch10_exer_009a();
   Translation Matrix
1.00 0.00 0.00 3.00
0.00 1.00 0.00 4.00
0.00 0.00 1.00 5.00
0.00 0.00 0.00 1.00

      Rotation Matrix
0.85 0.53 0.00 0.00
-0.18 0.29 0.94 0.00
0.50 -0.80 0.34 0.00
0.00 0.00 0.00 1.00

       Scaling Matrix
3.00 0.00 0.00 0.00
0.00 4.00 0.00 0.00
0.00 0.00 5.00 0.00
0.00 0.00 0.00 1.00
```

Hints:

1. You can access matrix elements using either of the two preferred indexing methods listed in the "Array Accesses and idx()" section of chapter 10.
2. You'll want to refer to table 10.7 in the book for rotations.
3. You may want to create temporary matrices for some of this work.

2 Multiplication Order

Goal: Examine the effects of applying matrix transforms in various orders.

Starter Code: You are provided with one console function definition, ch10_exer_009c().

This console function will set up a set of matrices (transform, rotation, and translation).

```
ConsoleFunction(ch10_exer_009c, void, 1, 1, "")
{
   MatrixF tMatrix; // Translation Matrix
   MatrixF rMatrix; // Rotation Matrix
   MatrixF sMatrix; // Scaling Matrix

   MatrixF workMatrix;

   tMatrix.identity();
   rMatrix.identity();
   sMatrix.identity();

   Point3F tVec( 3.0f , 4.0f, 5.0f );               // Translation vector
   Point3F rVec( 0.0f , 0.0f, mDegToRad( 45.0f ) ); // Rotation vector
   Point3F sVec( 3.0f , 4.0f, 5.0f );               // Scaling vector

   tMatrix.setPosition( tVec ); // Set translation matrix
   rMatrix.set( rVec );         // Set rotation matrix
   sMatrix.scale( sVec );       // Set scaling matrix
```

Then, it will apply these transforms in four different orders to a matrix initialized to the identity matrix.

```
   // A
   workMatrix.identity();
   workMatrix.mul( rMatrix );
   workMatrix.mul( sMatrix );
   workMatrix.mul( tMatrix );
   dumpMatrix( "A - rot -> scal -> trans ", workMatrix );

   // B
   workMatrix.identity();
   workMatrix.mul( tMatrix );
   workMatrix.mul( sMatrix );
   workMatrix.mul( rMatrix );
   dumpMatrix( "B - trans -> scal -> rot ", workMatrix );
```

```
    // C
    workMatrix.identity();
    workMatrix.mul( sMatrix );
    workMatrix.mul( rMatrix );
    workMatrix.mul( tMatrix );
    dumpMatrix( "C - scal -> rot -> trans ", workMatrix );

    // D
    workMatrix.identity();
    workMatrix.mul( sMatrix );
    workMatrix.mul( tMatrix );
    workMatrix.mul( rMatrix );
    dumpMatrix( "D - scal -> trans -> rot ", workMatrix );

}
```

The four different orders are,

A – rotation, scaling, translation

B – translation, scaling, rotation

C – scaling, rotation, translation

D – scaling, translation, rotation

Questions:

1. Do any of these orders result in the same output?
   - If so, which ones?
   - If not, why not?
2. Without looking at the output, what is the translation result (as a vector) for series A? Series D?
4. Which translation is going to be larger?
5. Are these affine matrices? (Not sure? Then write the code to test them.)