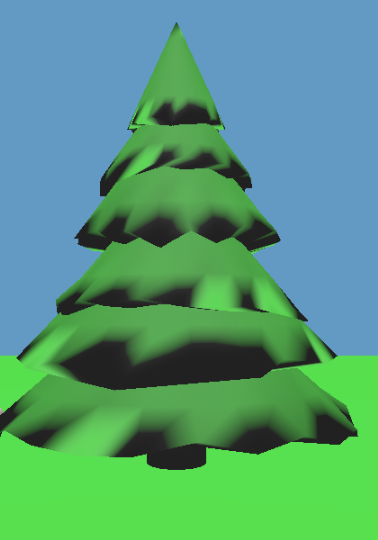
HW3 – Excited bunny

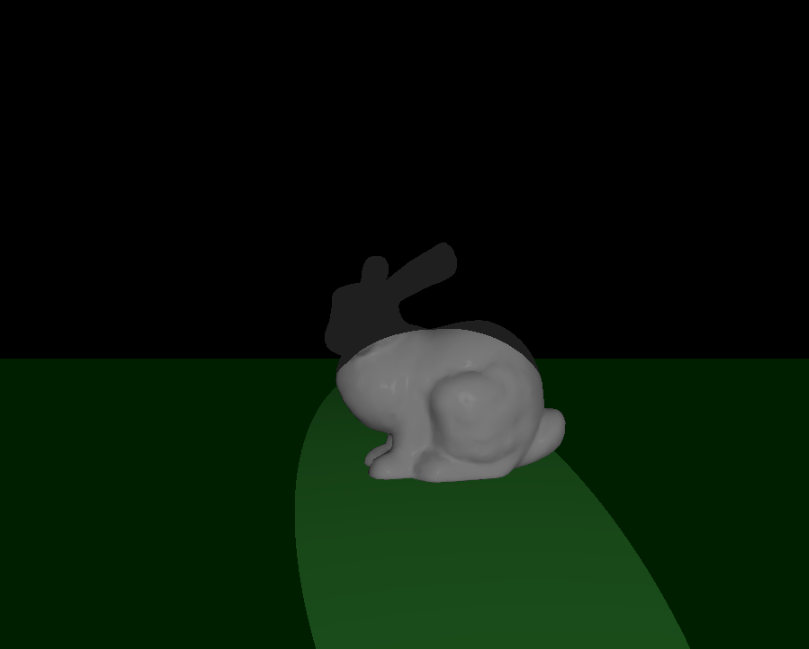
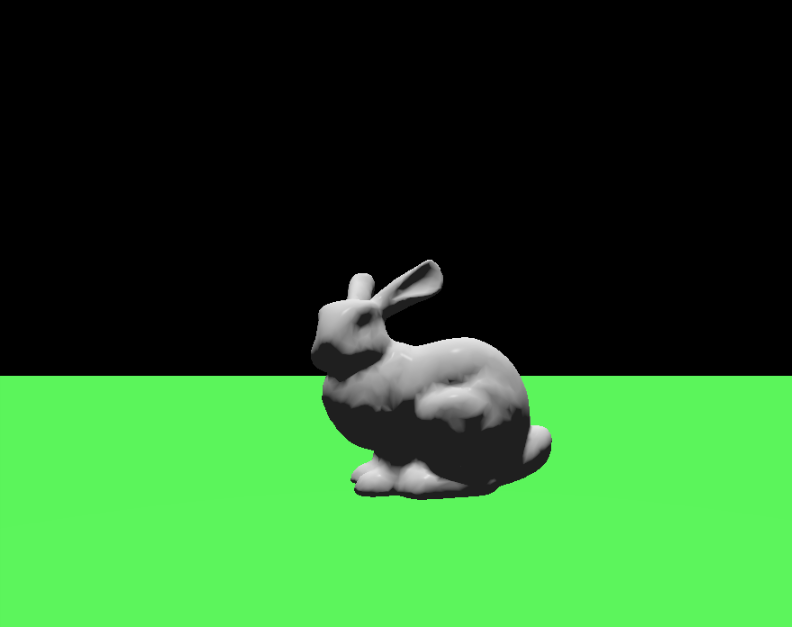
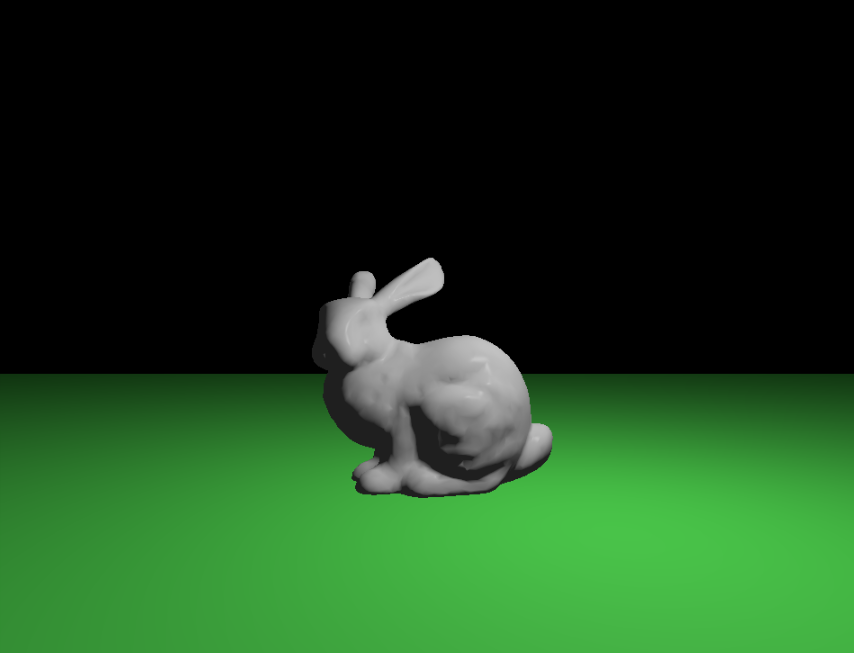
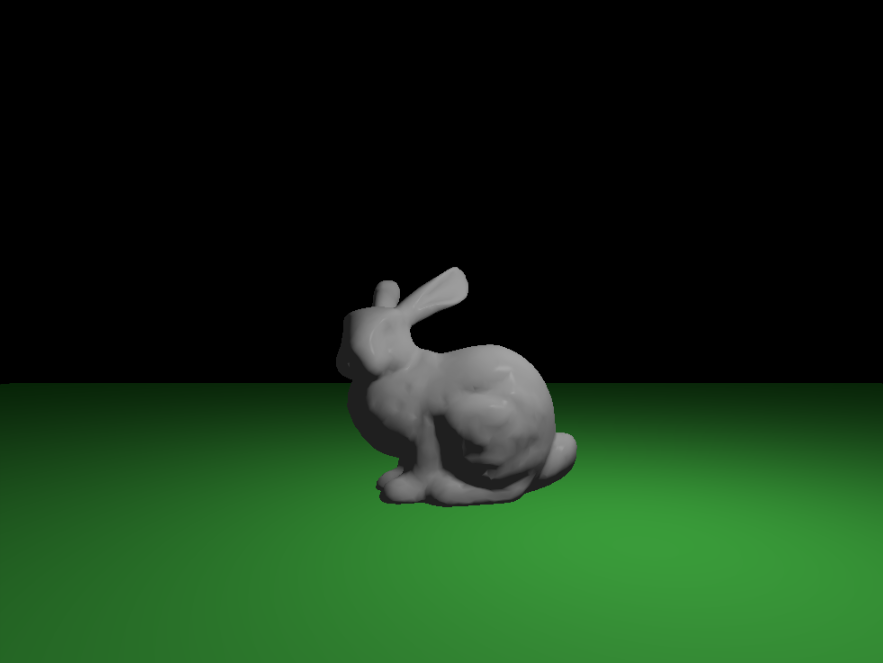
Additional models:

1. Tree



1. Lights
   1. Point Light: I calculated distance between fragment and source, after that defined attenuation as, where r is radius of imaginary source.
   2. Point Light Linear: Since point light is dim, I made additional light type, the difference is that I am not taking square, so that intensity degrade slower.
   3. Directional light: usually toLight vector is defined as LightPos –FragPos, but in this case, I consider position of light as it’s direction, toLight = -LightPos
   4. Spot Light: The idea is to compute angle between direction of light and toLight vector, say , if it is higher than given angle, set spotlight constant to 0, else I assign it’s color as function of that angle

Comparison of lights



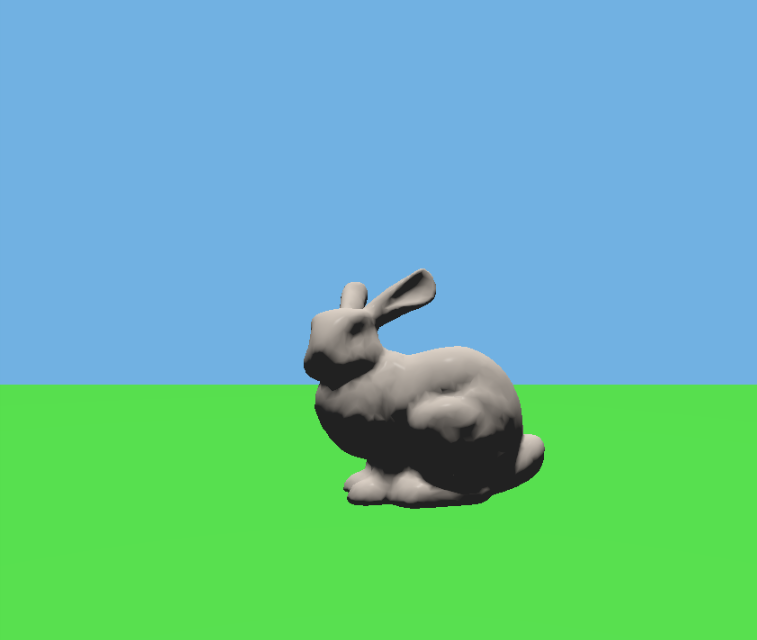
Same order as above.

1. Shaders
   1. Phong shading: Let L = to light vector, E = to camera, H = half angle, N = normal. All normalized. To get diffuse coefficient we need to calculate dot (L, N). specular coefficient calculated as

After that total is computed as

* 1. Toon Shading: I have 4 levels. They are determine by dot(L, N), for each level assign specific intensity, after that do same computation as in phong, and multiply by intensity.

1. Animations
   1. Cycles day and night

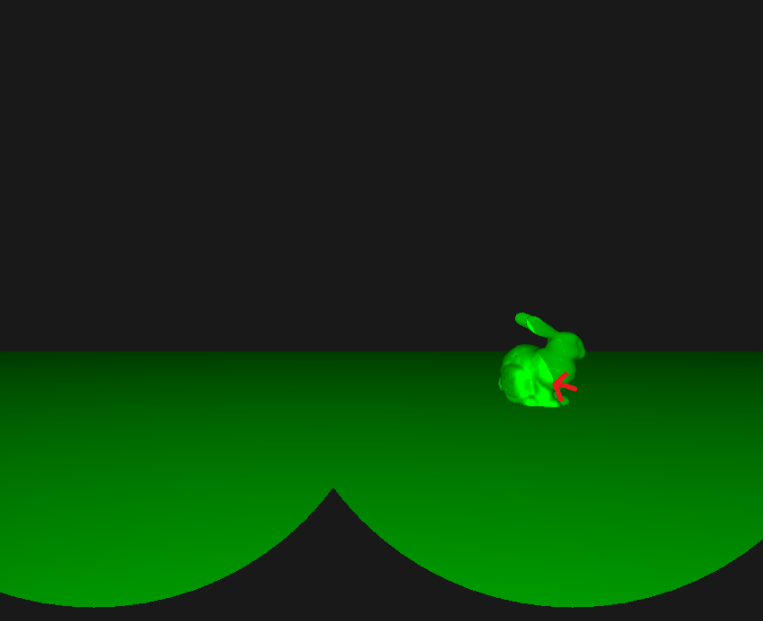


Every 2.5 sec my sun switches off and on, so 1 day is 5 sec. At the same time I change color of background accordingly.

* 1. Smooth animation: Bunny is jumping around tree in circle. I simply used loop to create keyframes that resembles circle, also sun rotates in world, the same implementation. Orbits of bunny and sun

Creativity:

1. Night vision material: We can see object regardless of whether light is present in scene or if it is dim, exception is when we have spotlight then this area is brighter.



I have also used stencil buffer to restrict area of vision, so that it seems like we are looking through binocular.

Fog: Phong material now have fog at a distance



For convenience, I added camera motion. Arrows up, down, left, right. Camera always face origin.