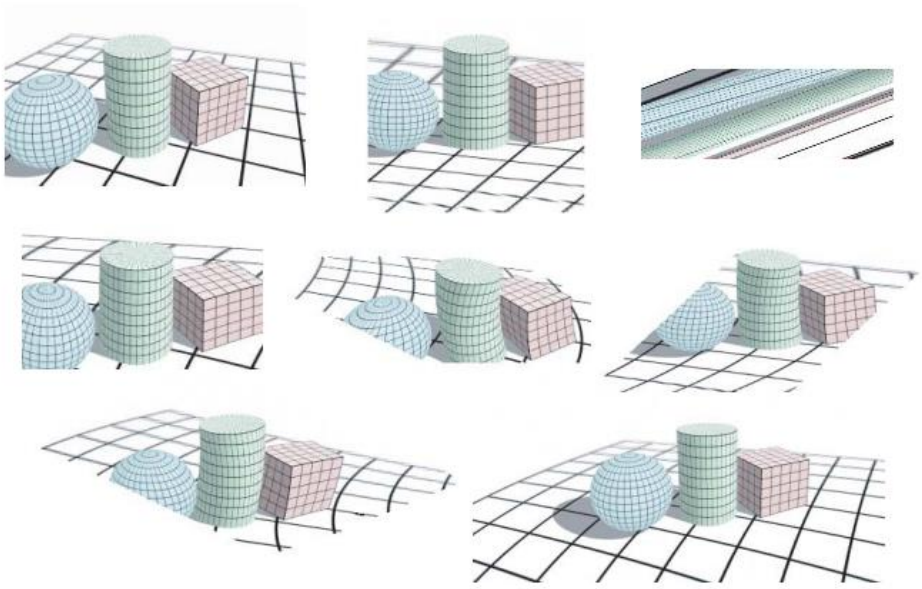
CS535: Assignment #1 – Camera Models

Out: Jan 27, 2026
Back/Due: February 10, 2026

Objective:
This objective of this assignment is to obtain a good understanding of 3D camera models, basic 3D transformations, basic shading, and some geometric computations. You have two weeks and I recommend you start **immediately**. You will write a program that implements a Generalized Linear Camera (GLC) Model and will demonstrate a subset of the possible camera models it can imitate. You program will make use of OpenGL but most of the work will be vector math and geometric computations.

Specifics:
(0) **(5%) Object**. Load a simple object. You may use the .OBJ loader (from assignment 0) to load a simple plane or cube object. You may choose a more complex object but you will have to do ray-object intersections – see next section. Define a background color as well. Minimal requirement is loading a 3D object that is "more complex" than a cube and composed of triangles. Also define a solid background color (not equal to the object colors).
(1) **(20%) Ray-object intersection**. You object should be formed by a set of triangles, thus you need to implement ray-triangle intersection. If your object is not flat then you might need to intersect the all triangles with the ray and pick the closest intersection to the camera or implement some other front to back sorting mechanism.
(2) **(20%) GLC Definition.** This means a structure with a two-plane parameterization of three rays. You are basically implementing equation 1 from the GLC paper (see reference below). Place the first plane at z=0 and the second plane at z=1. To define a GLC you then need to specify the 3 rays. Given those rays and the aforementioned planes you have defined a camera model.
(3) **(20%) GLC Rendering**. To render using the GLC, place an imaginary image plane at z=1 at some chosen pixel resolution (e.g., 256x256 or 512x512). Then, compute the ray corresponding to each pixel and intersect it with the object (e.g., use your ray-triangle intersection). This will result in a pixel color (either of the object or of the background color). Assemble all pixels into an image like structure and draw-to-screen as a texture (e.g., a window size textured quadrilateral). Example of image to texture to screen will be sent to class.
(4) **(10%) Shading and Illumination**: Color each pixel using at least a simple diffuse shading model (e.g., "N dot L"); no need to worry about shadows.
(5) **(25%) GUI**. As a minimal requirement,
    a. (5%) enable using the GUI to rotate/translate the object (or the two planes defined the GLC). This way you can experiment and see the difference between different GLC rays,
    b. enable the user to select from one of the two below camera models:
        i. (10%) pinhole/perspective camera model,
        ii. (10%) another camera model of your choosing (that is NOT orthographic camera model) – please give it a reasonable name and a brief explanation in

the code of what it is; it should produce some sensible projection. You may choose one of the other models from the GLC paper.



(example outputs; you do not need to support the "gridded" lines, but it helps…)

(6) **(Up to 10%)** As extra credit:

a. you can support more elaborate objects,
b. you can implement more complex shading and illumination, and
c. you can extend the GUI so that GLC rays and/or planes can be defined on the fly.

GLC papers: "General Linear Cameras", Jingyi Yu and Leonard McMillan, European Conference on Computer Vision, 2004; and "Multiperspective Modeling and Rendering using General Linear Cameras", J. Yu, Y. Ding, and L. McMillan, Communications in Information and Systems, 7(4), pp. 359-384, 2007.

Grading:
Your program will be tested against the aforementioned functionality and your code will be inspected.

Turn-in:
To give in the assignment, please use Brightspace. Give in a zip file with your complete project (project files, source code, and precompiled executable). The assignment is due BEFORE class on the due date. It is your responsibility to make sure the assignment is delivered/dated before it is due. If you wish to receive confirmation of receipt, please ask by email in advance.

If you implement the extra credit, please ensure instructions are given on how to use it --- put such in the GUI or have clear instructions printout – do not assume we will read your code and decipher how to use your extra credit.