

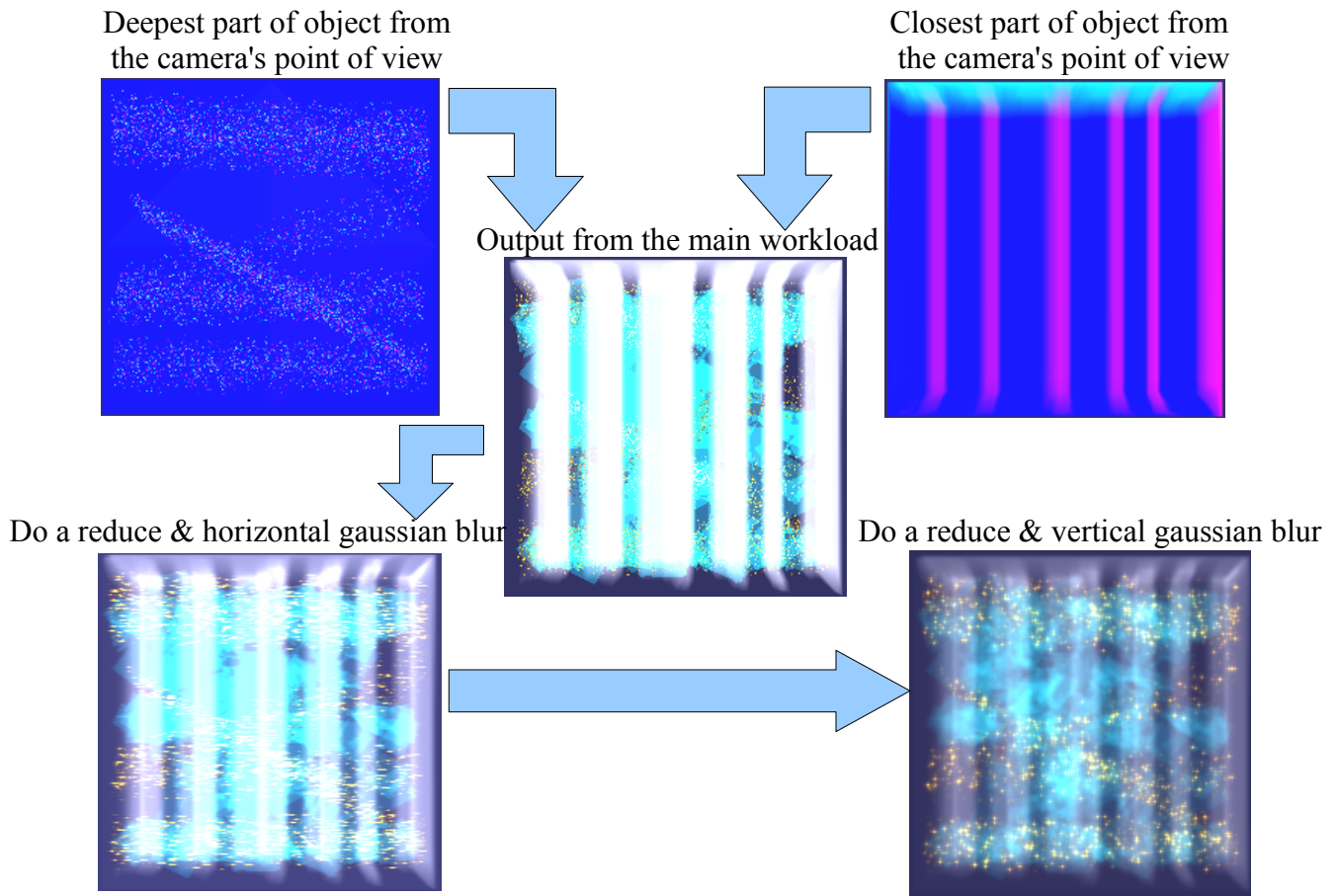
Charles Lohr
CS691G FA2007
Dr. Olano, UMBC
Assignment 2 Readme

The object I had was the strange little pinkish tile with embedded shiny copper.

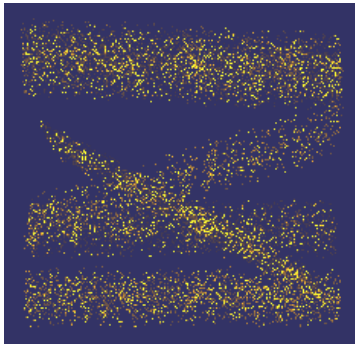
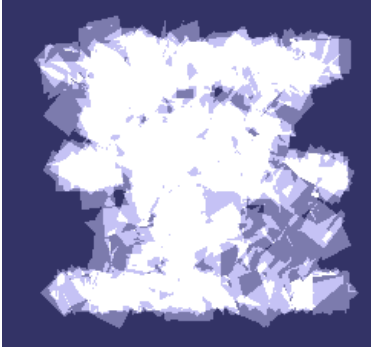
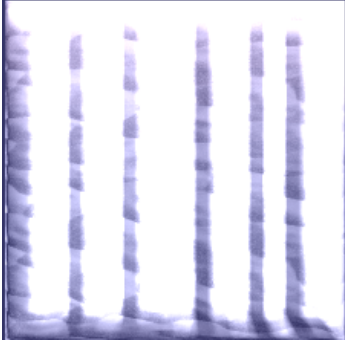
The major things I noticed (most important to least importance):

- Specs of metal
- Milky color
- Semitranslucent
- Bands of milkyness inside are not uniform.
- Very high specularity and perturbed surface

Well, instead of doing a basic list, I felt it made more sense to do a visual diagram of how my series of shaders got put together. Note: Normal mapping is turned off for the main workload shell.



Most of the interesting stuff actually goes on in the the main workload. This includes the generation of the specs, addition of the fluff, and shading of the cremyness and specular shell of the plastic. If you split it up, you can see the three main portions below, with an explanation of them next to the pictures.

	<p>Stage 1:</p> <p>The bands of copper are rendered simply as a series of about 10,000 GL_POINTS. Each point has a normal, normally pointing to the direction of the surface closest to them. They are all rendered with extremely high specularity.</p> <p>In the fragment shader, they compare against the closest Z's. If they are in front, they are discarded, since the copper does not exist outside the object. This lends itself naturally to a speckle look.</p>
	<p>Stage 2:</p> <p>Render the fluff on the inside, this step could definitely use improvement. It's basically a lot of squares rendered in addition to one another.</p>
	<p>Stage 3:</p> <p>This is rendered using the depth information to see how much "cream" color should be added as a homogenous substance material. This value is added to a specular component for the surface to give the shiny plastic feel. Also, note that this is normal mapped to perturb the normal enough to mimic the surface.</p>

With all of these stages put together, it runs on my GeForce 6600M @ 640x480 @ 45FPS. I tested it on a GeForce 7600GS @ 640x480 and it got 137FPS.

I didn't have to use any external sources past what I used originally to write the GLUT utilities and the OpenGL additional code for shaders and such. Most of this came from NeHe and the GLUT stuff was just rewritten from scratch. The model was drawn in Blender, and exported as a .obj file.

There is a limitation or two with this project. The sparkles currently use an arbitrary fixed location for the light source and do not attempt to sparkle with any respect to a specific light. Additionally, the specular stage of the shell is created using an arbitrary light rotating around the shape. When put into an environment, it will be important to add extra lighting around and it and give that environment a sheen to help this object fit in.

Please grade assn2.cpp and the frag/vert files. Note this is not tested on MAC, only on Linux, GeForce 6 Series +. Most interesting stuff is found in the .frag's, especially Shell.frag, and the Specs.frag/vert.