**Computer Graphics** Fall, 2014

# TP 5

*17 October*

The main goal of this assignment is for you to understand and implement the use of textures for bump mapping, and FBOs.

### Bump mapping

- Bump mapping will need two textures: a bump map, and a standard texture map, often called the diffuse map. The bump map will store the perturbed normals, while the diffuse map will contain the standard texture.
- As done in the Textures TP, read and store both the bump map and the diffuse map in `myTexture` variables in your `myObject3D` class.
- As done in the Textures TP, use `sampler2D` variables (set properly!) in your shaders to pass these texture maps to the shaders.
- create an array in the `myObject3D` class to store the tangent coordinates for each vertex.

  ```
  vector<GLfloat> tangents;
  ```

- I will give you the function `computeTangents()` for the `myObject3D` class which computes the tangent vector for each vertex as discussed in the class. Call that to compute the tangent for each vertex of your object.
- As done before many times, create object buffers to send these tangents to the shaders.
- In your fragment shader, compute the matrix to transform the light direction, and the eye direction from the world space to tangent space.

  ```
  vec3 n = normalize (normal_matrix * mynormal);
  vec3 t = normalize (normal_matrix * mytangent);
  vec3 b = normalize (cross(n,t));
  mat3 in_m = mat3(t,b,n);
  mat3 out_m = transpose(in_m);
  ```

  Use the matrix `out_m` to transform light and eye directions to tangent space.
- In your fragment shader, then read the perturbed normals from the normal texture map, and the diffuse coefficients from the diffuse texture map.

  ```
  normal = normalize (2.0 * texture2D(bumptex, mytexturecoords.st).rgb - 1.f);
  diffuse_color = texture2D(tex, mytexturecoords.st);
  ```

- Then use the shading as before (adding up ambient, diffuse and specular colors) with this normal vector and diffuse coefficient, as well as the transformed light and eye direction vectors.
- I have provided the bricks diffuse and normal map for trying out your code. You can (and should!) download many different bump maps from `FilterForge`.

**FBOs**

- Generate an id (which will be `GLuint`) for a FrameBufferObject:

```
glGenFramebuffers(1, &fboId);
```

- Generate and set up the texture to store the color values for that FrameBufferObject:

```
GLuint color_tex;
glGenTextures(1, &color_tex);
glBindTexture(GL_TEXTURE_2D, color_tex);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, Glut_w, Glut_h, 0, GL_RGB, GL_UNSIGNED_BYTE, 0);
```

- Generate and set up the texture to store the depth values for that Framebuffer:

```
GLuint depth_tex;
glGenTextures(1, &depth_tex);
glBindTexture(GL_TEXTURE_2D, depth_tex);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_DEPTH_TEXTURE_MODE, GL_INTENSITY);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_COMPARE_MODE, GL_COMPARE_R_TO_TEXTURE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_COMPARE_FUNC, GL_LEQUAL);
glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT24, Glut_w, Glut_h, 0,
GL_DEPTH_COMPONENT, GL_UNSIGNED_BYTE, NULL);
```

- Connect the color and depth textures to the FrameBufferObject (you will need to bind the FrameBufferObject before though):

```
// attach the color texture to FBO
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, color_tex, 0);

// attach the depth texture to FBO
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT, GL_TEXTURE_2D, depth_tex, 0);
```

- Now to use the FrameBufferObject to capture a snapshot of the scene, first bind it and then draw the scene you want to capture, and then unbind it.

```
glBindFramebuffer(GL_FRAMEBUFFER, fboId);
drawObjects();
glBindFramebuffer(GL_FRAMEBUFFER, 0);
```

- Now the scene is contained in the texture `color_tex`! You can use it in the shaders in whatever way you want.