CISC 3620, Homework 7:
Applying bump maps to a cube

Prof Michael Mandel

1 Introduction

For this assignment, we will modify the bump mapping example shown in class. Instead of mapping the texture to a single square, we will map it to all sides of a cube in our cube viewer.

See this video for what the final cube should look like:
http://m.mr-pc.org/t/cisc3620/2019sp/hw7screencast.mp4

2 Fork my project

2.1. Go to https://jsfiddle.net/asterix77/gaxkw7p6/

2.2. Make sure you are logged in to the account you created for homework 1

2.3. Click on the "Fork" button to create your own copy of the fiddle

2.4. Click on the "Run" button to run it. You should see a diamond textured with a brick pattern that appears to be illuminated by a light from above that is moving in a circle in the $y = 5$ plane.

3 Convert normals and tangents from uniforms to attributes

3.1. Convert them in the vertex shader from `uniforms` to `attributes`.

3.2. Change their names in the vertex shader from `normal` and `objTangent` to `vNormal` and `vObjTangent` in both their definition and use.

3.3. In the JavaScript code, create arrays to hold normals and tangents.

3.4. Put six copies of the existing normal and tangent vectors in those arrays.

3.5. Send those array as attributes to the vertex shader.

4 Write function to create cube faces

The existing `makeCube()` function is just a placeholder that really makes one face ofo the cube. Rewrite it so that it will work for any cube face. To do so, you

will need to calculate the normal and tangent and add the relevant attributes to the arrays for position, texture coordinate, normal, and tangent. You can base your function on `makeCubeFace()` from previous homeworks, for example https://jsfiddle.net/asterix77/o27u6ma3/.

4.1. Compute the normal to the face. To do this, you should find the vectors from the first vertex to the second, call it `t1`, and from the second to the third, call it `t2`. The normal is then the normalized cross product of `t2` with `t1`. Use the `MV.js` functions `subtract()`, `cross()`, and `normalize()`.

4.2. Compute the tangent to the face. To do this, normalize the vector `t1`.

4.3. Add six vertices making two triangles to the `pointArray`.

4.4. Add the corresponding six points of the texture coordinates to the `texCoordsArray`.

4.5. Add six copies of the normal to the array holding the normal attributes.

4.6. Add six copies of the tangent to the array holding the tangents.

5 Create all cube faces

Now create all of the faces of the cube using the points in the `vertices` array and your `makeCubeFace()` function.

You probably want to reference the cube code from the last homework: https://jsfiddle.net/asterix77/o27u6ma3/.

5.1. Write down a representation of where each of the elements of the `vertices` array is in the cube to figure out which vertices make up each face.

5.2. Call `makeCubeFace()` once for each face. Make sure to define the vertices in the proper order, from the lower left, counter clockwise to the lower right, upper right, and upper left so that the normal points outwards according to the left hand rule. This will also make sure the texture coordinates correspond to the array `texCoordsBase`.

6 Debug your cube

- Make sure all of the cube faces are squares and not M shapes.
- Make sure all of the textures are mapped correctly so that they appear continuous on each square and not as disconnected triangles.
- Make sure all of the normals point outward from the cube so that the lighting appears to travel around the visible faces.

- Make sure the tangents are correct so that the highlights on the edges of the bricks point towards the face that is illuminated.

7 Extra credit extensions

For up to 3 percentage points of extra credit on your final course grade, submit up to three of the following extensions before the final exam. Each extension successfully implemented will be worth 1 percentage point.

- Place this scene in the cube viewer that we have been using for previous homeworks so that you can move the eye of the viewer around the scene. The eye should move relative to both the object and the light. The light should still move around on the same trajectory it is here.
- Apply this brick texture to the object that you created for homework 4, but without implementing bump mapping.
- Apply the bump mapping procedure using this brick texture AND bump map to the object that you created in homework 4.

8 Record your model and moving viewer

Record a video of your cube showing the light moving around through its full range of motion. Use a screen-cast program like Quicktime or CamStudio.

9 Submit it

9.1. Click on the "Save" button to save your fiddle.

9.2. Log in to Blackboard and open the dropbox for Homework 7.

9.3. Add the video of your fiddle to the submission

9.4. Paste the URL of your fiddle as part of the comment for your submission.