Stefan Johansson,
stefanj@cs.umu.se

UMEÅ UNIVERSITY
DEPARTMENT OF COMPUTING SCIENCE

3D Studio—Part 3

Walking in the air

Please, read the whole project description before starting the implementation.

Purpose

This is the third and final part of the project. The aims of Part 3 are to add different lighting models, shading techniques, and textures to your 3D object. Especially you will get a good understanding of the different steps in the geometric pipeline. In addition, a purpose of the final part is to practice to present and discuss the design and functionality of the final result in a small group.

Specification

Extend your software with the following (in addition to the tasks in Parts 1 and 2):

- Implement Gouraud shading using Phong (or Blinn-Phong) lighting model in the vertex shader.
- Change default material of the object and properties of the light source by a GUI.
- Add a texture of your choice to the object. The user should be able to choose in the GUI if the object is shown with a texture or not. Compute the texture coordinates using an appropriate method of your choice (not using the texture coordinates in the OBJ file). It does not need to look perfect and it is not mandatory to read the MTL file.

Any extension or alternation of the functionality is encouraged as long as it does not limit the usability or intention of the assignment.

Bonus tasks

The following extensions give bonus points on the first written exam. Bonus points are only given if Parts 1 and 2 are completed and demonstrated in time and the basic functionalities work. The bonus points can only be used to get a higher grade (not to pass) the exam. Any other addition is also encouraged and can render bonus points if discussed with the tutors **before** handing in the project. A maximum of 6 bonus points are given.

One (1) bonus point each:

- Implement Phong shading (Gouraud shading can be omitted).
- Add a ground floor below your object. The floor can be of any color or have a texture. (A ground floor makes it easier to navigate in the world.)

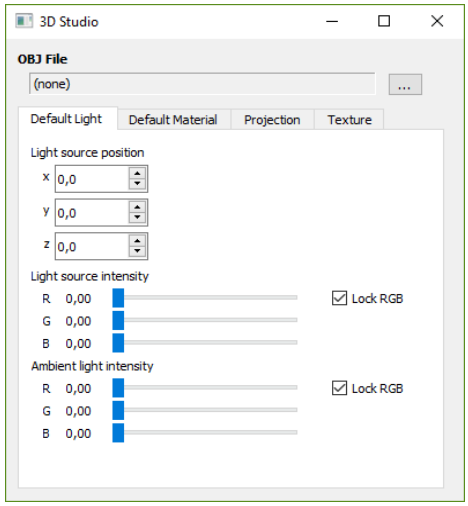
- Implement Toon shading.
- Add a Sky box.
- Read and use the texture coordinates (if present) in the OBJ file.
- Read and use the MTL file.

Two (2) bonus points each:

- Several light sources, including one that moves along a parametric curve, e.g., a sequence of segments forming a closed curve.
- Activate smooth movement of the camera using parametric curves (see last page).
- Several objects in the same scene and/or read OBJ files with compound objects (several objects in the same OBJ file).
- Add mirror or shadow effect on the ground floor and/or object.

The user interface

The GUI from Part 2 is now going to be extended with easy adjustments of the parameters for material, textures, and light properties. It should also be possible to show the object with texture. A GUI is available for you to use, as shown on the right. Code for this can be found under 'Resources/Project' on Cambro. The layout of the GUI may be changed or extended as long as the original functionalities are included. A good idea can be to change the default values to something you think is more appropriate.



Notes on migrating the UI

The best practice to migrate from the UI for Part 2 to the new UI is as follows. We assume you have not changed the name of the files.

1. Unzip all new files in the same directory as the project source files.
2. Copy all the callback functions that you are using from *ass2widget.cpp* to *ass3widget.cpp*. Do not forget to change the class name of these functions to `Ass3Widget`.
3. Remove *ass2widget.cpp*, *ass2widget.h*, and *ass2widget.ui* from the project and add the new files (*ass3widget.cpp*, *ass3widget.h*, *ass3widget.ui*, *qdoubleslider.cpp*, and *qdoubleslider.h*).
4. Do the appropriate changes in the main function and update all includes.
5. Rebuilt all.

External Libraries and Licenses

If you use any existing libraries then it is important that you read the license for each library and check in which circumstances you are allowed to use it. An appropriate disclaimer and any necessary license file(s) must be included in your final source bundle.

Oral presentation

The result of the project (Parts 1-3) must be presented in a smaller group. Notice, the assignment does not have to meet all the requirements listed under the specification, but in a state such that the idea is made clear and the object is visible. The group will consist of 4-6 students and the instructor. The presentation must fulfill these requirements:

1. You do your presentation *logged in at the CS Linux System on a by us provided computer*.
2. The presentation should take about **15 minutes**.
3. You will do a **demo of the software** from the assignments that shows the required functionalities, and extra features if present.
4. Make sure you have at least two 3D models (OBJ files) prepared. If you use the texture coordinates, one should have non (showing that you can apply a texture without pre-computed texture coordinates).
5. You must also **present your system design** (1-2 slides) and your reflections about the current design (1 slide). Have your focus on the responsibilities of different components, generality, modularity, etc. Notice that the presentation should work on the CS Linux System (i.e. use for example PDF, Google Doc, or Impress).

The presentations are done Wednesday January 10 and Friday January 12, 2018. You can sign up for a session on Cambro, starting on January 11. If you do not have specific requirements for a specific time, please wait a day or two before doing so.

At the end of the presentations we will do a short course evaluation.

Instructions

This is an individual assignment. The code should follow good programming practice and be compliant and executable on the computers in MA416 or MA426. If you are using any libraries that are not pre-installed, include them in the separate subfolders ./lib and ./include and set appropriate parameters in the Makefile. No written report is required.

Post your finished solution as a zip-file on Cambro **no later than January 12, 2018, 8:30**. Time for **oral presentation** is provided **January 10 and 12**, in D425 (CS department's conference room Turing on 4:th floor, MIT-building).

Some tips

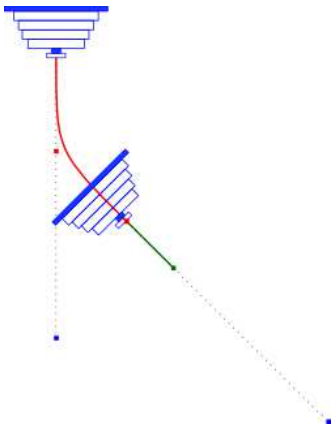
- During development, auto-load an object when the program starts.
- Note that you need the vertex normals to implement the lighting. Only the face normals are provided by the OBJ file.
- Set some default values for the light and material so you don't have to do that every time. In Qt Creator, you can set a default values of all widgets.
- A very good source of information when programming the shaders is the reference card for OpenGL (starting on Page 6):

  https://www.opengl.org/sdk/docs/reference_card/opengl43-quick-reference-card.pdf

Smooth movement of the camera (bonus task)

By activating smooth movement of the camera, the controls start acting differently. For example, then

- (A/Mouse) starts to turn the camera to the left
- (D/Mouse) starts to turn the camera to the right
- (W) increases the speed of the camera
- (S) decreases the speed (not negative)



Speed should be units/time and not units/frame rate and be reasonable corresponding to the size of the world coordinate system.

Paths of the camera should be calculated with a *quadratic* Beziér curve, and line segments. The camera travels by default along the line-of-sight. Each time we gets a turn signal, a new Beziér curve should be computed. The size of the curve should be such that the camera travels along the curve and reaches the end approximately at the same time independent of speed. As the camera reaches the end of the curve, it should continue in the tangent direction. Do not move the camera too fast, and have a shortcut to reset the camera.