In this assignment you will replace your software renderer with an OpenGL version.

Your assignment is to use OpenGL for rendering. Depending on the current state of your software, there are two approaches you may follow:

- Change the (internal) implementation of the renderer so it will use OpenGL.
- Discard the renderer and let the scene handle calls to OpenGL directly.

You can choose either option. For each choice, there are a few implementation hints you should consider:

**Change Implementation**

- When a model is first created, it should call the renderer to initialize it. The renderer will create a vertex buffer object for the model and return the handle to the model. When the model asks the renderer to be drawn, the model will pass only the handle – not the geometry – to the renderer.
- Shader programs should be managed by the scene, but handled by the renderer. For example, the scene might ask the renderer to initialize several programs when the application starts. The renderer will compile them and return their handles to the scene, so the scene will be able to ask the renderer to bind one of them later.
- Remember, there should be no OpenGL code outside the renderer.

**Discard Renderer**

- Since the renderer is gone, all calls to the renderer should be replaced with calls to OpenGL. This could be fairly straightforward if you follow previous design hints.
- Some of the states the renderer handled previously may now need to be handled by the scene.
In addition to implementing all existing features in OpenGL, you should also implement the following using shaders:

- Texture mapping. Use texture-coordinates when available, and additionally implement two canonical texture-coordinates. To load textures you can use any external image library. We recommend:
  - pngLib. Can be found on the course site. Just include the header files and link the libraries. An example is found in the Image File Formats tutorial
  - Cimg. Found here: http://cimg.sourceforge.net/. This is just a huge header file. There is a short explanation on the site and it is very easy to use
- Normal mapping.
- Environment mapping.
- Toon shading.
- Silhouette rendering.
- Color animation – change the color of the model gradually over time, in some smooth manner. You can use HSV representation to change the hue of each color in the model. Add some other interesting color changing effect of your choice and provide a selection between the two in the UI.
- Vertex animation – Do something similar to the above with the vertices. Namely, move the vertices gradually over time in a non-uniform way.
- Add a volumetric texture turbulence function to the model. You may implement anything from marble to wood as we learned in the tutorial.

You can use a shader designer like Shader Maker to help you with programming the shader.

**SUBMISSION**

Submission is frontal. Before the submission deadline, we will schedule timeslots for you to come and see us. Presentations will last 15-20 minutes, during which you will show us your work and answer our questions.

**FINAL NOTES**

- This is not MATAM – there is no automatic checker. This means that all the features that are to be implemented should be intuitive to the developers with plenty of room for personal interpretation. It also means that the features that you implement should behave quite differently compared to other works. Copying of any kind will not be tolerated!
- DO NOT USE any external code without permission. If you have any doubt, please contact Omri.
- You are very much encouraged to experiment with your program and add more features to it. Previous experiences show this assignment can be addictive!
- You have three weeks to complete this exercise, this time is more than enough, but you are strongly encouraged to start working on it right away.