Com S 336
Fall 2023
Homework 5

Please submit an archive on Canvas including the files indicated at the beginning of each problem. The third problem is actually quite short, but will require you to read and modify some of the shader code for doing lighting and textures.

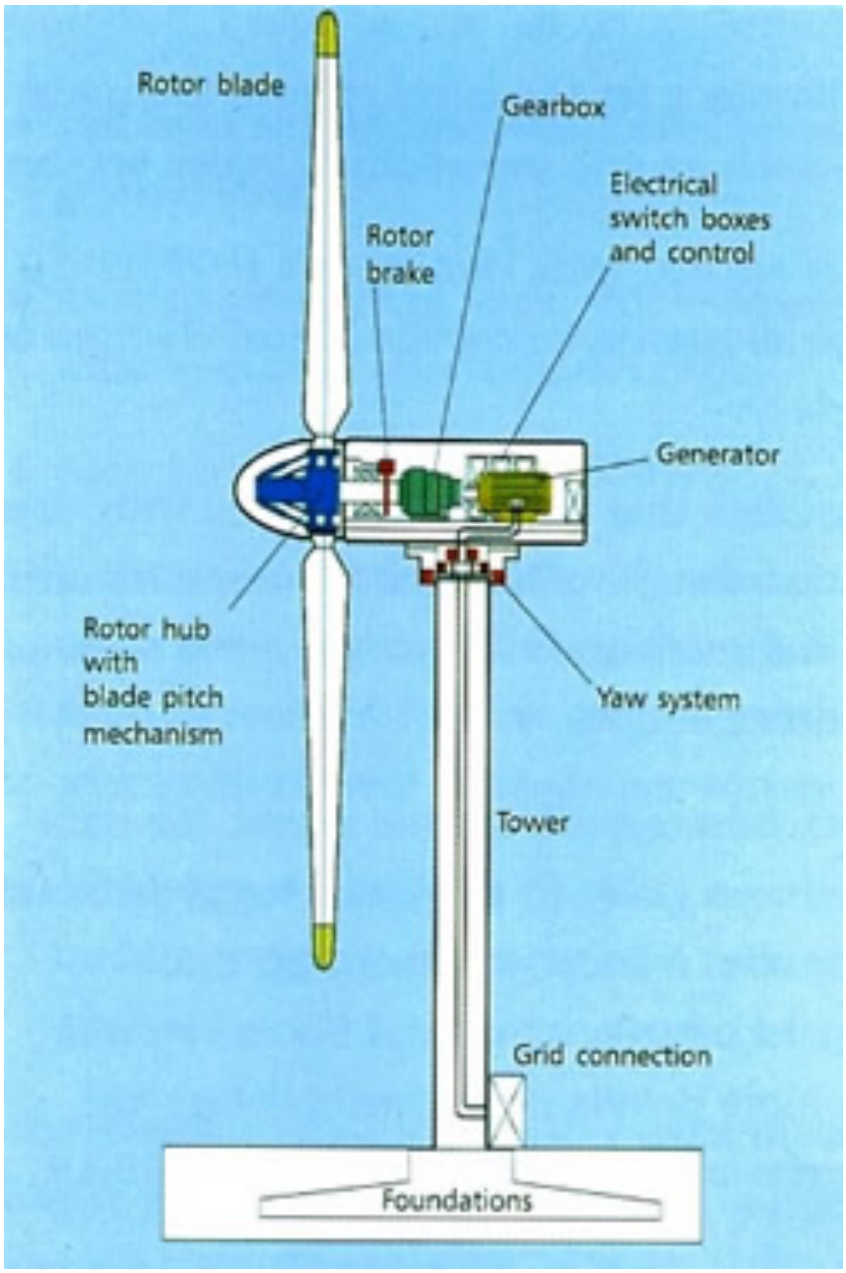
1. (*Please turn in your modified version of HierarchyWithTree3.js*.)

Create a hierarchical scene using the techniques of examples/hierarchy/HierarchyWithTree3.js. Your basic task is to create a clunky model of a wind turbine using five simple objects. Parts should include

- a vertical shaft, or tower,
- a generator housing at the top of the shaft,
- a rotor hub, and
- two rotor blades.

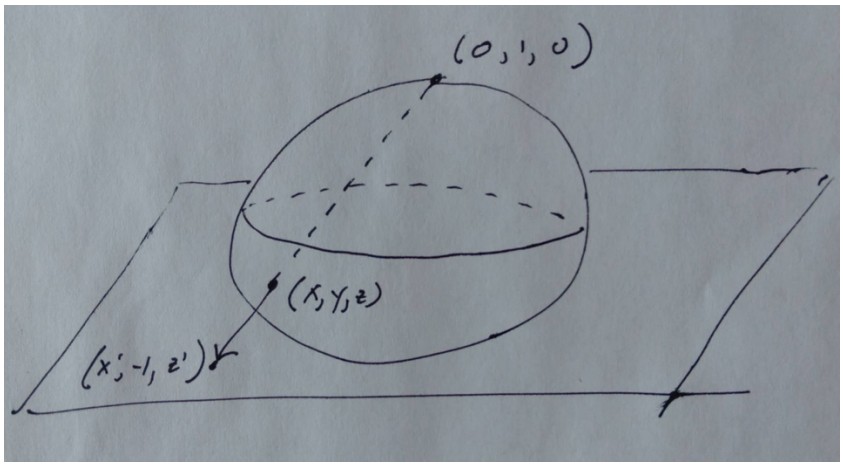
The housing should turn on the vertical shaft (the "yaw" control to make it face the wind), the rotor should rotate, and the blades themselves should rotate relative to the rotor to adjust the "pitch" of the blades (angle relative to the rotor). Animate the rotor to rotate continuously, and add some keyboard controls for yaw and the blade pitch. Instead of cubes, use scaled spheres for the blades and scaled cylinders for the other parts.

On the next page is a basic diagram. (Obviously you should ignore internal details such as the "Generator" and "Rotor brake" and "Foundation".)



2. *(Please turn in your modified version of LightingWithTextureAndModel.js)*

Write a function to generate stereographic texture coordinates for a sphere. Here is the general idea: Imagine a sphere with radius 1, centered at the origin and sitting on the plane y = -1. Given a point (x, y, z) with y <= 0 (that is, it's in the southern hemisphere), extend a vector **v** from the point p = (0, 1, 0) (i.e., the north pole) through (x, y, z) and find the point (x', -1, z') where it intersects the plane y = -1.



One way to think about this is to write

(x', -1, z') = p + a**v** = (0, 1, 0) + a(x, y - 1, z)

where a is the scale factor needed for the vector **v** to intersect the plane y = -1 when its tail is at p. Looking at the y coordinate, you can write -1 = 1 + a(y - 1), or a = 2 / (1 - y) to solve for a. Then it follows that

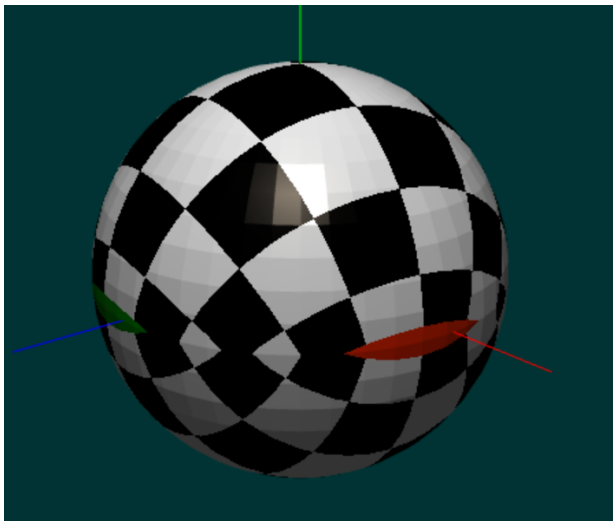
x' = ax = 2x / (1 - y)
z' = az = 2z / (1 - y)

Notice that the values x' and z' are in the range [-2, 2], so rescale them to lie in the range [0, 1] and use those numbers as the s and t values for vertices in the southern hemisphere. Do a similar calculation for the northern hemisphere.

Create a JS function `createStereographicTextureCoords` that, given a Float32Array of vertex coordinates for a model, returns a parallel Float32Array of the stereographic texture coordinates. That is, we could write:

```
theModel = getModelData(new THREE.SphereGeometry(1, 48, 24));
theModel.texCoords = createStereographicTextureCoords(theModel.vertices);
```

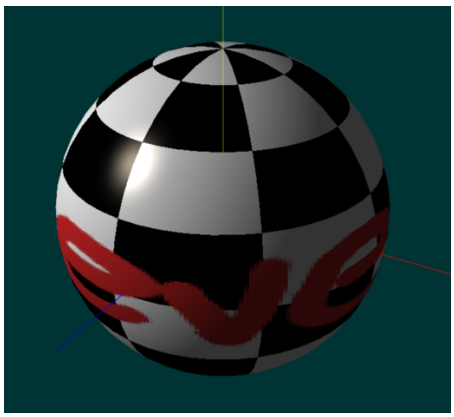
and it should render something like the screenshot below. Notice that for this to work sensibly, we a) can't have any polygons that cross the equator, and b) have to assign the texture coordinates to the vertices on the equator in a consistent way (same values for upper and lower hemisphere). Try this out by adding the function to LightingWithTextureAndModel.js.



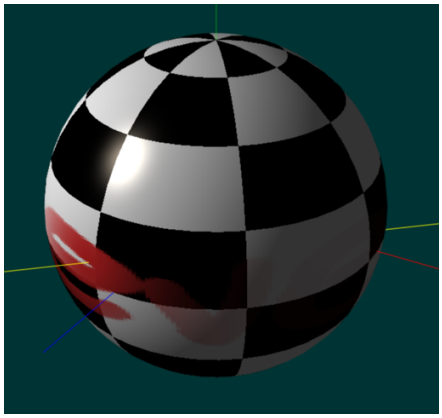
3. (*Please turn in your modified version of LightingWithTextureForHW5.js*)

The code **`examples/homework5/LightingWithTextureForHW5.js`** is similar to the previous example LightingWithTextureAndModel but it has the spot light direction and controls from homework 4 built into it. However, the spot light data is not currently used for anything except to draw the yellow line indicating the spot direction. (The light that is actually used for lighting the scene is in a fixed position.). Out of the box, it is set up to use an image with a transparent background and "decal" it over the existing surface color.

a) Modify the application so that it has two textures. Use one for the surface color, and one for the "decal" effect. (We did this in class; see examples/texture/LightingWithTextureAndModelTwoTexture.) Using a checkerboard for the second texture, it would look like this, for example:



b) This part creates a cool effect that is surpisingly easy. Imagine there is secret writing on the surface of an object that can only be seen by shining an ultraviolet light or some special wavelength on it. You have a special flashlight (the spot light) that shines the required wavelength, so within the beam of this flashlight, the writing becomes visible. Otherwise, the flashlight has no effect on the scene, since the "light" it emits is not visible light. It might look like the screenshot below (note the direction of the yellow line, indicating what would be the spot direction). To implement: generate two surface colors, one without the decal, and one with the decal included. Use the spot factor to blend them.



For a better illustration, see
https://stevekautz.com/cs336f23/examples/homework5/animation2.mp4