CPSC 314 Assignment 4: Introduction to Shaders

Out: Monday March 13, 2017
Due: Friday March 24, 2017

[24 marks total; worth 8%]

1. The objective of this coding question is to gain some hands-on experience with using texture mapping, vertex shaders, and fragment shaders. See the course web pages for the starting template.

   The code runs in the browser and can thus be run by simply opening `a4.html`. You will need to enable local file access, as you did for Assignment 1, and as described here: `https://threejs.org/docs/#Manual/Getting_Started/How_to_run_things_locally`.

   The template code comes with the following keybindings:
   `a,s`: rotates the scene left and right
   `space`: starts and stops the bunny animation
   `t`: toggles the bunny texture from texture B to texture D.

   You will be making changes to the javascript (`a4.js`), the vertex shader (`shaders/mesh_vs.glsl`), and the fragment shader (`shaders/mesh_fs.glsl`). After making edits, a page reload on your browser will then run your code again. Note that error messages will be displayed on the javascript console.

   Complete the following modifications to the template code in `a4.js`.

   (a) (2 points) Change the texture-mapped square, which represents the ground, to be twice as large, and at the same time it should be centered underneath the bunny. The ground plane geometry is defined in `buildPlane()`.

   (b) (2 points) Change the texture coordinates assigned to the four vertices such that the texture becomes much finer than it currently is, i.e., so that it repeats four times in each dimension, for a total of 16 times on the ground plane. The ground plane texture coordinates are defined in `buildPlane()`. It is drawn using texture map `A` and its parameters, and so to get it to repeat (instead of clamping), you will need to modify `loadTextures()`.

   (c) (2 points) Change the texture map that is used for the bunny from the UBC logo to something else. Note that it is using texture map `B` by default, and so you should be changing that. See the `loadTextures()` function for where the texture map file names are specified. Note that the dimensions of images used for texture mapping must by powers of 2, i.e., `256x256`, `512x512`, etc. If you like, use `psychedelic.png`. Or, better yet, take any image you like, and resize it using your favorite image resizing application. This is optional, but it can count as one of the "extras" you do for the last step.

(d) (4 points) Now you will be changing the cube. First, translate the cube so that it is sitting on the ground plane instead of penetrating it. Next, change the texture mapping for the faces of the cube. First, view the image `ubcTexture.png`, which is currently used to texture map the cube. Now change the texture coordinates for each of the cube faces (see `buildCube()`) in order to map the four individual sub-images in that texture map to individual faces. Do this in a way that makes sense, i.e., upright and legible. See the solution image at the start of this question for an example.

(e) (3 points) We'll now be making changes to the vertex shader. The goal in this step is to distort the bunny geometry over time. The time is known via `uniform float u_DistortionTime`, which is the animation time, in seconds. Also, another variable, `uniform float u_DistortionAmp`, will be used to control the desired distortion amplitude. This has a value of zero, except for the bunny, for which it has a value of 0.05. This will be a way to only apply distortions to the bunny.

Add the code below to the vertex shader, and use it to replace the original `newPosition` assignment. Run the code. Something as simple as a missing semicolon will result in a blank page. For effective debugging, make small changes to the code and look at the development console to understand the problems. With this code in place, hitting the spacebar should make the bunny head move up and down. Once this is working, change the code to do something else that is more interesting or impressive. The instructors and TAs will not respond to any questions that ask us to define this more precisely.

```
   float yoffset = 0.0;
   if (a_Position.y>0.14) {
      yoffset = u_DistortionAmp*0.2*(1.0+sin(3.0*u_DistortionTime));
   }
   vec4 newPosition = vec4(a_Position.x, a_Position.y + yoffset,
                           a_Position.z, 1.0);
}
```

(f) (3 points) Now take a look inside the fragment shader, (`mesh_fs.glsl`). The ultimate output of the fragment shader is `gl_FragColor`. Comment out the last line and add a new line that assigns `gl_FragColor = u_FragColor;`. Observe what this does, i.e., produce a simple flat-shaded rendering without textures. Now the goal for this step is to give your bunny a 'colour tinted' texture. Look for the code in `drawScene()` where `u_FragColor` is assigned before drawing, i.e.,
`gl.uniform4f(prog.uniforms.u_FragColor, ... );`
Change the colour assignments to something distinct and observe the resulting change in your 'flat shaded' rendered version.

Now compute the product of the default colour with the texture-map colour and use that as the colour for the fragment. Note that the shading language allows for component-wise multiplication using a statement such as: `vec4 c = a*b;`, where `a` and `b` are also of type `vec4`. You should now be able to produce a texture mapped bunny with a desired color tint.

(g) (3 points) We'll now use the fragment shader to produce an elliptical viewing window, which touches all four sides of your window and which colours all pixels outside of the ellipse black. In the supplied fragment shader, you are already given the NDCS coordinates of the fragment being rendered. Use this to evaluate an implicit function, `f`, for a circle, which will appear as an ellipse on screen because of the aspect ratio. Replacing the default `f=1.0;` line with your function should result in the fragment being assigned the colour black when it is outside the elliptical border.

(h) (5 points) Develop an idea of your own for augmenting the scene. You could add more bunnies, animate the colours, displace the geometry along the normals, displace texture coordinates as a function of time, etc. The instructors and TAs will not respond to any questions that ask us to define this more precisely.

Submit your code using `handin cs314 a4`.
Include a README.txt file that contains: (a) your name; (b) your student number; (c) any comments and explanations that you wish to include.