COM3503: 3D Computer Graphics: Assignment (40%)

Dr. Steve Maddock

Deadline: 3pm, Tuesday 9 December

1. Introduction

The **aim** of the assignment is to test your ability to construct and control a scene graph (hierarchical relationship of pieces) and to implement this using OpenGL commands. A model will be built from simpler geometric objects combined using transformations and then controlled to produce animation

Start by *designing* your program. This should only take a few hours since you may re-use any program code from the exercise sheets used in lab classes and you will already have worked through most of this code in order to understand it. Write down all the OpenGL commands that you will use and how you will use them. Then implement the program. Don't try to implement the entire program in one sitting. Do it in stages. And keep the interface simple.

2. The task

In lectures we looked at an example of a hierarchical object at The Guerrilla CG Project (see Figure 1 and http://www.youtube.com/user/GuerrillaCG/videos). In this assignment you will build and animate something similar.

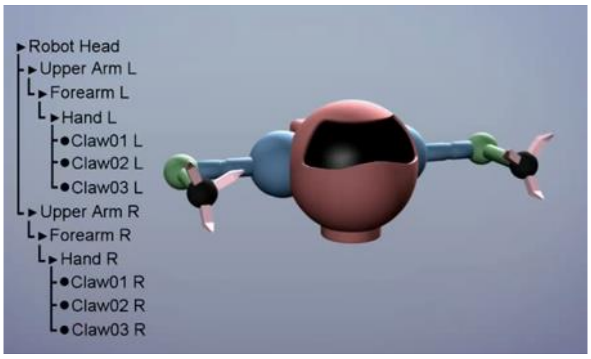

Figure 1: Building a hierarchical robot (The Guerrilla CG Project: http://www.youtube.com/user/GuerrillaCG/videos)

You must satisfy all the following requirements:

- You must design and build a flying robot made out of at least 8 pieces arranged in 4 hierarchical layers, e.g. head (including eye (s)), upper arms, lower arms, hands. Each of the pieces can be made out of simple geometric objects, such as spheres, cylinders and cuboids, as with the example in Figure 1. For example, the elaborate robot head in Figure 1 can instead be made from a single sphere with either a single sphere or two spheres as an eye or eyes, respectively, protruding slightly at the front of the head. The hands could also be simpler than those shown in Figure 1.
  It is perfectly acceptable to use the standard OpenGL objects to model the robot parts (however, see later note on texture mapping). If you wish, you can use pieces that you have created in a separate modelling package, although then you would need to write code to load the data. But it must be pieces, so that they can be joined in a hierarchy. However, the hierarchy and associated transformations are more important than the quality of the pieces in the hierarchy. I want you to demonstrate that you understand transformations and a scene graph hierarchy. (*Hint*: You need to be thinking about object-oriented programming when you develop your model, so that multiple flying robots could be easily produced.)
- Either (i) model the eyes (or single eye) as spheres with spotlights shining out of them in the direction of travel, or (ii) add a lamp (with spotlight) to the top of the robot's head so that it shines in the direction of travel. There must be an option in the interface to turn the spotlight(s) on and off. (*Hint*: A spotlight should look like a spotlight, as well as behave like a spotlight. So, consider emission properties for the object that is modelling the part where the spotlight is shining from.)
- You may choose whatever colours you wish for parts of the flying robot. (To improve the quality of the look of your model, you may consider using texture mapping. However, the standard glut objects do not include texture coordinates, so they cannot be texture mapped. For this project it will suffice to use the standard ambient, diffuse, specular, and emission properties to change the colour of parts of the flying robot.)
- The robot should be placed inside a large room. Figure 2 illustrates this and shows the robot with one eye which has a spotlight shining from it in the direction of travel. (*Note*: the relative size of the robot with respect to the room is not necessarily to scale.)
- The floor, the four walls and the ceiling of the room must be texture-mapped with appropriate textures, e.g. the floor could be texture-mapped with a wood pattern. (*Hint*: A set of triangles organised to make a plane – see exercise sheets – could be used to model the floor, walls and ceiling, with appropriate transformations, and a separate room class could be created for collecting together the walls and floor of a room.)
- The room should be illuminated with some general world lights, which it should be possible to turn on and off (or dim, i.e. reduce the intensity) from the interface. When turned (or dimmed) the effect of the robot's spotlight(s) will be much more dramatic, like someone shining a torch in a darkened room.

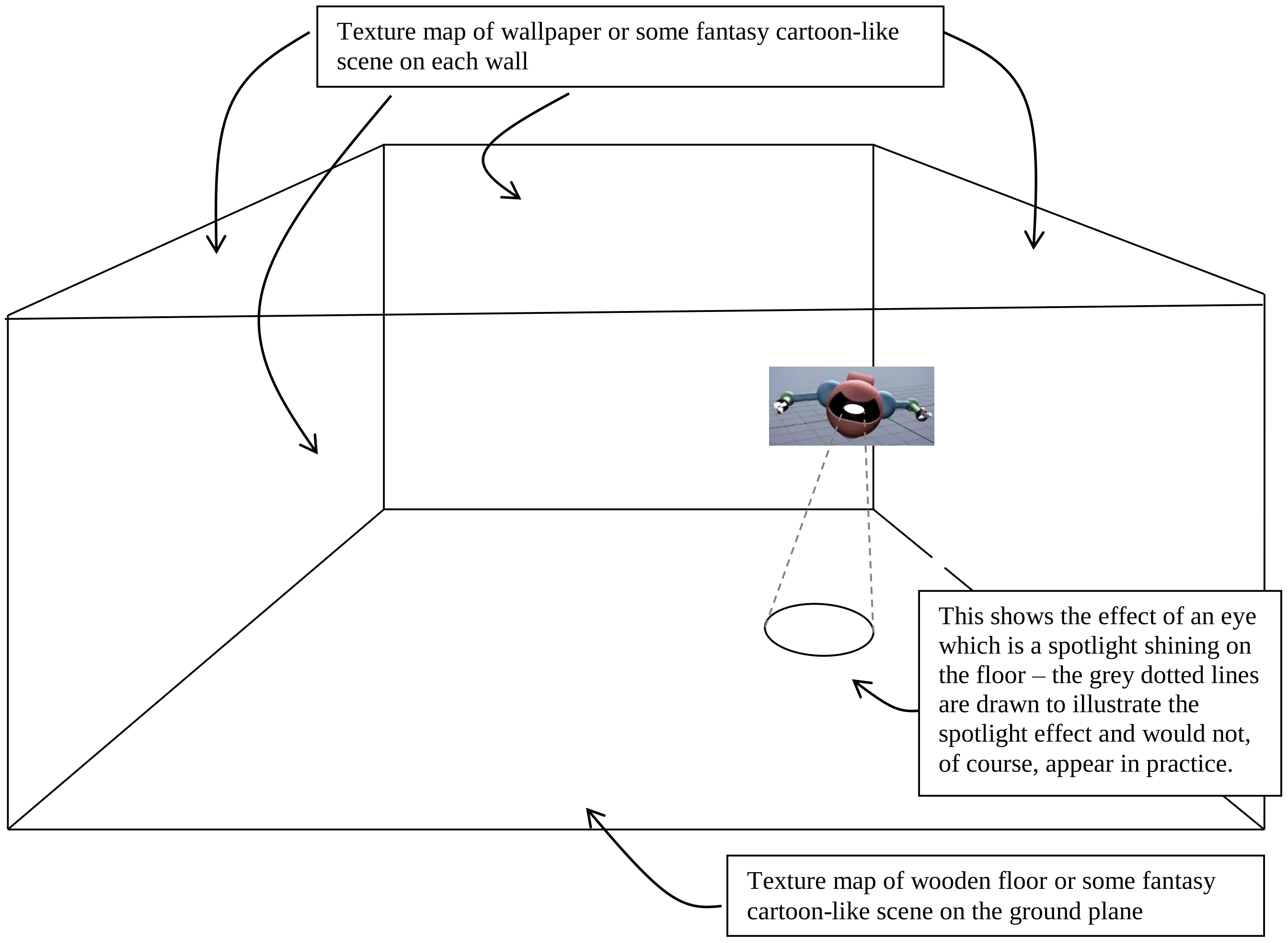
Texture map of wallpaper or some fantasy cartoon-like scene on each wall
This shows the effect of an eye which is a spotlight shining on the floor – the grey dotted lines are drawn to illustrate the spotlight effect and would not, of course, appear in practice.
Texture map of wooden floor or some fantasy cartoon-like scene on the ground plane

Figure 2: The room containing the flying robot

- The ceiling of the room should have two spotlights pointing down from it. These will create pools of light in specific areas of the room.
- The robot should fly around the room. This could be done by setting a random point in the room for the robot to fly to, then another random point, and so on. Alternatively, a sequence of points (e.g. a circular-like path) stored in a simple data structure could be used. This requires you to design the program code necessary to control this.
- Add three obstacles in the room that the robot must avoid whilst flying around – these can be modelled using simple geometric objects and can be floating objects or tall objects standing on the floor.
- The animation should look plausible. Thus, whilst the robot is flying it should be animating its parts and its orientation with respect to the ground. This requires you to design the program code necessary to control this. Don't worry about detecting self-collisions, e.g. the arm penetrating the body – for the purposes of this assignment, self-collisions do not matter.
- At the start of the animation sequence the robot should face the camera, then wave to the camera and then fly around the room. The waving operation can be achieved using interpolation of the Euler angles between the hierarchical pieces – do not consider quaternions, as this is beyond the scope of this project; it is perfectly acceptable to alter the Euler angles to achieve a simple wave of an arm.
- Use simple interface controls for the animation, e.g. buttons to stop, start and reset the animation. A reset button should reposition the robot at the start of the animation sequence, i.e. facing the camera before the wave to the camera.
- The camera should be positioned so as to point towards the centre of the room (which may be the world origin). The user should be able to use the mouse to rotate the camera around the scene. It doesn't matter if the camera remains within the room or not, as long as the scene is visible.
- You do NOT have to do shadows. Do not worry about shadow effects.
- Do **ONE** of the following advanced effects:
  - Model two robots, with one robot chasing the other.
  - Add the ability to ride on the robot looking in its direction of travel.

3. Deliverables

- You should submit a zip file containing a copy of your program code (and any other necessary resources, e.g.

image files for the textures and a readme.txt file that describes everything) via MOLE – this can be done via the link to the assignment handout. You should submit whatever you have done, even if you have not completed all the requirements. If you submit nothing, you cannot receive any marks. ***The program MUST compile and run from the command line*** on a standard PC, such as the ones in the Lewin Lab. I will not install 'YetAnotherIDE' to make your program work.

- You must include appropriate comments in your program to identify that you wrote the code, e.g.

  ```
  /* I declare that this code is my own work */
  /* Author <insert your name here> <insert your email address here> */
  ```

  If your program is based on code that I gave you, you must state that as part of these comments, and identify which bits you added.
- The body of the MOLE submission message should state that the work you have handed in is your own. It should also state which of the advanced options outlined in the deliverables above that you have undertaken.
- You must include a readme.txt file as part of your submission which describes how to run the program. It would also be useful to include a .bat file to automatically compile and/or run the program. The readme.txt file should also state which of the advanced options you have undertaken.
- Optional: You might like to make a short video of your animation using software such as Fraps (www.fraps.com). If you do so, **DO NOT** include this in the handin as it will be too big for MOLE to handle – we tried using MOLE for this in the past and it crashed the system!!. Instead, put the animation on youtube or your personal website and give the URL of the animation in your readme.txt file. (Indeed, if you are thinking of a career in the graphics industry, then you should be adding such animation pieces to your personal website (your digital portfolio) to show off what you are capable of.)

4. Marking

I will check that the program meets the requirements listed above. To make sure you get some marks, the program **must** compile and do some part of the work requested even if it is not complete, e.g. you might produce a model but not be able to animate it. Your program code will be run and exercised thoroughly.

Marks will be available for:

- The quality of the programming (20%)
- Satisfying the requirements (80%)

In assessing the quality of your program code, I will be looking for well-laid out, elegant, object-oriented program code. As examples: consider using variables rather than literals to make a program more flexible; Use separate classes for the robot and the room and for the animation control. You should make good use of data structures and OpenGL commands.

In considering the requirements, 5 aspects will be considered:

- Modelling the robot, the wall and the room. The robot must be a hierarchical model. (*Hint*: consider drawing a scene graph for the robot model and another for the full scene before starting to program, as this will help your to structure your use of glPushMatrix and glPopMatrix.) Consider the use of high-resolution models so as to give better spotlight effects and the effect of this on rendering speed (*Hint*: consider display lists). The *quality* of the relevant models will be considered.
- Texturing. Have the relevant objects in the scene been textured? The *quality* of the texturing will be considered, e.g. seams between textures
- Lighting and interface controls. Any spotlights should behave correctly such that their effect is seen on the scene. Also, make sure you include general lights and the necessary interface controls, as described in the above specification.
- Animation: Is the animation smooth? Does it look plausible? The *quality* of the animation will be considered.
- The advanced effect: this will be mainly about whether or not you have achieved the advanced effect. Nonetheless, the quality of the advanced effect will be considered too.

5. Some tips on developing your solution in stages

It is possible for you to start work on the assignment immediately. For example, you could draw the scene graphs for the robot and for the full scene. You could build a draft robot model using the GLUT objects. Then, you could look at animation control of the robot.

After the exercise sheet on lighting, you could add a spotlight into the scene. When relevant material on creating meshes for different objects (e.g. a plane represented as a mesh) has been presented, you can build the room and decide whether or not to replace the GLUT pieces with more complex pieces for the robot. Finally, after the exercise sheet on texture mapping, you could add texture mapping to your solution. And so on…

This is only one possible development scenario. You must find the right process for you.