Objectives

1. To understand how skinning character animation technique works.
2. To understand how to implement localized transformations, such as scaling and rotation.
3. To understand how HLSL smoothstep( ) can be used to achieve a smooth transition between transformed parts and non-transformed parts of a triangle mesh object.
4. To learn how to implement character animation in Direct3D 11.

Exercise 1. Load sdkmesh module using DXUT

The SDKMesh geometric model used in DirectX samples can be loaded using the mesh loader provided in DXUT, a "GLUT"-like framework for Direct3D 11.x Win32 desktop applications. The information about DXUT can be found in https://github.com/Microsoft/DXUT/wiki/DXUT.

1. Download DirectX 11 Tutorial08-10 DXUT Tutorial Win32 Sample Direct3D11TutorialsDXUT from https://github.com/walbourn/directx-sdk-samples, together with the DXUT and Media folders.
2. Set Tutorial10 as the StartUp project. Compile and run the project. Go through the tutorial and see how the sample works.
3. Create some of your own sdkmesh models using Meshconvert.exe, which can be downloaded from https://github.com/Microsoft/DirectXMesh/releases/download/sept2017/meshconvert.exe
   The syntax on using the converter can be found from the following link: https://github.com/Microsoft/DirectXMesh/wiki/Meshconvert
4. Use an sdkmesh model of your own in Tutorial 10 project. You may get some errors if the material information is not available in your model. You may still be able to view the module if you provide your own shader resource view for the model.

Exercise 2. Locally controlled pulsating character

1. Create a new vertex shader VS_Ex1( ), say, by making a copy of exiting vertex shader.
2. Specify the range of vertices you would like to transform, say, only for vertices whose z-coordinates are larger than 160. This can be done easily by using an if-statement, but the visual effect generated in this way does not look great. A better way of achieving this effect is to use HLSL smoothstep( ) function, which is basically a fuzzy logic-based if-statement. For instance, deform the character in the following way:

```
float scale=Magnitude*smoothstep(140, 160, vAnimatedPos.z)
                                         * (sin(g_fTime)+0.5);
vAnimatedPos += scale*float4(vNormal, 0);
```

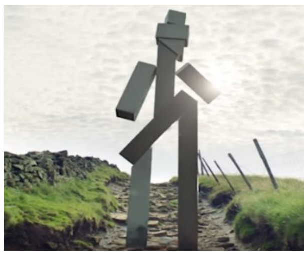
3. Replace the original vertex shader with your new vertex shader.
4. Run the application, you should see a locally pulsating character.
5. Introduce new variables and corresponding UI sliders to control the scaling range and the pulsating frequency of the object.
6. (Optional effort) Write a new vertex shader that create long thin waving hair on the head of the character.

Exercise 3. Per-vertex local skinning transformations

1. Further implement transformations such as translation, rotation, scaling, shearing in vertex shader VS_Ex1( ).
2. Add sliders in UI to perform different kind of transformations.
3. Create following animations using smoothstep() function:
    a. Rotate the head left and right.
    b. Make the character walking.
    c. Make mouth open and close.

Exercise 4. Character animation using a cube model

1. Model a character using a sequence of transformed cubes. Have a look at the video shown at https://www.youtube.com/watch?v=2Ei_XdwSghQ on Channel 4 identity logo animation to get some basic idea on how to build the character.



2. Write a vertex shader to animate the walking of the character.

Exercise 5. Projectile animation

1. Add a mesh object to model a ground. You can use the terrain model you have created in Exercise 8 described in Lab 1.
2. Add a new model, a cube or a sphere, as the football, and animate its bouncing behaviour on the ground.
3. Animate the flying behaviour of the football.