# CS535: Assignment #1 – Camera Projections

**Out**: August 30, 2007
**Back/Due**: September 13, 2007, 8:59am

**Objective:**
This focus of this assignment is to obtain a good understanding of 3D transformations including translations, rotations, perspective projections, and coordinate transformations. You have two weeks and I recommend you start **today**. You will write a program that draws and animates simple objects on the screen using wireframe rendering and gives the user basic 3D navigation tools to move the camera within the scene.

**Summary:**
The assignment is to implement a program which renders three or more bouncing 2D objects within a rectangular-shaped plane. The objects are to lie in the XY plane with Z=0. The width and height of the plane in which the objects can move is determined by the user. The objects will be one square, circle, and triangle (other shapes are optional). The objects will remain completely inside the rectangular plane and will each have an initial position and velocity. Upon hitting the "wall", the object should bounce. The user will be provided with interactive camera controls. All user control is via a GLUI-based interface.

**Specifics:**
(0) Transformations: you will have to write a library and/or object-classes that support point, vector, and matrix manipulations (e.g., addition, subtraction, multiplication, dot product, cross product, etc.) as well as 3D transformations including rotation, scaling, translation, and 3D->2D perspective projection. You may *not* download existing libraries. For this assignment you have to create at least a minimal subset of such functionality in order to implement the required operations of the assignment. Note that you might find this library useful in later assignments and thus applying good software engineering principles is good.
(1) Camera Navigation: using the GLUI-interface, you will provide the user with the ability to translate and rotate the camera viewpoint around the objects bouncing in the XY plane at Z=0. All assignments should include at least 6 "spinners" to change the XYZ position of the camera and then the roll, pitch, yaw of the camera *at the current position* (note: this is not the same as specifying the roll/pitch/yaw and then specifying the XYZ translation). For this assignment, roll corresponds to rotation about Z, pitch corresponds to rotation about X, and yaw corresponds to rotation about Y (this is in logical agreement with the standard camera coordinate frame of "looking down –z axis with y up"). The moving/bouncing objects should continuously move even if the user is interacting with GLUI controls or not (e.g., use GLUT idle function and/or glutPostRedisplay).
(2) Object rendering: each object should be rendered in wireframe and moving continuously. The program shall select a random initial position and velocity for each object (the velocity should be in a 'reasonable' range to generate smooth and not excessively slow nor fast rendering). Upon each newly drawn frame, you will have to

update the position of the object. An object should *never* leave the interior of the bounding rectangle of the object space. This means that if the object touches or will pass the wall in the next frame, it should "bounce" off the wall. Assume a direct specular-type reflection (e.g., incident angle equals reflected angle). Moreover, an object should appear to *exactly* hit the wall before bouncing. This implies changing the object position to the position of exact contact with the wall. To render an object, use a for-loop to render a set of object points along all its edges. The object points will have to be "projected" from 3D space to 2D space. Use a perspective projection transformation for this. Please allow the user to change the field-of-view (FOV) of the virtual camera via the GLUI interface. The default FOV should be 60 degrees. Once all frame rendering is done, draw pixels to screen using glDrawPixels.

(3) For this assignment, you may *not* use OpenGL/GLUT functions or any other downloaded libraries for geometric calculations, manipulations, rendering, or any part of this assignment except as indicated in the template of Assignment #0. You must implement yourself all necessary routines.

**Extra Credit (15%):**

The extra credit item is to implement a virtual trackball interface using the mouse. For this item, you will have to use the GLUT mouse and mouse-button callbacks. The trackball interface mimics there being a "sphere" surrounding the bounded plane containing the objects and then the mouse effectively rotates this sphere on which the camera is fixed. When you "left click" the mouse grabs onto the sphere and then as the mouse moves (and the left button is held down), the sphere rotates according to the movement of the mouse. This enables intuitive rotations around the world space. To provide translations, an alternate mode must be provided. Use the right button to specify translations. Thus, when the right button is held down, the XY movements of the mouse correspond to moving the bounded plane within the *current* image plane (i.e., not in its original XY plane but in the current plane corresponding to the image plane for the current camera viewpoint and viewing direction). Scale factors have to be computed to generate reasonable rotation rates and translation rates.

**Grading:**

Your program will be tested against the aforementioned functionality and your code will be inspected. In summary, the GLUI should at least contain the following controls: bounded-plane width, bounded-plane height, camera x/y/z, camera roll/pitch/yaw, and field-of-view. For all of these, use GLUI "spinners". Please note: for camera control, you cannot use GLUI's built in camera transformation tools (that would, of course, defeat the purpose of the assignment). We will use the interface to alter the aforementioned parameters and expect to see the correct behavior. If the interface does not allow a certain parameter to be changed, it will be considered not implemented at all. If the program does not compile, zero points will be given. The extra credit item should at least implement the functionality as described using the left/right mouse buttons. Additional bells and whistles should be well described so that it can be tested/evaluated.

To give in the assignment, email the TA, as with the previous assignment, by the due date and time. It is your responsibility to make sure the email is delivered/dated before it is due. **Hint: don't wait until the last moment to hand in the assignment**.