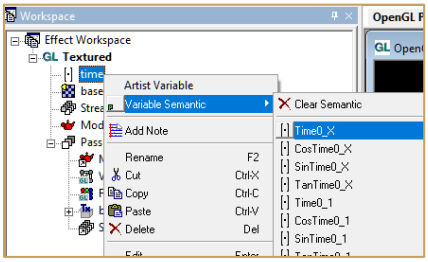
The aim of this lab is to understand how vertex shader works and how to implement geometric transformations and deformation, either uniformly or locally, using a vertex shader written in GLSL.

Exercise 1. Pulsating objects

1. Open the default GLSL default textured effect.
2. Introduce the Rendermonkey's built-in timer into your vertex shader. To add the timer, we first add a float variable in the workspace, say, **time**, and then specify the variable's semantic as **time0_X**. You can specify the semantics of a uniform variable shown in the Effect Workspace by right clicking on the variable and select the required semantics from the drop-down menu.



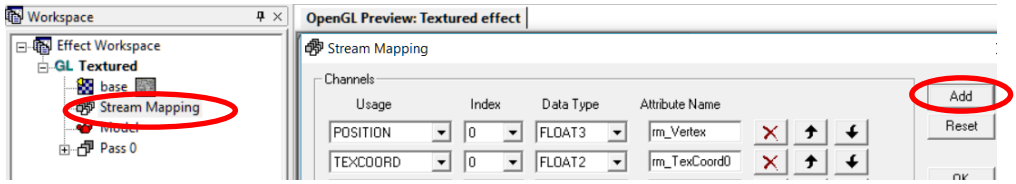
3. Pass the variable 'time' defined in workspace to the vertex shader as a uniform variable:

```
uniform float time;
```

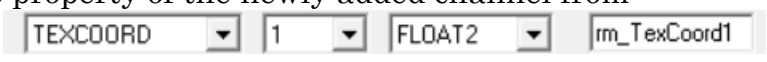
4. Add uniform float variables **S** and **freq** in the vertex program to specify how you would change the size of the object. You can start with some functions that have been considered in the lecture. For example,

```
float disp = S*sin(freq * time );
```

5. If you want to extrude a vertex position along the normal vector associated with the vertex, the normal vector attribute need to be input into the vertex shader. To input the normal vector into the vectex shader, double-click the Stream Mapping node shown in the Effect Workspace and then click the Add button, a new stream channel will be added.



6. Change the property of the newly added channel from



to

7. Edit the vertex shader to translate each input vertex position along the normal associated with the vertex by a mount specified in the variable **disp**:

   ```
   inPos.xyz +=disp * normalize(gl_Normal);
   ```

8. Transform the dynamically changed vertex position into clipping space.

   ```
   gl_ModelViewProjectionMatrix*inPos;
   ```

9. Save and recompile your programs. You should see a pulsating object. You may need to set the values for variables **S** and **freq** properly to create a required pulsating effect.

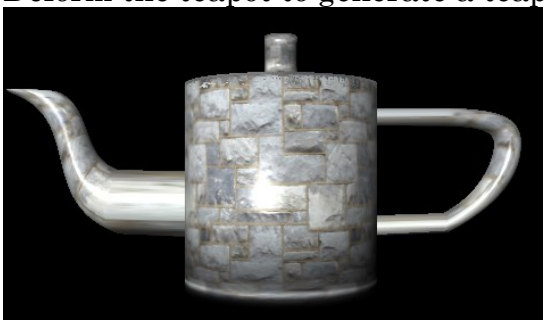
Exercise 2. Waving effects

Modify the function disp defined at step 3 in Exercise 1 to create different per-vertex waving effects. For example,

```
float disp = S*sin(freq * time + inPos.x*inPos.y);
```

Exercise 3. Locally deformed Teapot

1. Open the default Textured Phong effect and change the model to be a teapot.
2. Deform the teapot to generate a teapot with the following form:



Exercise 4. Locally animated pulsating objects

Modify what you have achieved in Exercise 1 such that only the lower part of the object is animated. Make sure that there is a smooth transition between the animated area and the non-animated area.

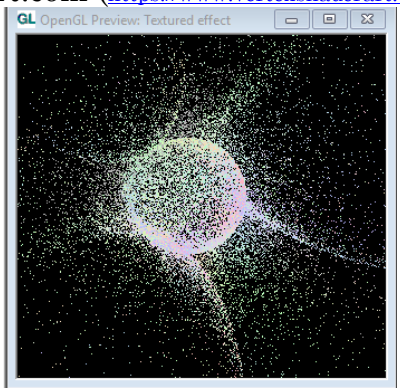
Exercise 5. An animated flying teapot

1. Open the default Textured effect and change the model to be a teapot.
2. Deform the teapot locally to create a teapot with wings shown below:

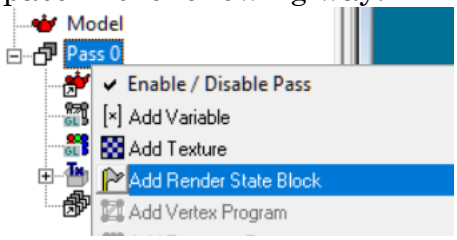
3. Use the idea described in Exercise 1 to animate a flying teapot.

Exercise 6. Point cloud effect

Write a vertex shader in RenderMonkey to achieve the point cloud effect shown in VertexShaderArt.com (https://www.vertexshaderart.com/art/nL6YpkW8YvGKNEKtj).



In RenderMonkey, you can render a triangle mesh object as a set of points by editing the render state. To edit the render state of a pass, first, add the render state block to the pass in the following way.



Double click the added render state node and then edit the rasterization mode for GL_PolyFrontMode from FILL to POINT.

==========

Written by Dr Qingde Li(q.li@hull.ac.uk).
Mar. 11, 2019.