Objectives

1. Understand the differences between linear and affine transformations
2. Understand how to represent required transformations as matrices, such as scaling, rotation, and translation, shearing
3. Understand how to concatenate several transformation matrices as one transformation matrix through matrix-matrix multiplication
4. Understand how to convert coordinates from one coordinate system to another using matrices
5. Learn how to use DirectXMath to impconcerning transformation matrices

All the exercises described below can be done directly based on what you have achieved in Lab 1. However, if you want to see some examples and to get some ideas about how to perform the transformations specified in the exercises, you can have a look at the Tutorial05 C++ source code from the Tutorial 05 project and do the exercises based Tutorial05.cpp.

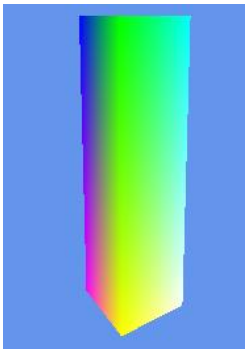
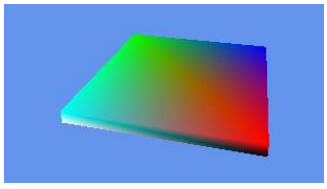
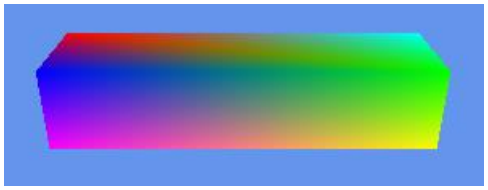
You may want to use the DirectXMath Library Matrix Functions to specify the required transformations. You can find all these functions form https://msdn.microsoft.com/en-us/library/windows/desktop/ee415594(v=vs.85).aspx . The scaling and translation matrices can be directly constructed by using XMMatrixScaling( ), XMMatrixTranslation( ), but there are several different functions in DirectXMath for you to specify a rotation transformation. The DirectXMath vector functions can be found from https://msdn.microsoft.com/en-us/library/windows/desktop/ee415644(v=vs.85).aspx .

Exercise 1.

Transform the cube into following shape using scaling transformation. The required transformation can be specified in the following way, say

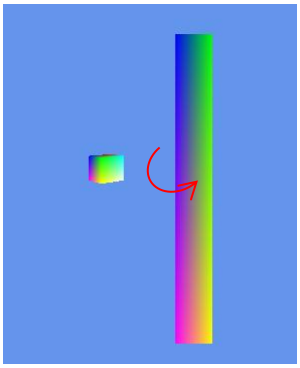
```
XMMATRIX mScale = XMMatrixScaling(2.2f, 0.5f, 0.5f);
g_World = XMMatrixIdentity();
g_World *= mScale;
cb.mWorld = XMMatrixTranspose(g_World);

g_pImmediateContext->UpdateSubresource( g_pConstantBuffer, 0, nullptr, &cb, 0, 0 );
```

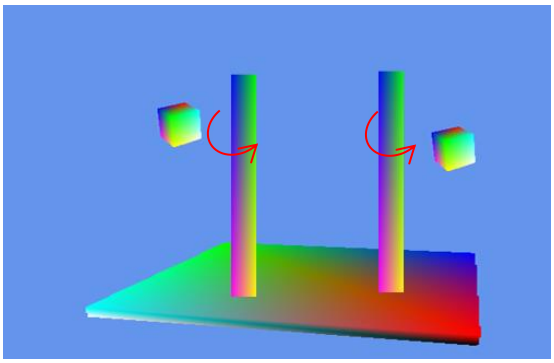




Exercise 2.

Perform scaling, translation and rotation transformation to achieve the following effects: (1) a cube rotates by a vertical rotation axis (as shown in figure (a)); (2). Two cubes rotate by two different rotational axes with different rotational speeds respectively (as shown in figure (b)). The axes are modelled by scaling the cube model provided in the tutorial.



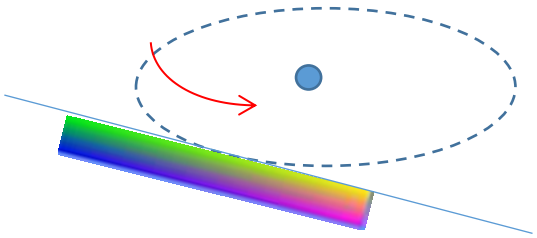
(a)



(b)

Exercise 3

In Exercise 2(a), scale the small rotated cube into a long-thin stick and rote the stick by a rotation axis such that the stick is always tangent to the rotation path, as shown below. If you are able to do this, also consider how to get the stick flying along a general curve.



Exercise 4

Scale the cube into different sizes corresponding to the Sun, the Earth and the Moon respectively and then combine a set of rotation and translation transformations to animate a simple solar system.

Exercise 5

In the Tutorial04, the view transformation and projection transformation are created from the XMMatrixLookAtLH( ) method and the XMMatrixPerspectiveFovLH() method. To develop a better understanding of the two transformations, you can directly define the matrices in your c++ program and observe if you can get exactly the same effect. You can define the two matrices directly using XMMatrixSet( ) method.