Homework 1

By Anton Volkov

1849922

Table

The table is created with Object Oriented techniques, with a base Entity class where common functions and data would be held and a specialized table class where the table is constructed. The table is defined by 5 cubes, that are transformed to make the shape of the table. Normals are calculated per triangle by getting two sides of the triangle and calculating the normalized cross product. Texture coordinates are simply the association of every quad corner to the corresponding texture corner.

During rendering the coordinates of the vertices are multiplied by the table transform matrix, so to get their world position.

Rotation around point

To implement the rotation around a point, the table centre is set on the point it has to rotate around, done by simple translation, rotated around the set rotation axis, and the translated back by the inverse of the original translation. This operations are done on the table transform matrix, so the table's vertices are just changed in the shader and not recalculated every step, rendering the rendering faster, but the vertices stay in the same place in the CPU and RAM, so not best in case of need of physics simulations.

Perspective projection

Basic implementation with a modelView matrix and perspective matrices, created with the provided functions in the MV.js file.

Spotlight

The spotlight is implemented with a Spotgliht object, that holds the light position, direction, opening angle and attenuation factor. These parameters are then passed to the shaders and used to calculate if a region on the table is illuminated or not. The implementation is simple, but it doesn't account for multiple light sources.

The attenuation from spotlight direction is implemented by $cos^{\alpha}(\varphi)$, where phi is the angle between the LightToPosition vector and the spotlight direction vector. It's essy to calculate but it gives a bug at more than 90 degree angle where everything is set to black.

The attenuation by distance is a simple quadratic attenuation by 1/1+d+d^2 where d is the distance between the object point and the light source position.

In the controls the opening is entered as an angle but saved in radians.

Material

The material is defined as a set of parameters in the Entity, diffuse, specularity, ambience reflectivity and a shininess factor. In the spotlight the same set of parameters are present minus the shininess. After table and spotlight instantiation and setup these parameters are multiplied correspondently and the sent to the shaders to calculate illumination, which is done with a standard Phong shading technique.

Per-vertex and per-fragment shading

Both shadings use a Phong shading technique, adding the spotlight in mind by checking that the angle between the light direction and LightToPosition vector is less than the cosine of the opening of the spotlight, if its less the diffuse and specular are added otherwise only the ambience is rendered.

Wood Texture

Texture is implemented by a wrapper around the WebGL texture object. In the object are held the webgl texture and the texture coordinates, and a function for loading an image to the texture, where the webgl texture object is setup and bound to the image.

Motion Blur

Motion blur is implemented with multiple textures and framebuffer objects. Another framebufferObject is created and in that is rendered the 3D scene onto a texture, while on the default frameBuffer there is a single quad where the previously mentioned texture is rendered onto. There are also other textures, onto which the previous frame is saved, with one frame of difference between each, and those sent to the motion blur shader. The effect is obtained by setting the main frame fragments where there is the background to the sum of previous frames colors with diminishing alpha.