Computer Graphics 203.3710 Assignment #1
Wireframe Viewer

July 12, 2018

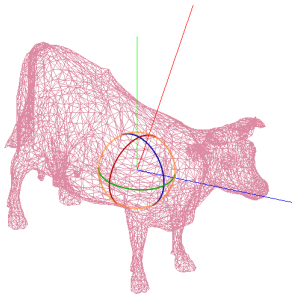
Introduction

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. Geometry:
    - Implement the class MeshModel. The user should be able to load an OBJ file and transform it (translation, rotation and scaling) in both the model and world frames. The function draw() will send the renderer the geometry and transformations of the model, and any other information the renderer might require to draw the model.
    - If the OBJ file contains normal-per-vertex, allow the user to choose if the normal should be drawn. Decide how you want to visualize the normal.
    - 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. Cameras:
    - Allow the user to add new cameras to the scene.
    - Allow the user to transform the camera in world and view frame, and to select the type of projection and its parameters (z_near,z_far, aspect ratio, etc)
    - Allow the user to choose if the cameras should be rendered (as a small plus sign for example)
3. The 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.
    - Implement the function draw(). The function will call the draw method of all the models and will submit the renderer any information necessary in order to render the scene.
4. The renderer:
    - Implement the functions in the renderer's signature. You may add additional methods.
5. GUI:

- 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 challnging in their own way.

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 have two weeks to complete this exercise, this time is more than enough, but 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.