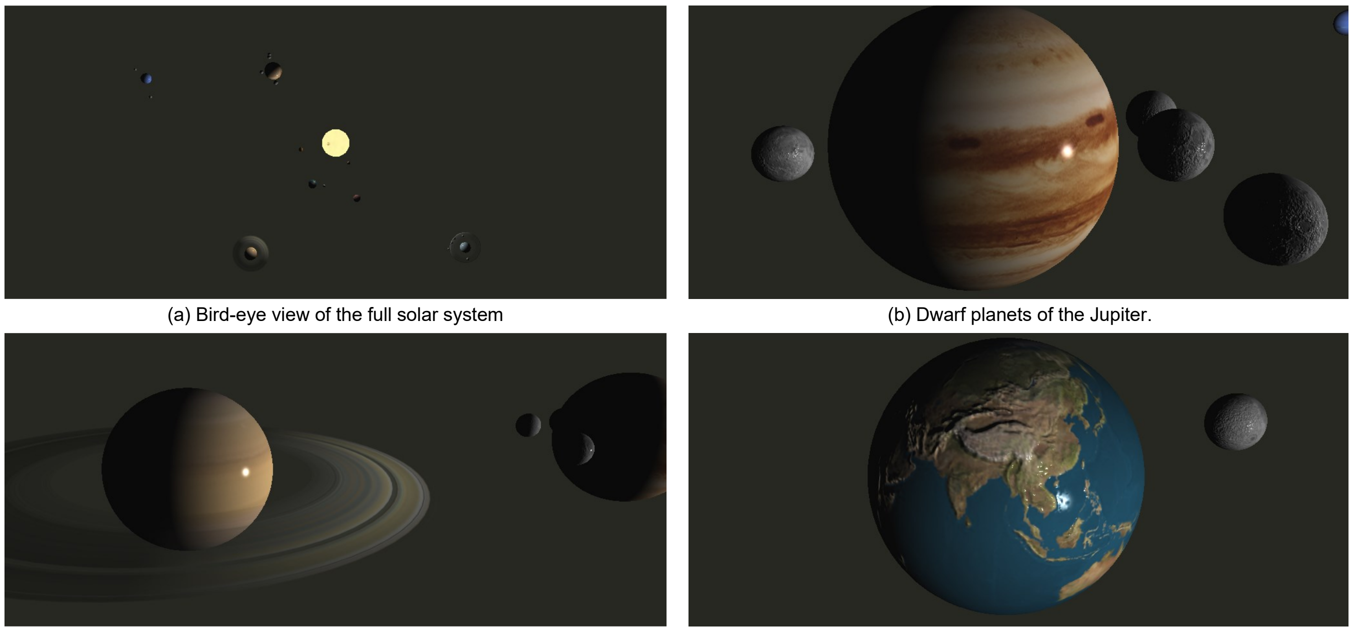
Computer Graphics

Instructor: Sungkil Lee
Assignment 4: Solar System



(a) Bird-eye view of the full solar system

(b) Dwarf planets of the Jupiter.

(c) The ring system of the Saturn

(d) Earth with additional bump mapping (optional)

Figure 1: Example renderings of a solar system.

First, refer to the general instruction for assignment submission, provided separately on the course web. If somethings do not meet the general requirements, you will lose corresponding points or the entire credits for this assignment.

1. Objective

In the last assignment, you learned how to place and transform 3D objects. This assignment is to complete your solar system with shading/textures and advanced features. The rendering results will be similar as shown in Figure 1. Refer to them for your implementation.

2. Requirements

- (20 pt) Make and animate dwarf planets of each planet.
- (20 pt) Apply a point light (the Sun) based per-pixel shading using Blinn-Phong model. The Sun should not need to be shaded (otherwise, -5 pt).
- (20 pt) Apply texturing using freely-available resources; e.g., http://planetpixelemporium.com/planets.html. Please *do not distribute the maps*, which violates copyrights.
- (20 pt) Make a textured ring system of the Saturn. The ring geometry needs to use an additional vertex array.
- (10 pt) Apply the alpha blending to the ring system. For the alpha blending, you need to configure OpenGL as:

```
glEnable( GL_BLEND );
glBlendFunc( GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA );
```

  Then, apply alpha from the alpha textures to `fragColor.a`.

3. Bump Mapping (Optional)

When you implement additional features listed below, you can get additional credits.

- (20 pt) Apply bump mapping on planet surfaces.

We know what the bump mapping implies, but its implementation is non-trivial. In particular for a correct implementation, we need to define *tangent* and *binormal* vectors (as additional vertex attributes) as well as normal vector in your vertices or meshes. For the moment, suppose the tangent and binormal vectors are already available. Then, we can find a bumped normal in the world space and the eye space in your fragment shader (see Fig. 2). When shading is performed in the eye space, make sure to use the eye-space normal instead of the world-space normal.

```
// NORM: normal map
// normal: world-space normal
vec3 tnormal = texture( NORM, tc ).xyz;
tnormal = normalize(tnormal-0.5); // tangent-space normal

// tbn: rotation matrix to world space from tangent space
mat3 tbn = mat3( tangent, binormal, normal );
vec3 wbnorm = tbn * tnormal; // world-space bumped normal
vec3 ebnorm = view_matrix * wbnorm; // eye-space bumped normal
```

Figure 2: An excerpt of fragment shader to get a bumped normal.

When we use a normal from the normal map, it is defined in a *local* tangent space. The basis vectors of the local tangent space are *tangent* (1,0,0), *binormal* (0,1,0), and *normal* (0,0,1). Hence, when there is no jittering in the normal map, the normal is simply defined as (0,0,1) and usually slight jittering is applied to (0,0,1).

What matters next is transforming the normal from the local tangent space normal to the world tangent space. Then, we need to know the world-space tangent and binormal. This is done by the matrix `tbn`, which is a rotation matrix for a change of coordinate systems; this can be inspired how we derive a view matrix.

However, we still do not know how to define the tangent and binormal vectors in the world space. In general, we may use texture coordinates to define tangent and binormal vectors, but this is too hard for beginners and the texture coordinates are often not available. Also, the standard vertex shader requires altering. Therefore, we instead use a tricky way as follows; refer to http://www.ozone3d.net/tutorials/mesh_deformer_p2.php for more details.

```
    vec3 c1 = cross(normal,vec3(0,0,1));
    vec3 c2 = cross(normal,vec3(0,1,0));
    vec3 tangent = normalize(length(c1)>length(c2)?c1:c2);
    vec3 binormal = cross(normal,tangent);
```

The tangent is defined by the direction of the normal vector, and the binormal is naturally derived from the cross product. This approximation works well in practice, and suffices for our purpose.

4. Additional Information

Read the followings for additional descriptions and hints.

Double-Sided Geometry of Ring Systems Unlike the sphere, the ring requires to be seen from both the sides. When you simply define single-sided surfaces, the other side is not seen. A precise solution is the definition of double-sided ring geometry, but we can simply handle this by just disabling *back-face culling*, as

```
    glDisable( GL_CULL_FACE );
```

This is equivalent to defining a double-sided geometry, but is much simpler. Once the ring is drawn, make sure to re-enable the back-face culling for other regular geometries.

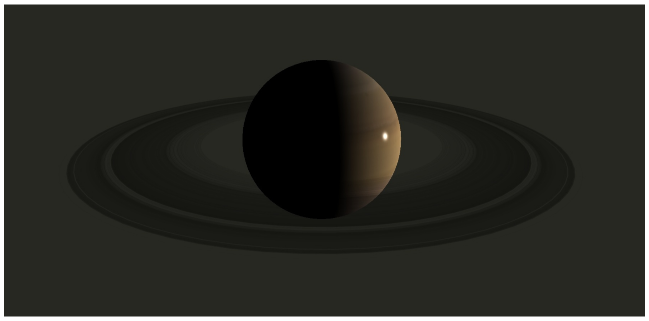


Figure 3: The ring of the Saturn shaded with precise normals.

Normal Vectors of Ring Systems When you precisely define the normal vectors of the ring systems, they can be orthogonal to the surfaces, being directed upward (or downward). Hence, the shading might be too dark, as shown in Figure 3. A simple trick to handle this is to suppose that the ring geometry is virtually a set of cylinders (with heights of zero) and their normals are coplanar with the ring plane. To this end, we do not change the geometry but normal vectors. For example, when the vertices of the ring is defined with respect to the origin in the local frame, their normal vectors can simply be unit vectors normalizing their positions (similarly to what we have done for the sphere). For example, Figure 1(c) is rendered with this trick.

5. What to Submit

- Source, project (or makefile), and executable files (90 pt). Executable files (e.g., *.exe) should also include shader files (and asset files, if necessary) in /bin/shaders/ directory. Also, please make sure to conform to the directory structure convention of distributed samples (e.g., /src/, /bin/, and /bin/shaders/).
- A PDF report file: yourid-yourname-a4.pdf (10 pt)
- Compress all the files into a single archive and rename it as yourid-yourname-a4.7z. (or yourid-yourname-a4.zip)
- Use i-campus to upload the file. You need to submit the file at the latest 23:59, the due date (see the course web page).