Tutorial – Introduction to Matrices

Introduction:

This tutorial serves as an introduction to the basic ways matrices can interact with each other and how matrices are implemented for graphics programming.

You will look at multiplying and transposing matrices by hand before implementing these operations in your own code.

Tutorial:

Try doing these Matrix Multiplications by hand. Refer to the lecture slides if you are unsure about the process:

$$\begin{bmatrix} 9 & 3 & 2 \\ 6 & 3 & 4 \end{bmatrix} * \begin{bmatrix} 6 & 7 & 10 \\ 3 & 9 & 4 \\ 4 & 6 & 9 \end{bmatrix} =$$

$$\begin{bmatrix} 65 & 32 & 7 \\ 3 & 11 & 19 \\ 8 & 3 & 6 \end{bmatrix} * \begin{bmatrix} 14 & 15 & 20 \\ 14 & 22 & 3 \\ 15 & 10 & 13 \end{bmatrix} =$$

Transpose these Matrices:

$$\begin{bmatrix} 7 & 3 \\ 9 & 2 \end{bmatrix}^r =$$

$$\begin{bmatrix} 6 & 4 & 8 & 8 \\ 3 & 1 & 2 & 3 \\ 3 & 9 & 5 & 3 \\ 3 & 5 & 3 & 2 \end{bmatrix}^r =$$

Multiply this matrix and vector:

$$\begin{bmatrix} 3 & 12 & 6 \\ 7 & 10 & 4 \\ 5 & 2 & 9 \end{bmatrix} * \begin{bmatrix} 8 \\ 7 \\ 2 \end{bmatrix} =$$

multiply the above using a row vector

In graphics programming we use square matrices in order to transform positions. In 2D we use 3x3 matrices and in 3D we use 4x4. Don't worry too much about why we use these sizes, it will be covered in the next session. For the 2D games you will be making this year, we need to implement a class for a 3x3 matrix.

For now we will just focus on the operations we've covered so far.

Write a C++ class for a 3x3 matrix. Include a constructor that builds the identity matrix, a function that transposes the matrix and overload the +, - and * operators to add subtract and multiply matrices with each other.

Here is an example .h to work your implementation from:

```
#ifndef _MATRIX3X3_H_
#define _MATRIX3X3_H_

#include "Vector3.h"

class Matrix3x3
{
public:
	Matrix3x3(){}
	~Matrix3x3(){}

	//builds and returns a new identity matrix
	static Matrix3x3 Identity();
	//transposes matrix and returns *this
	Matrix3x3& Transpose();
	//builds and returns a new matrix that is the transpose of this matrix
	Matrix3x3 GetTranspose() const;

	Matrix3x3 operator +(const Matrix3x3& a_RHS) const;
	Matrix3x3 operator -(const Matrix3x3& a_RHS) const;
	Matrix3x3 operator *(const Matrix3x3& a_RHS) const;
	Vector3 operator *(const Vector3& a_RHS) const;

	Matrix3x3& operator +=(const Matrix3x3& a_RHS);
	Matrix3x3& operator -=(const Matrix3x3& a_RHS);
	Matrix3x3& operator *=(const Matrix3x3& a_RHS);

	float m_afM[3][3];
};

#endif //_MATRIX3X3_H_
```