WebGL Assignment 1

Steps to run

1. Install Node.js and Visual Studio Code.
2. Open this lab folder in Visual Studio Code.
3. Open a terminal (Terminal > New Terminal).
4. run `npm install` . If it failed for any reason, try again.
5. run `npm run watch` .
6. Ctrl + click the link shown in the terminal (usually it will be http://localhost:1234).

Note: you can use yarn to enable caching so that you don't download all the packages with project. You can download yarn from yarnpkg.com. Then replace `npm install` with `yarn install` and `npm run watch` with `yarn watch`.

Requirements

Requirement 1: **ColoredSphere**

Modify the function `ColoredSphere` (in `src/common/mesh-utils.ts`) to return a colored sphere.

The function takes 3 parameters:

- `gl: WebGL2RenderingContext` which is the WebGL2 context.
- `verticalResolution: number` which defines the number of segments from the north pole to the south pole of the sphere.
- `horizontalResolution: number` which defines the number of segments along the equator of the sphere.

The function must return a mesh that contains a sphere of radius 1 centered around the origina (0,0,0) and the number of vertical and horizontal segments must be as defined by the input parameters. The color of each vertex will depend on its position and must follow the following formula:

```
(r,g,b) = ((x,y,z) + 1) / 2 * 255
a = 255
```

Requirement 2: **SolarSystem**

Modify the function `drawSystem` (in `src/scenes/02-SolarSystem.ts`) to draw the whole given solar system.

The function takes 2 paramter:

- `parent: mat4` which is the transformation of the parent to the homogenous clip space "MVP".
- `system: SolarSystemDescription` which defines the stucture of the solar system.

The output of this requirement will be compared to the expected output using blink-diff.

The data for the solar systems is found in `static/data/solar-systems.json` and it is loaded by the scene before starting.

Hints

Drawing a sphere in the euclidean xyz-space is equivalent to drawing a plane in the spherical rθφ-space. To get the shape a sphere, you can follow this thought process:

1. Imagine a rectangle parallel to the θφ-plane which covers along the θ-axis from `-PI/2` to `PI/2` and along the φ-axis from `-PI` to `PI`. The plane will be at `r=1`.
2. Subdivide the rectangle into smaller rectangles where the number of subdivision along the x-axis equals the `horizontalResolution` and the number of subdivision along the y-axis equals the `verticalResolution`.
3. Convert every vertex in the subdivided rectangle from spherical coordinates (r, θ, φ) to euclidean (x, y, z). This is now a sphere.

Note: r is the radial distance, θ is the polar angle, and φ is the azimuthal angle. You can read more about spherical coordinates on Spherical Coordinate System.

Now you need to write that generates the same result as the thought process. However, there are some optimizations that can be done:

1. All of the points on both ends of the φ-axis will meet (φ=`-PI` and φ=`PI`) and become the same so we can remove one side and save the space needed by `verticalResolution + 1` vertices.
2. All points at θ=`PI/2` will collapse to a single vertex. The same happens at θ=`-PI/2`. We can only keep one vertex and this will save us `2 * horizontalResolution - 2` vertices.

Delivery

The deadline for the lab delivery is **Saturday 9/11/2019 at 23:59pm**. It should be delivered by mail and will discussed in the labs in the same week.