Assignment 5

OS: Windows 10 (64-bit)

Compiler: Visual Studio Code

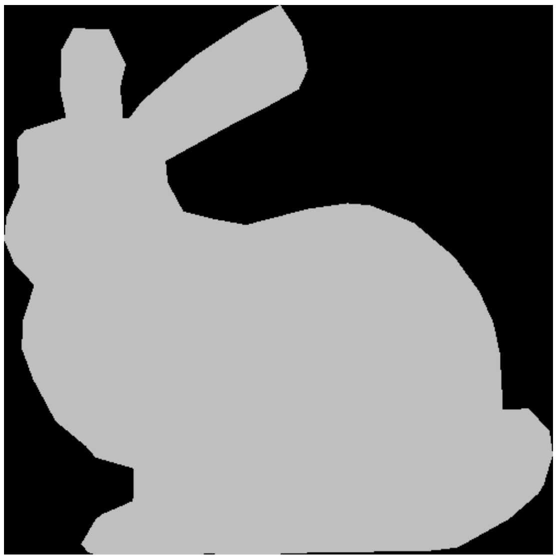
Notice: Sometimes program gives after compilation:

Assertion failed: (internal::UIntPtr(eigen_unaligned_array_assert_workaround_gcc47(array)) & (15)) == 0 && "this assertion is explained here: " "http://eigen.tuxfamily.org/dox-devel/group__TopicUnalignedArrayAssert.html" " **** READ THIS WEB PAGE !!! ****", file c:\users\kuany\desktop\nyu\1st year\fall 2021\math-ga 2270 computer graphics\week 1\cg-master\ext\eigen\eigen\src/Core/DenseStorage.h, line 109

Please check triangle.png or triangle.gif both before executing assignment5.exe and after. Sometimes it executes automatically after compilation, sometimes you have execute it (at least in Visual Studio Code).

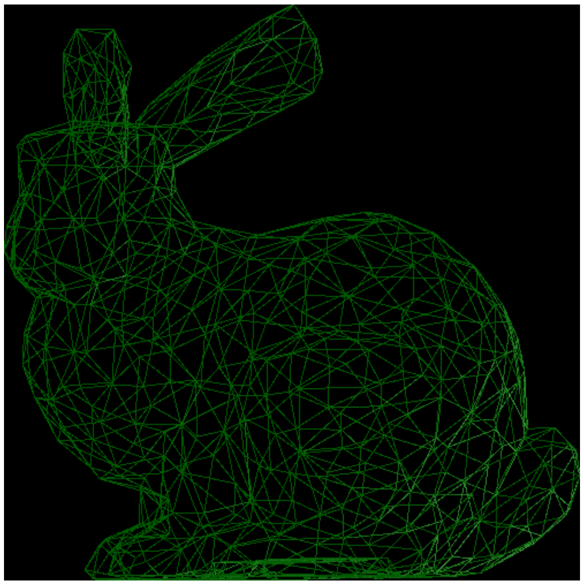
Ex.1: Load and Render a 3D model

Loading part is between lines 108-177. Nothing special just reading coordinates and pushing into vertices. Orthographic matrix is between 219-223 and camera matrix is between 232-237. Camera position is stored in uniform.camera.

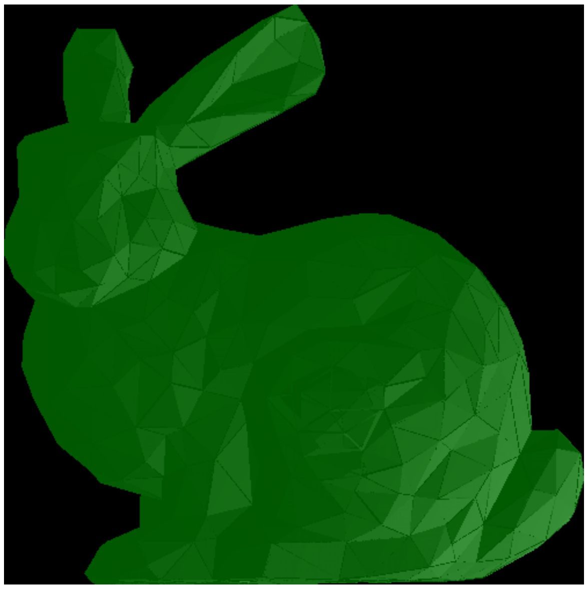


Ex.2: Shading

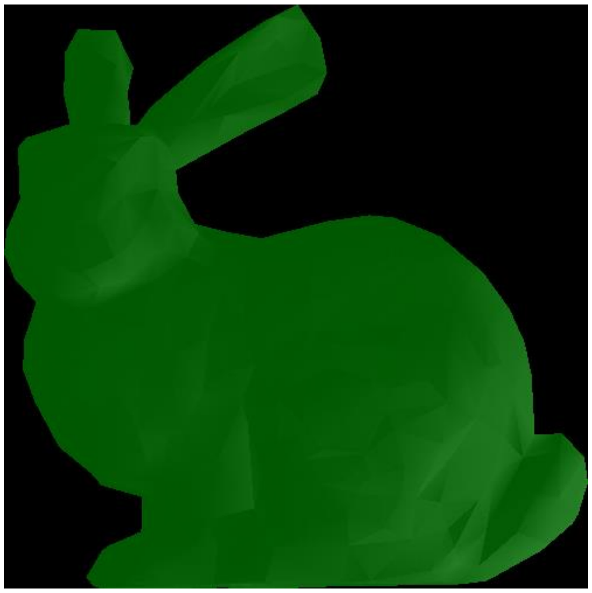
1. Wireframe: First of all shading and light equations are in vertex shader in lines 58-84. Then we load each side of the triangle in lines 150-155 and do the line rasterization.



2. Flat-Shading: Now we just add triangle rasterization and line rasterization together. Normal calculations are in lines 157-159 (just a simple cross product.



3. For this part we have to average the normals. In lines 160-169, we add all the corresponding normals for the specific vector and then in lines 179-181, we normalize them. It’s still a bit sharp, but it’s not a big deal. It just needs to be configured little bit.



Ex.3: Object Transformation

In the folder results you can find all the results (for all three shadings) and gifs. Code is between 256-285. Also, for rotations we just use cos, sin transformation and for rotations above barycenter we just translate it to the zero and translate it back. Translation matrices are between 242-252. I've also added translation not towards camera, but the other way, since when it's too close to camera you can go through the texture (I've added going through the texture gif too). It disappears when it goes to far from camera, because it goes to far from the light (which is near the camera) and light does not reach it.

Ex.4: Camera

For perspective, we just add the new matrix in lines 213-217 and fix parameters in lines 205-209. To not get distortion due to the dimensions we set r = t*aspect_ratio. You can see that it's indeed smaller if it's far away even though we have camera and ortho matrices.