TORQUE MATH #3 MATRICES

1 Constructing Matrices

Finished Code: (Your code may not look exactly like this, but should be fairly similar.)

```
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

   // Set up translation matrix manually
   tMatrix[MatrixF::idx( 3 , 0 )] = tVec.x;
   tMatrix[MatrixF::idx( 3 , 1 )] = tVec.y;
   tMatrix[MatrixF::idx( 3 , 2 )] = tVec.z;

   // Set up rotation matrix manually
   // Temporary X-Axis rotation matrix
   MatrixF xMatrix;
   xMatrix.identity();
   xMatrix[MatrixF::idx( 1 , 1 )] = mCos( mDegToRad( 70.0f ) );
   xMatrix[MatrixF::idx( 1 , 2 )] = -mSin( mDegToRad( 70.0f ) );
   xMatrix[MatrixF::idx( 2 , 1 )] = mSin( mDegToRad( 70.0f ) );
   xMatrix[MatrixF::idx( 2 , 2 )] = mCos( mDegToRad( 70.0f ) );
```


TORQUE MATH #3 MATRICES

```
   // Temporary Z-Axis rotation matrix
   MatrixF zMatrix;
   zMatrix.identity();
   zMatrix[MatrixF::idx( 0 , 0 )] = mCos( mDegToRad( 32.1f ) );
   zMatrix[MatrixF::idx( 0 , 1 )] = -mSin( mDegToRad( 32.1f ) );
   zMatrix[MatrixF::idx( 1 , 0 )] = mSin( mDegToRad( 32.1f ) );
   zMatrix[MatrixF::idx( 1 , 1 )] = mCos( mDegToRad( 32.1f ) );

   // Combine Rotation
   rMatrix.mul( xMatrix );
   rMatrix.mul( zMatrix );

   // Set up scaling matrix manually
   sMatrix[MatrixF::idx( 0 , 0 )] = sVec.x;
   sMatrix[MatrixF::idx( 1 , 1 )] = sVec.y;
   sMatrix[MatrixF::idx( 2 , 2 )] = sVec.z;

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

It is worth mentioning that the rotation calculation I created by combining two matrices via matrix multiplication is not the best way to make rotation matrices. In fact, if I changed the combination of steps to this order, my results would no longer match those from ch10_exer_009a().

```
   // Combine Rotation
   rMatrix.mul( zMatrix );
   rMatrix.mul( xMatrix );
```

This is because most matrix multiplication is non-commutative. That is AB != BA, except in special cases. Thus, it is much better to use the Euler rotation (set()) method supplied by the engine).

However, if both rotations had used the same theta, say 45.0 degrees, I could have hand written the rotation matrix like this.

```
   // Always start with an identity matrix.
   rMatrix.identity();
```



```
// Set up rotation matrix manually
rMatrix[MatrixF::idx( 0 , 0 )] = mCos( mDegToRad( 45.0f ) );
rMatrix[MatrixF::idx( 0 , 1 )] = -mSin( mDegToRad( 45.0f ) );
rMatrix[MatrixF::idx( 1 , 0 )] = mSin( mDegToRad( 45.0f ) );
rMatrix[MatrixF::idx( 1 , 1 )] = mCos( mDegToRad( 45.0f ) );
rMatrix[MatrixF::idx( 1 , 2 )] = -mSin( mDegToRad( 45.0f ) );
rMatrix[MatrixF::idx( 2 , 1 )] = mSin( mDegToRad( 45.0f ) );
rMatrix[MatrixF::idx( 2 , 2 )] = mCos( mDegToRad( 45.0f ) );
```

2 Multiplication Order

Answers:

1. No, they do not.

• As I mentioned above, matrix multiplications are not commutative. So, we have to think about what we're trying to do and be sure to apply multiplications in the right order.

2. <3 4 5> and <9 16 25>
3. Series D, since it is a scale multiplied by a translation. (Remember, these operations are non-commutative.)
4. Are these affine matrices? Yes, but if you doubt it you could write this code.

```
if( !tMatrix.isAffine() )
{
   Con::printf("The translation matrix is not affine!");
}

// ... same for rMatrix and sMatrix
```

