Project #2 (fall, 2023)

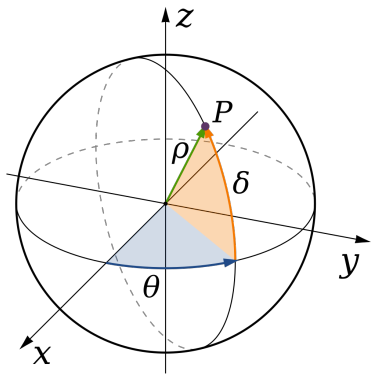
1 Objectives

- To implement an interactive camera system around the earth.
- To apply the Phong (or Blinn-Phong) shading to the earth with color, normal, and specular maps on the fragment shader.
- To implement displacement mapping on the vertex shader.

2 Requirements

- The size of the canvas is $1,024 \times 1,024$.
- Use the clear color $(0.1, 0.1, 0.1, 1.0)$.
- The only 3rd party library you can use is glMatrix. You will get ZERO point if you use any other 3rd party library. (Refer to the skeleton program for the usage of glMatrix.)
- If your program doesn't run at all (with errors) you get ZERO point.
- Do not submit the skeleton program just to get any partial point.

3 Interactive Satellite Camera System



- Draw two tracks as circles with their radii (ρ in the figure) 10 units.

  - One track (white circle in the video) denotes the "equator" and the camera moves along the equator as its **longitude** changes. (θ in the figure)
  - The other track (yellow circle in the video) denotes the path along which the satellite moves as its **latitude** changes. (σ in the figure) Note that this track should be transformed as the longitude changes.

- To show the satellite location (P in the figure), draw the "line of sight" (the pink line in the video.) from the satellite position to the origin. The distance to the camera from the origin should be 10 units.

- On the left half of the canvas, draw the whole scene (the earth, the satellite tracks, and the line-of-sight) seen from the fixed camera. (You do not need to draw the axes.)
  - The camera is located at $(30, 10, 30)$ in the world coordinates.
  - The camera is looking at the origin of the world coordinate system.
  - The camera is a perspective on with its fovy value 30 degrees.
  - You CAN use `mat4.setLookAt()`/`mat4.lookAt()` of the `glMatrix` library for this viewing transformation.

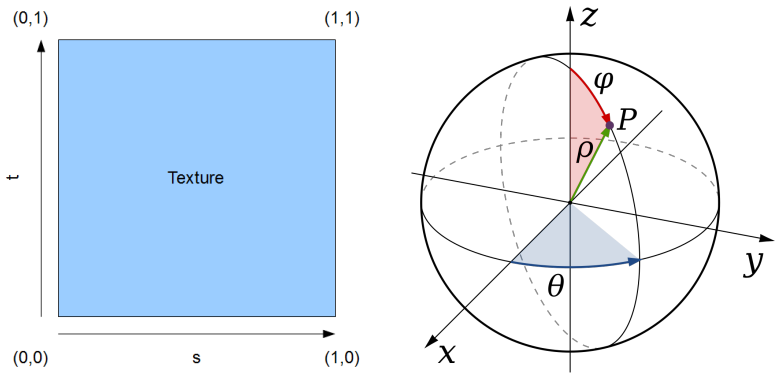
- On the right half of the canvas, draw the scene viewed from the satellite camera.
  - You should use a perspective camera with fovy value 30 degrees.
  - The distance from the camera to the origin should be 10 units.
  - When `longitude`=`latitude`= 0, the camera should be (in world coordinates)
    * located at $(0, 0, 10)$
    * looking along the direction $(0, 0, -1)$
    * with it "up direction" $(0, 1, 0)$.
  - You should **NOT** use `mat4.setLootAt()`/`mat4.lookAt()` function of the `glMatrix` library for this viewing transformation.

- The satellite camera should be controlled by the arrow keys and the slide bars. (The range of the longitude is $[0, 360]$ degrees and that of the latitude is $[-90, 90]$ degrees.)
  - Left/Right: Decrese/increse the longitude by 1 degree.
  - Down/Up: Decrese/increse the latitude by 1 degree.

  When the camera is moved by the arrow keys, the slide bars should be changed accordingly.

4 Earth with Lighting and Displacement Mapping

- Texture maps for the earth: color + bump + specular map [2]
- UI



(a) (b)

Figure 1: (a) Texture space [1] and (b) Spherical coordinates [4].

  - A checkbox to turn on/off the point light.
  - A checkbox to turn on/off the spot light.
  - A slide bar to change the cutoff angle of the spot light (in the range $[0, 45]$ degrees)
  - A slide bar to change the angle of rotation of the earth (in the range $[0, 360]$ degrees)
  - A slide bar to change the height of the earth (in the range $[0, 100]$ units)

- Displacement mapping: For the earth, apply the displacement mapping based on the bumpmap textures in the vertex shader. (Refer to the note below.)

- Lighting/shading
  - A point light is located at $(15, 35, 15)$. (in world coordinates)
  - A spot light is attached to the satellite with its direction to the center of the earth.
  - Per-fragment Blinn shading for the earth.
    * Use the colormap for diffusive and ambient properties.
    * Use the specularmap for the specular color.
    * Set the shininess appropriately.
    * Compute the normal vector correctly following the instructions below.

5 Notes

5.1 Earth as a Parametric Surface

Let $\mathbf{p}(s, t)$ is the parametric representation of the earth. Using the spherical coordinate system in Figure 1(b), $\theta(s, t)$ and $\phi(s, t)$ have simple forms as follows:

$$\theta(s, t) = 2\pi s, \text{ and } \phi(s, t) = \pi(1 - t).$$

(Note that t in Figure 1(a) has the opposite direction with ϕ in Figure 1(b).) Now $r(s,t)$ denotes the "perturbed height" which is defined by the "bump map" texture $h(s,t)$ as follows.

$$r(s,t) = r_0 + S \cdot h(s,t)$$

where r_0 is the radius of the earth, $h(s,t)$ is the height value stored in the bumpmap, and S is the (constant) scaling factor modified by the "height" slide bar. Summing up, as in Figure 1(b) [4], our earth can be defined as

$$\mathbf{p}(s,t) = \begin{bmatrix} r\cos\theta\sin\phi \\ r\sin\theta\sin\phi \\ r\cos\phi \end{bmatrix}.$$

Note that, in our case, $r(s,t)$, $\theta(s,t)$, and $\phi(s,t)$ are all functions of (s,t).

5.2 Tangent Planes on Earth/Moon

On the surface of the Earth, we can find two tangent vectors as

$$\frac{\partial \mathbf{p}}{\partial s} \text{ and } \frac{\partial \mathbf{p}}{\partial t}.$$

By the chain rule,

$$\frac{\partial \mathbf{p}}{\partial s} = \frac{\partial \mathbf{p}}{\partial \theta}\frac{\partial \theta}{\partial s} + \frac{\partial \mathbf{p}}{\partial \phi}\frac{\partial \phi}{\partial s} + \frac{\partial \mathbf{p}}{\partial r}\frac{\partial r}{\partial s}$$
$$\frac{\partial \mathbf{p}}{\partial t} = \frac{\partial \mathbf{p}}{\partial \theta}\frac{\partial \theta}{\partial t} + \frac{\partial \mathbf{p}}{\partial \phi}\frac{\partial \phi}{\partial t} + \frac{\partial \mathbf{p}}{\partial r}\frac{\partial r}{\partial t}.$$

Note that

$$\frac{\partial \mathbf{p}}{\partial \theta} = \begin{bmatrix} -r\sin\theta\sin\phi \\ r\cos\theta\sin\phi \\ 0 \end{bmatrix} \qquad \frac{\partial \mathbf{p}}{\partial \phi} = \begin{bmatrix} r\cos\theta\cos\phi \\ r\sin\theta\cos\phi \\ -r\sin\phi \end{bmatrix} \qquad \frac{\partial \mathbf{p}}{\partial r} = \begin{bmatrix} \cos\theta\sin\phi \\ \sin\theta\sin\phi \\ \cos\phi \end{bmatrix}$$
$$\frac{\partial \theta}{\partial s} = 2\pi \qquad \frac{\partial \phi}{\partial s} = 0 \qquad \frac{\partial r}{\partial s} = ?$$
$$\frac{\partial \theta}{\partial t} = 0 \qquad \frac{\partial \phi}{\partial t} = -\pi \qquad \frac{\partial r}{\partial t} = ?$$

The problem is that we do not have any analytic formula for $\partial r/\partial s$ and $\partial r/\partial t$ since we cannot differentiate $h(s,t)$. So we have to approximate them using, e.g., the two-point central difference formula, as

$$\frac{\partial r}{\partial s} \approx \frac{S(h(s+\delta,t) - h(s-\delta,t))}{2\delta}$$
$$\frac{\partial r}{\partial t} \approx \frac{S(h(s,t+\delta) - h(s,t-\delta))}{2\delta}$$

with small δ value.

Finally, we can compute the normal vector at $\mathbf{p}(s,t)$ using the cross product as [3]

$$\frac{\partial \mathbf{p}}{\partial s} \times \frac{\partial \mathbf{p}}{\partial t}.$$

Note that we need to transform the computed normal appropriately.

6 Hints

- The following example will be helpful for "multiple viewports" (`https://minho-kim-uos.github.io/webgl/src/multiple_viewports.html`)
- Refer to this example to loading multiple images asynchronously using `promise.all`.
- GLSL provides `cross` function to compute the cross product.
- You can compute the (perturbed) normal either in the vertex shader or the fragment shader. But if you compute it in the fragment shader, be sure to pass the normal vector from the vertex shader without any transformation. Since all the computation is done in the local coordinate system (of the Earth), you have to apply the normal matrix after computing the (local) normal vector.
- As usual, flip along the y-direction when uploading texture images so that the texture matches with the coordinate system in Figure 1(a).
- Refer to `create_mesh_sphere.js` for generating vertex attributes for a sphere. Note that you need to generate 'high resolution' spheres for the earth to apply displacement mapping effectively.

References

[1] Kevin Brothaler. *OpenGL ES 2 for Android: A Quick-Start Guide*. Pragmatic Bookshelf, 2013.

[2] James Hastings-Trew. Jht's planetary pixel emporium. `http://planetpixelemporium.com`.

[3] Nicholas M. Patrikalakis, Takashi Maekawa, and Wonjoon Cho. Shape interrogation for computer aided design and manufacturing. `http://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/node27.html`.

[4] Eric W. Weisstein. Spherical coordinates. from MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/SphericalCoordinates.html`.