M. Stamminger, C. Weber *Erlangen,* ***Monday, 01.12.2014***

Computer Graphics - Programming Exercises

Assignment 7 [6+2 Points] (Space Shuttle & Farewell immediate mode)

This assignment focuses on the creation of geometry. We will abandon the old immediate mode, where primitives were created vertex after vertex. Instead we use `vertex array objects` (which use `vertex buffer objects` and `index buffer objects`), consult the documentation on how to use them.

The principle was explained in the lecture; you have a set of unique vertices (characterized with position, normal and tex coords) and a set of indices with which you construct the geometry.

That means you will need to update your existing drawing methods, instead of constructing a new sphere for every planet, you will create the geometry once, and reuse it whenever you need.

This assignment also adds the possibility to add geometry from external sources such as a file. You will add a giant space shuttle to the scene. The main part is to write a 3D object loader which is capable of loading an object from a .off file and computing normals for each vertex. An off-file stores an indexed face set of an object in ascii format. It has a very simple structure:

line 1: `OFF`
line 2: `v f e` (number of vertices, faces and edges)

followed by v lines which hold the xyz-components of a vertex. This is followed by f lines which hold the indices of a polygon. The first number of such a line is the number of vertices in the polygon. You can assume that this is always 3, so we are only dealing with triangle meshes. The remaining integers in the line are the indices of the vertices which form the respective triangle. For example the line
`3 0 2 1`
denotes a triangle which consists of the vertices on the poitions 0,2 und 1 in the vertex-list. The number of edges is not required and can be ignored (it is set to 0 anyway).

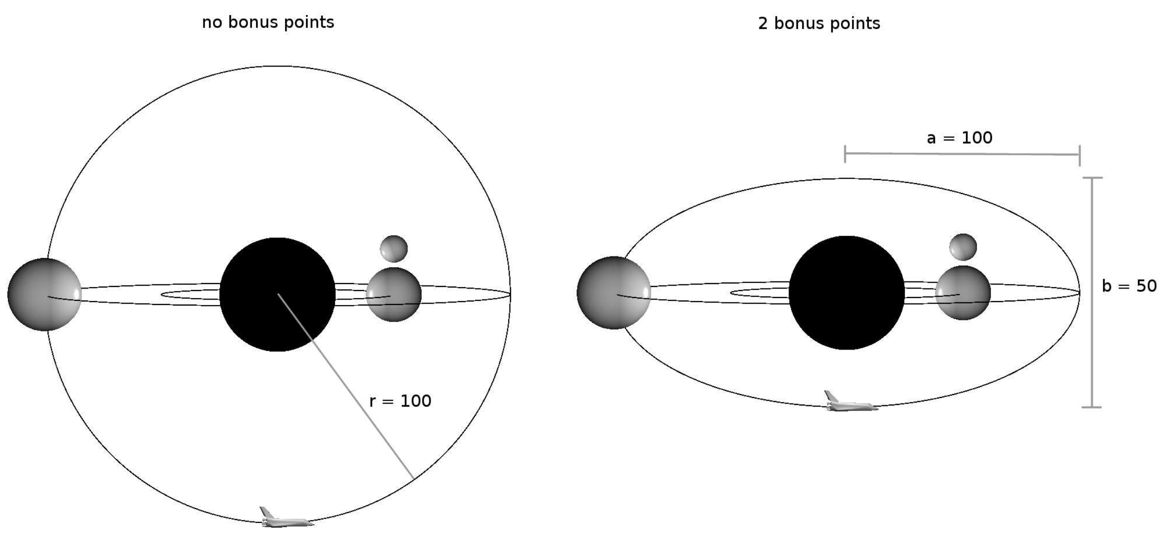
a) Since we abandon the immediate mode, you will have to update the geometry construction. Consult the creation of rings and the ”galaxy cube” on how to create unique vertices and how to create triangles (or lines or quads) using the indices. Finally create a sphere with positions,normals and texcoords. Since we have only one actual sphere, modifying the size of each planetary object should be done via its `model matrix`. Same goes for the rings. Take care that your scaling does not affect the rest of the system.
Drawing is done by binding the `vertex array object` and drawing the elements.

b) Complete the class `offObject` in the file `offLoader.h` which takes a filename in the constructor as parameter. A convenient way to load text files in c++ is to use the class `ifstream` which allows you to load data with the >> operator without needing to explicitly cast to float or integer. The constructor should also compute the normals of each vertex using Newell’s formula.

The vertex-, normal- and index-data is internally stored using the STL-class `vector` which allows for a dynamic adding of elements using the member-function `push_back`. Elements in the container can be simply accessed with the []-operator.
Drawing is done analogous to the planets.

d) In its initial position, the space shuttle is located in the center of the coordinate system facing to the positive y-axis. Edit your display-function so that the shuttle is flying on an orbit with radius 100 around the sun with the top of the shuttle always facing the sun. The orbit should be perpendicular the the orbits of the planets, however, it should have the same radius and speed as the saturn (Make sure that they don't collide).

BONUS You can collect two bonus points by solving the following tricky task:
Replace the circular orbit of the shuttle by an ellipsoidal orbit like it is done in the solution. The parameters of the ellipse can be seen in the subsequent figure. As a matter of course, the front of the shuttle should always be in the direction of flight, no matter which orbit you use.



Additional Information

- As always, we recommend you to implement the solution in your previous code. Alternatively, feel free to copy any necessary files such as the textures and the camera handler into the current working directory.
- If Newell's formula hasn't been discussed yet, here is a link:
  `http://www.opengl.org/wiki/Calculating_a_Surface_Normal`

Implementation Guidelines Please make sure that your code is free of any `glVertex`.

Good Luck!

Your source code will be copied from your handin directory on:
Monday, 08.12.2014 14:00 pm
all subsequent changes cannot be taken into account!