COMP 557 - Fall 2020 - Assignment 2
ArcBall and Shadow Maps

Available Tuesday 13 October

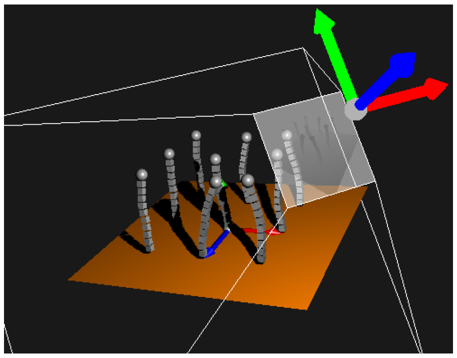
Due 11:30 PM Tuesday 27 October

Getting Started

Download the provided code from McCourses and dump it into a new java project. Add the JOGL fat jar to the build path, and note the updated mintools jar (to help with OSX problems) and vecmath jar should be added too.

You will note that much of the assignment structure matches that of A1, but with some changes. There is a Pipeline class that implements a modeling matrix stack, but there is another GLSLProgram class that sets up the shaders and finds the IDs of common uniforms and attributes. That is, the modeling, viewing, and projection matrices are used in a few different programs: one to draw the light depth, one to draw withe the light depth for debugging, and one for drawing from the main camera with per pixel lighting. Another difference is that the glUniform call to send the modeling matrix (and the inverse transpose) is called every time you change the matrix (i.e., not on push, but on pop, translate, scale, rotate, etc.). See that there is also a multMatrix call you can use to modify the modeling matrix by any given matrix (provided that it is invertible)!

Objectives



The purpose of this assignment is to work with viewing and projection matrices from both a camera at the light (for shadow maps) and from a camera for the drawing the scene. You will also write code to implement ArcBall interaction.

1. ***Implement the TrackBall controls (2 marks)***

   Implement the TrackBall (ArcBall) as discussed in class. Mouse screen positions are projected onto an imaginary ball in the canvas. The fit parameter describes how big the ball is relative to the smallest screen dimension (width or height). With a square window and fit of 2, the ball will just touch the edges of the screen. Values less than 2 will give a ball larger than the window while smaller values will give a ball contained entirely inside.

   Implement the setTrackballVector helper function to set 3D vectors from 2D points generated from mouse events. Note that if the mouse point is not "on" the screen space ball, then you must compute the point on the silhouette of the ball.

   Using normal length 3D vectors for the current and previous mouse positions, compute a cross product to find the axis and the angle of rotation. The axis will be the direction of the cross product, while the angle is the angle between the two vectors. Note that vecmath will be useful for these operations, and that likewise you may like to use an AxisAngle4d to specify the rotation. Note that you must scale the angle you computed by the trackballGain value. A matrix can easily have its rotation set by an AxisAngle4d. Accumulate the rotations in the bakedTransformation matrix, as this is the matrix used in the applyTransformation method.

   You can find more information in your class notes and in [ARCBALL: A User Interface for Specifying Three-Dimensional Orientation Using a Mouse](), a paper inGraphics Interface 1992 by Ken Shoemake.

2. ***Viewing Transformation (2 marks)***

   Set up the camera viewing transformation given the parameters (position, look at, up).

3. ***Projection Transformation (2 marks)***

Set up the camera projection transformation given the parameters (field of view in the y direction, the aspect ratio, the near and far planes). Note that you should follow exactly the projection combined with windowing transformation from the lectures, such that the map to the canonical viewing volume (CVV) has near mapping to -1 and the far to 1.

4. ***Draw Light Frame (1 mark)***

   Use a fancy axis to draw the light coordinate frame. Note a call to disable lighting is at the beginning of the light camera debug code as this geometry will not be well illuminated by the light at the origin of the point light camera coordinate frame.

5. ***Draw Light Frustum (1 marks)***

   The point light camera view projection is set up with some reasonable default values for the fovy, aspect, near, and far. For instance, 55 degrees is reasonable for the test scene, and the near and far should contain the shadow casting objects in the scene.

   The objective here is to visualize the frustum by applying the inverse view and inverse projection matrices for the light view on the modelview mtrix stack, and then draw a wire cube that is 2 units across.

   See code for drawing a wire cube in the geom package. There is code that draws the depth view as a texture on the near plane which may also help you with debugging.

6. ***Draw Light Depth Debug (1 marks)***

   There is a GLSL program to draw the depth texture. The light debug code includes a call to draw a quad with texture coordinates, but in this objective you are asked to set up a transformation so that the quad draw on the near plane of the light frustum. The size of the quad should match the size of light view frustum on the near plane.

7. ***Shadow mapped lighting (1 marks)***

   Your GLSL program needs to compute the point light camera CCV coordinates for each fragment. You will need a windowing transform to convert the -1 to 1 x and y coordinates to be 0 to 1 for the texture, but you will also need a windowing transform to convert the -1 to 1 depth to be 0 to 1 for the shadow map comparison. Compare the light depth of the current fragment with the light depth of the closest surface to the light. Be sure to use sigma as a small offset in your comparison to avoid self shadowing artifacts!

Optional

Up to two bonus marks will be given if you add extra polish to your assignment:

1. Add your character from A1 to the scene (EASY?).
2. Use percentage closer filtering (PCF) to get softer anti-aliased shadows (EASY).
3. Modify the code for multiple lights (HARD).

Finished?

Great! Be sure your name and student number appears is in the window title, in your readme, and in the top comments section of each of your source files.

Prepare a zip file with your source code folder and readme file if you made one. Only use a *zip* archive! Submit it via MyCourses. **DOUBLE CHECK** your submitted files by downloading them. You cannot receive any marks for assignments with missing or corrupt files!! **Treat deadlines as HARD**. If you submit multiple times, only your last submission will be graded

Note that you are encouraged to discuss assignments with your classmates, but not to the point of sharing code or answers. All work must be your own. Please talk to the TAs or the prof if the academic integrity policies are not clear.