

Computer Graphics 203.3710 Assignment #1

Wireframe Viewer

October 17, 2018

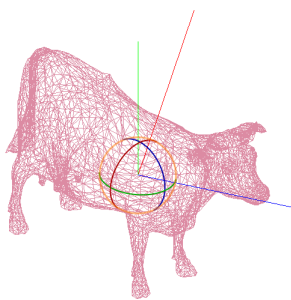
Introduction

In this assignment you will develop the first stage of a modeling software. Your program will render models in wire-frame (edges only) . The emphasis in this assignment is the correct application of transformation and the design of a GUI.

The Skeleton

You are given a code skeleton for your program. The skeleton is only a suggestion. You are allowed to change its structure or ignore it completely. The skeleton includes a cmake script that builds a visual studio solution and consists of five projects:

- glfw: Handles events and windowing (in multiple operating systems). You can ignore it.
- glad: Handles the openGL interface. Ignore it for now.
- ImGui: handles the GUI.
- ImGuizmo: Guizmo addon to imgui.



- nfd: Small library for a cross platform file opening dialog.
- MeshViewer: This is the main part of the skeleton and where you will implement your program. From now on, when we refer to the skeleton, we mean only this project. The project enables ImGui if anyone wants to use it. The project contains all the objects we discussed in the tutorial, implemented to some degree. It is advised that you run the skeleton step by step, and make sure you understand what various parts of it do.

Requirements

Some of the requirements below involve developing some user interactions. Design it however you like, using the mouse and menu, or just the keyboard. As long as you can manipulate the scene in a reasonable amount of effort, it is fine with us. Your assignment is divided to the different aspects of a modeler as follows:

1. Model:

- The user should be able to load an OBJ file and transform it (translation, rotation and scaling) in both the model and world frames.
- If the OBJ file contains normal-per-vertex data, allow the user to choose if the normals (per vertex) should be drawn.
- Compute the normal-per-face. That is the normal to each triangle of the model. Allow the user to choose if normals-per-face should be drawn.
- Compute the bounding box of the model in model frame. Allow the user to choose if the bounding box should be drawn.
- Implement a class `PrimMeshModel` that inherits from `MeshModel`. The class will have at least one geometric primitive (like cube, pyramid, sphere, etc.) hard-coded into it. The user should be able to add a primitive to the scene.

2. Camera:

- Allow the user to add new cameras to the scene (for example, by manually specifying (eye, at, up) vectors tuple or randomly generating one).
- Allow the user to transform the camera in world and view frames, and to select the type of projection (perspective, orthographic) and its parameters (`z_near`, `z_far`, aspect ratio, etc)
- Use camera.obj file to render all existing cameras in the scene - For example, if a camera were added to the scene with (eye, at, up) configuration, render an instance of camera.obj at position 'eye', such that its lens are facing 'eye - at' direction.

3. Scene:

- Allow several models and cameras simultaneously in the scene.
- Allow the user to select the active model. The active model will be the model currently controlled by the user.
- Allow the user to select the active camera. The renderer will render the scene using the active camera transformation and projection. The user will control only the active camera.
- Allow the user to focus the active camera on the active model (use LookAt).
- Allow the user to zoom in/out on the active model.
- BIG BONUS - Allow the user to control the active camera as an 'orbit camera', by dragging the mouse while holding one of its buttons down. Orbit camera is the ability to move the camera around the surface of an imaginary sphere. Here is an [example](#).

4. GUI (Graphic User Interface):

- When the user resizes the window, re-render the scene correctly, while maintaining aspect ratio.

- Allow the user to set the step size of incremental transformation (e.g , when the user moves the mouse to translate an object)
- In general, the UI should be easy to use. At the very least, you should be able to set the position of models and cameras quickly.
- You are allowed to use the [ImGuizmo](#) addon to ImGui, we added it to your project. You can choose to use it, or just implement your own transformation interface. Be aware that ImGuizmo have it's fare share of bugs and complications, thus, both options are challenging in their own way.

How To Begin

Step by step: (This is just a suggestion)

1. First, read again the theoretic material (Bresenham's line algorithm, structure of OBJ files, transformations, the graphic pipeline, etc..)
2. Inspect the skeleton code. Try to understand the flow of the program - begin from the main loop in `main()`.
3. Implement the Bresenham's algorithm - make sure that you can draw lines correctly in all directions.
4. Implement basic camera functionality (view transformation and orthographic projection transformation)
5. Implement the viewport transformation (which transforms projected points on the z-near plane into pixels on the screen space).
6. Add a simple mesh-model into the scene (by loading it from an OBJ file), and add a camera into the scene.
7. Wrap everything together - Make sure that the model vertices are flowing through the graphics pipeline: model vertices \rightarrow view transform \rightarrow projection transform \rightarrow viewport transform \rightarrow draw lines on the screen.
8. Now that your eyes are open, you can see the scene... continue from here.

Submission

Submission is mainly frontal, but you should also commit your code to GitHub classroom. 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 a programming course – 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 two different works. Copying of any kind will not be tolerated!

- DO NOT USE any external code without permission. If you have any doubt, please contact Nave.
- You are very much encouraged to experiment with your program and add more features to it. Previous experiences show this assignment can be addicting!
- You are strongly encouraged to start working on it right away. We know that this assignment could be intimidating, and possibly the largest assignment you will encounter during your studies. We will publish hints and advices from time to time to help you.