CG Programming - S0006E - 2016
Week 4 - Input, Graphics Nodes

Shader Resource

Add a *ShaderObject* class that contains the the OpenGL program object and can be used to apply a shader to a rendered object. It has to have the following functionality:

- Loading shader source code from files. One option is to use one argument per shader type (vertex + fragment) or even loading them separately. After (successfully) loading the file compile and link the shader and write any errors to the console. Use proper error handling, do not silently fail if the there are errors while opening the files or compiling.
- Add an apply function that will do a use program call.
- Add functions to modify uniform objects of at least the type Matrix4fv and Vector4f (or Vector4fv). Optional: Add functions for the other types as well. It is a good idea to use a *std::map* for storing the uniform locations once instead of performing a lookup every time.

Graphics Node

Combine all the resources into a *GraphicsNode* class. It contains a *MeshResource, TextureResource* and *ShaderObject* and a transform.

- Store (smart) pointers to the resources and shader objects.
- Add set/get functions for all the resource types.
- Add a draw function that will bind the resource objects, apply the transform and finally render the object.
- It has to be possible to have multiple instances of *GraphicsNode*.

Mouse and Keyboard Input

Add Mouse and keyboard input to the program.

1. Add a *GraphicsNode* instance consisting of a textured cube to your program and use the mouse to apply rotation to it (while the left mouse button is pressed) and use the WASD keys to move it around on the screen.

Delivery

Commit your complete project to a dedicated folder inside your SVN repository (e.g. S0006E/assignment) and upload the number of the revision to Canvas.