CS32310: A virtual orrery
Semester 1, Coursework (Assessed Assignment, 50%)

Date of release: 13th October 2014.
Date of submission: Friday 21st November 2014. (23.59 GMT)

Topic: Building a 3D visualisation of the solar system.

Aims and Objectives: The aim of the assignment is to gain familiarity with transformations, texture mapping and shading and simple animation in WebGL.

The objective is to write an application for displaying texture-mapped sphere's representing the Sun, planets and moons including orbital rotation.

1. Provided code and data

The code available at www.learningwebgl.com (particularly lesson 11) gives a useful starting point, such as setting up the scene, drawing a textured sphere and mouse interaction. A possible scenegraph implementation will be discussed in the lectures. You are provided with texture images for the sun and planets. Note that these images came from http://planetpixelemporium.com/ (with permission) and certain copyright restrictions apply.

2. Your Tasks

You are asked to create a WebGL program to display a simple model of orbiting planets. The scene should include (as a minimum) the Sun, Earth and Moon with the Earth orbiting the Sun and the Moon orbiting the Earth, in circular or more complex elliptical orbits. Each planet, moon etc. should also spin on its axis. The moons, planets and Sun should be texture mapped and lit by a single light source located at the centre of the Sun. The lighting should use a Phong shading model (with ambient, diffuse and specular terms). The finished program should look something like the image shown in Figure 1.

For the basic spinning and orbits you can combine a rotation (the spin), followed by a translation (the orbital distance) followed by a further rotation (the orbital rotation). It is recommended that a stack of matrices be used to keep track of the current matrix and allow retaining information about the previous state. A scenegraph structure can also be helpful in arranging the scene information and will be discussed in lectures.

3. Possible extensions

For top marks (above 80%) you should implement some non-trivial additional functionality. One option is to add the rings of Saturn. Using a texture mapped disk is possible, but looks terrible. Using a semi-transparent square can produce much nicer results. You can use the supplied image, *ringsrgba.png*, which includes transparency information in the alpha channel. You will need to enable blending to use the transparency information in the image and set the appropriate blending function. You will also need to make sure that the transparent quad is drawn last, in order to prevent problems with the hidden surface removal.

Another option is to allow elliptical orbits, the equations for this are as follows:

$$r = R(1+e)/(1+e*\cos(\theta))$$

$$\delta\theta = \delta t*R*R*a/(r*r)$$

$$\theta \mathrel{+}= \delta\theta$$

where (r, θ) is the current polar coordinate position of the planet in a planar orbit, with the sun at the origin of the system, and $\delta\theta$ is the angular change of the planet's position in a time δt. R is the orbital radius when $\theta = 0$ (0 degrees), a is the planet's angular velocity at that point, and e is the orbital eccentricity (zero for a circle). Assume that R, a and e are constant for each planet and for each satellite.

The formulae give a reasonable finite difference approximation to planetary motion, as described by Kepler's first two laws of orbital motion. It is assumed the sun's location is constant, and that the planets do not gravitationally disturb each other's orbits. The orbits of any planetary satellites are assumed to be influenced by the parent planet only.

Other possible extensions include the use of multi-texturing (e.g. to add clouds to the Earth) or bump mapping (e.g. for the moon).

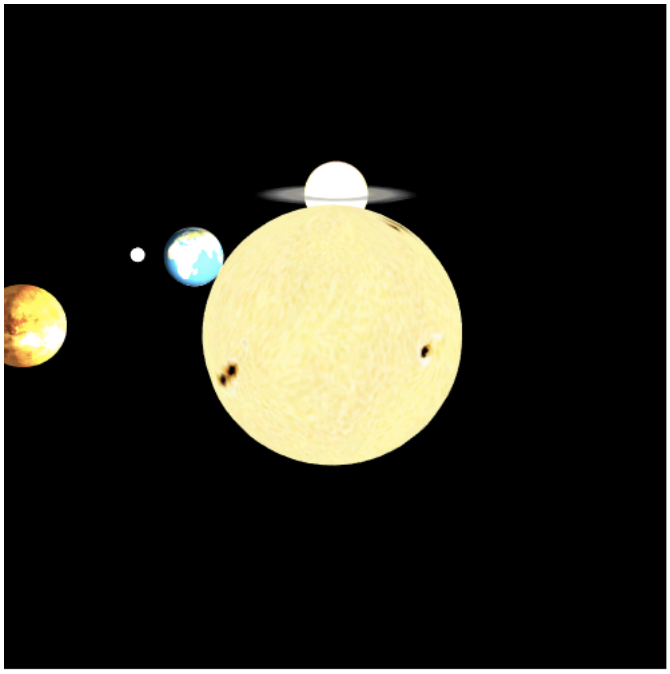

Figure 1. A screenshot of the finished application

4. Deliverables, marking scheme and handing in

You are required to deliver the following via Blackboard as a single zip file by Friday 21st November 2014, 23.59 GMT. You may also be asked to give a demonstration of your working solution to the examiners.
You should include the following in the zip file:

a) 20% of the marks are allocated for a brief report on your implementation, including a description of the program and its functionality and structure, any problems encountered and any extras implemented. This should be in pdf format. Note: it must be possible to mark your submission

based on the report only, therefore it must be an accurate and honest description of your submission. Also, no report means no grade.

b) 60% of the marks are allocated for functionality as specified in Part 2 above, as assessed from the information given in your report. The examiners will (most probably) also try to run your code in the Chrome browser in the Orchard lab to verify the claimed functionality and may examine your code. A rough break down of the marks is 15% for correctly drawing the spheres, 15% for correctly texture mapping the spheres, 15% for correctly lighting the spheres and 15% for correctly positioning and animating the spheres.

c) Up to 20% of the marks are allocated for any additional functionality implemented. Trivial extensions will only gain a small fraction of the possible 20%. This is an opportunity for first class students to "wow" the markers. Any extensions should be detailed in your report and clearly identified in your submitted code.

5 Useful resources

Learning webGL: http://learningwebgl.com
Chrome experiments: http://www.chromeexperiments.com/webgl/
WebGL specification: http://www.khronos.org/webgl/
OpenGL: http://www.opengl.org
GLMatrix library: http://glmatrix.net/docs/2.2.0/
WebGL reference: http://www.khronos.org/files/webgl/webgl-reference-card-1_0.pdf

Horst Holstein and Bernie Tiddeman
13th October 2014