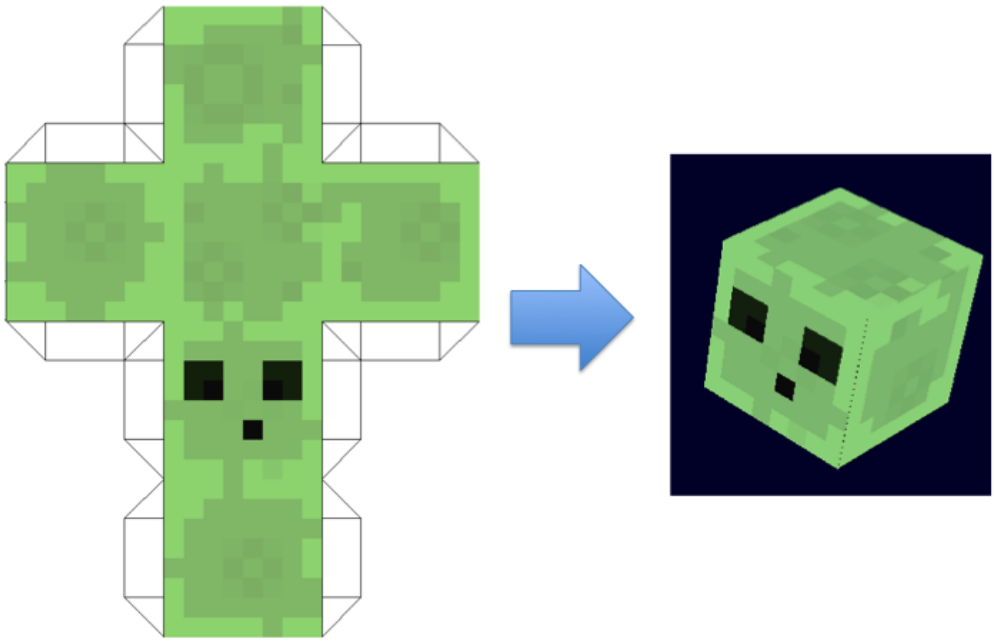
COSE436 Interactive Visualization (Fall 2021)
Instructor: Prof. Won-Ki Jeong
Due date: Nov 21, 2021, 11:59 pm.

Assignment 2: Texture and Environment Mapping

The goal of this assignment is to learn how to use texture mapping and environment mapping for realistic polygonal mesh rendering. The following sections describe the functions you need to implement.

1. 2D texture mapping (30 points)

Using the provided images of Minecraft characters, write a WebGL code to render 3D models as shown below (this is the simplest example, and you can create more complicated models using the images provided):



To do this, you need to load the image, create a 2D texture, and assign per-vertex texture coordinate. You can find the pixel location on the image using an image editing software (photoshop, paint.net, gimp, etc). You also need to design the 3D model's geometry manually (e.g., cubes).

Image files (jpg, bmp, etc) can be loaded using Image() function in Javascript as follows:

```
var image = new Image();
    image.onload = function() {
        configureTexture( image );
    }
    image.src = "mob.bmp";
```

Note: Some web browsers (e.g., Chrome) do not allow to open the local files due to security issue. I confirmed that *Firefox* browser does not have such issues.

Grading:
- Render accurate texture-mapped Minecraft Mob character (10 pts)
- Animate the object (rotation, translation, etc) (10 pts)

2. Environment mapping using cube mapping (30 points)

You need to implement environment mapping using a cube map texture. A cube map texture consists of six 2D textures, and each 2D texture can be created as follows (below example is generated from image1):

```
gl.texImage2D(gl.TEXTURE_CUBE_MAP_POSITIVE_X ,0,gl.RGBA,
              texSize,texSize,0,gl.RGBA,gl.UNSIGNED_BYTE,
              image1);
```

`image1` is the array storing raw texture values for positive X face. You need to do this six times for `GL_TEXTURE_CUBE_MAP_POSITIVE/NEGATIVE_X/Y/Z` to generate all six faces. The code for generating synthetic checkerboard texture images is given in the skeleton code as follows :

```
function configureCubeMap() {
      // Generate checkerboard texture
      for (var i = 0; i < texSize; i++) {
            for (var j = 0; j < texSize; j++) {

                  var c = ((((i&0x1)==0)^((j&0x1))==0))*255;

                  image1[4*i*texSize+4*j]   = c;
                  image1[4*i*texSize+4*j+1] = c;
                  image1[4*i*texSize+4*j+2] = c;
                  image1[4*i*texSize+4*j+3] = 255;

                  image2[4*i*texSize+4*j]   =  c;
                  image2[4*i*texSize+4*j+1] =  c;
                  image2[4*i*texSize+4*j+2] =  0;
                  image2[4*i*texSize+4*j+3] =  255;

                  image3[4*i*texSize+4*j]   =  c;
```

```
                    image3[4*i*texSize+4*j+1] =  0;
                    image3[4*i*texSize+4*j+2] =  c;
                    image3[4*i*texSize+4*j+3] =  255;

                    image4[4*i*texSize+4*j]   =  0;
                    image4[4*i*texSize+4*j+1] =  c;
                    image4[4*i*texSize+4*j+2] =  c;
                    image4[4*i*texSize+4*j+3] =  255;

                    image5[4*i*texSize+4*j]   =  255;
                    image5[4*i*texSize+4*j+1] =  c;
                    image5[4*i*texSize+4*j+2] =  c;
                    image5[4*i*texSize+4*j+3] =  255;

                    image6[4*i*texSize+4*j]   =  c;
                    image6[4*i*texSize+4*j+1] =  c;
                    image6[4*i*texSize+4*j+2] =  255;
                    image6[4*i*texSize+4*j+3] =  255;
            }
        }
}
```

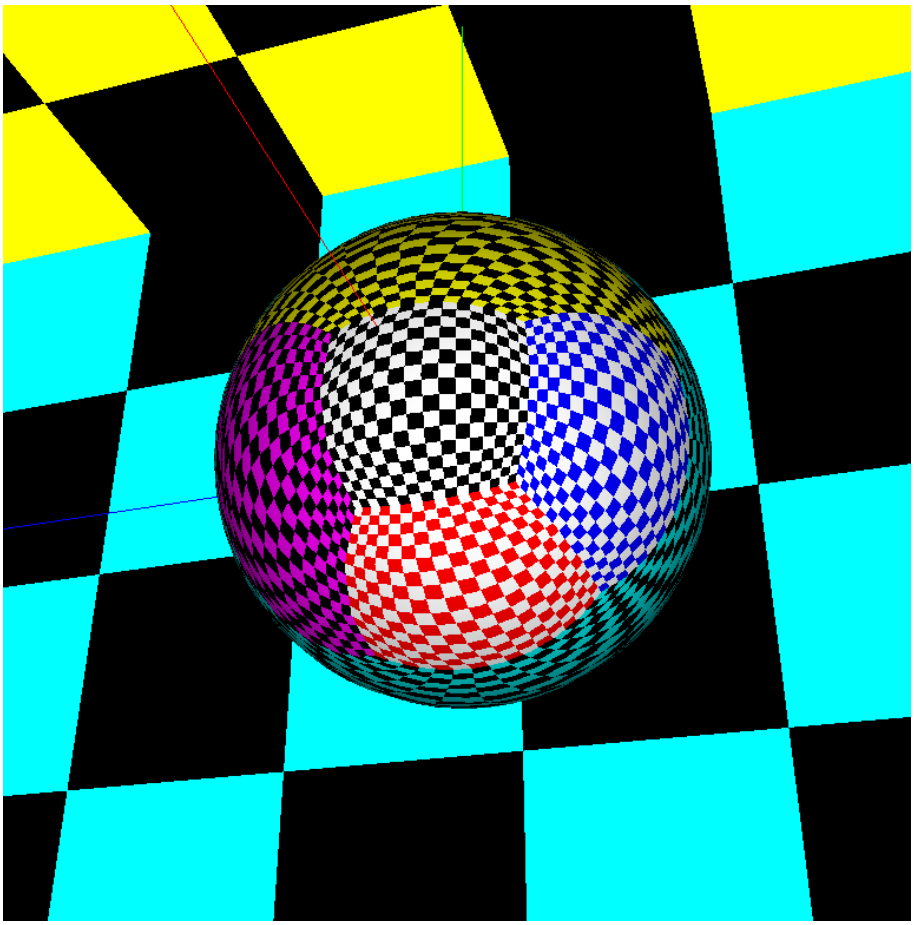
When you create your texture map, use texture wrap mode and min/mag filter mode appropriately. For example, below `nearest neighbor` for min/mag filtering.

```
gl.TexParameteri(gl.TEXTURE_CUBE_MAP, gl.TEXTURE_MAG_FILTER, gl.NEAREST);
gl.TexParameteri(gl.TEXTURE_CUBE_MAP, gl.TEXTURE_MIN_FILTER, gl.NEAREST);
```

Similar to other textures, you need to assign per-vertex texture coordinate. For a cube map, texture coordinate is a (s,t,r) 3D vector and actual texture sampling is done by shooting a ray from the center of the cube to the given vector direction. This direction should be calculated as a reflection vector defined by the surface normal and view direction.

In order to test the correctness of the cube map, you should draw a box whose faces are texture-mapped checker board textures generated above, and render a 3D object (sphere) inside the box with a cube map texture. Sphere.js external library is provided in Common directory (Sphere.js code is from Songho Ahn, http://www.songho.ca/glsl/files/js/Sphere.js).

The below is the screenshot of environment mapping on a sphere using a synthetic checkerboard cube texture map.



Grading:
- Render a cube mapped object as shown above figure (10 pts)
- Correctness of reflection (10 pts)
- Animate the object (rotation, translation, etc) (10 pts)

3. Build your own Minecraft game (35 points)

The goal of this problem is to create a simplified version of Minecraft game using texture and environment mapping. For this, you should implement the followings:

- Create a 3D scene using textures, such as walls, floors, objects, etc.
- Create Minecraft characters and place them in the scene created above.
- Make animations or interactions (key press, etc) so that the characters can move and execute some tasks (e.g., holding a brick, creating a new brick, etc)

Note that you must use environment mapping in the game, i.e., create some shiny object, such as mirror balls.

Grading:
- Scene modeling (10 pts)
- Minecraft character modeling. Create at least three complete textured characters (complete means a character having head, body, arms, legs) and use them as moving objects. (10 pts)
- Animation of objects (5 pts)
- Find interesting cube map textures on the Internet and use them. You can google "cube map image" (5 pts)
- Creativity (5 pts)

4. Report (5 pts)

Submit a report describing your code, implementation details, and results.

Good luck!!!