Computer Graphics Coursework

1)

a) *Describe the difference between attribute, uniform and varying in shader programming.* **[6 marks]**

Attributes are inputs for the vertex shader, for example the vertex coordinates. They are dynamic and can be changed. Uniform variables are static inputs that are passed to both the fragment and vertex shaders and can be used for scene variables such as the light direction. Varying denotes a parameter that is set in the vertex shader and passed to the fragment shader, for example the calculated normal of the vertex.

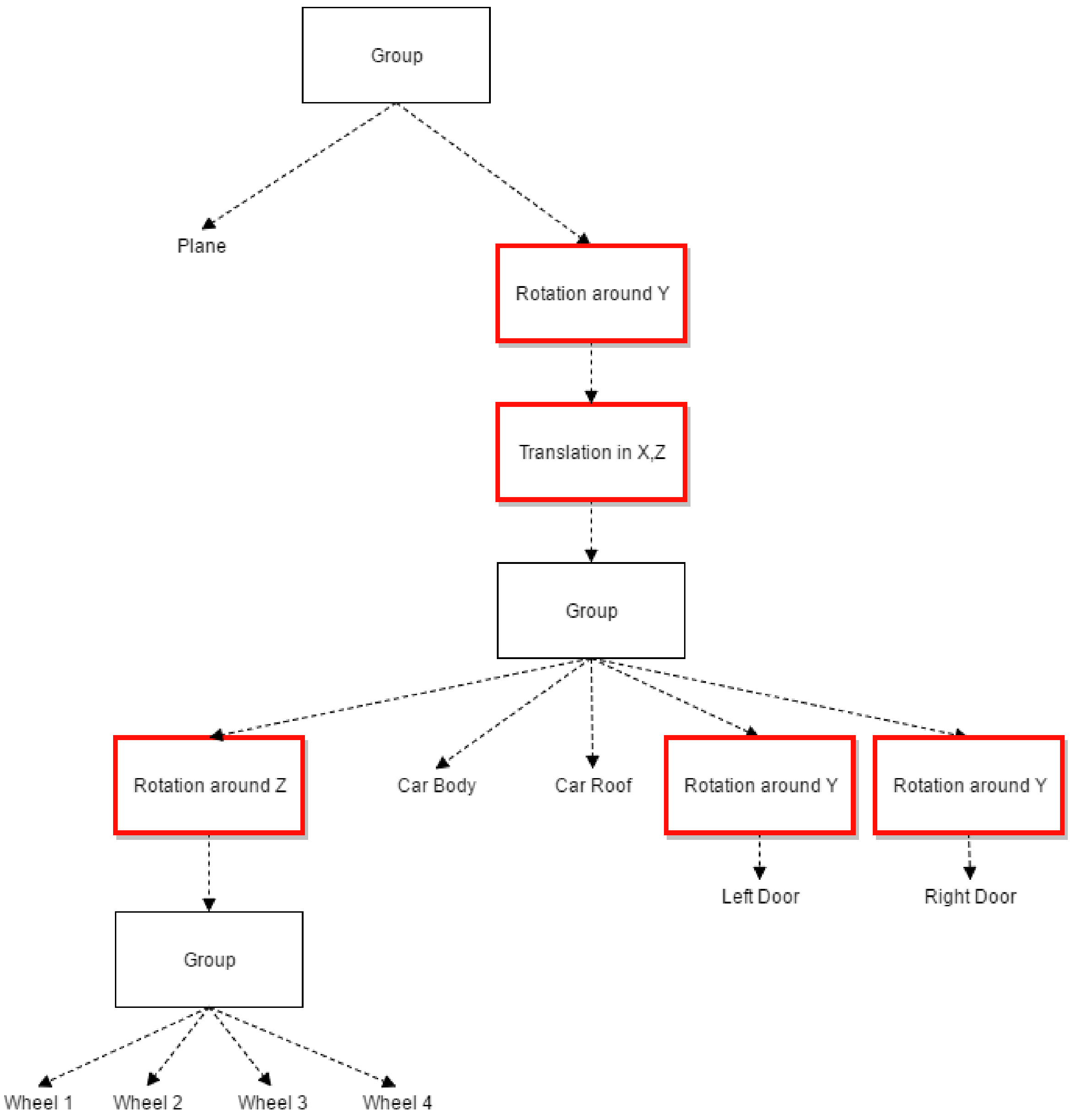
b) *Suppose you want to use a single array in the main() function of a WebGL programme, storing both the (x,y,z)-coordinates and the (r,g,b) colour information for every vertex of a polygon model. Show a programming statement for constructing such an array. Assume the polygon model contains 6 vertices.*

Also write down the programming statements for constructing the corresponding vertex buffer objects and assigning the array data to become the position and colour attributes of the vertex shader. **[10 marks]**

```
// Write our six vertices to an array (x,y,z)
var vertices = new Float32Array([
      5.0, 8.0, 0.0,
      3.0, 1.0, 0.0,
      5.0, 1.0, 0.0,
      5.0, 7.0, 0.0
      1.0, 1.0, 0.0,
      1.0, 1.0, 1.0
]);
// Create the vertex buffer and write to it
var vertexBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer);
gl.bufferData(gl.ARRAY_BUFFER, vertices, gl.STATIC_DRAW);
// Get attribute location
var vertexAttributeLoc = gl.getAttribLocation(gl.program,
"a_Position");
gl.vertexAttribPointer(vertexAttributeLoc, 3, gl.FLOAT, false,
0, 0);
// Assign to attribute var
gl.enableVertexAttribArray(vertexAttributeLoc);
// Unbind buffer
gl.bindBuffer(gl.ARRAY_BUFFER, null);

// Do the same for colors (r,g,b)
var colors = new Float32Array([
      1,0,0,
      1,0,0,
      1,0,0,
      0,0,1,
      0,0,1,
      0,0,1
]);
var colorBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, colorBuffer);
gl.bufferData(gl.ARRAY_BUFFER, colors, gl.STATIC_DRAW);
var colorAttributeLoc = gl.getAttribLocation(gl.program,
"a_Colors");
gl.vertexAttribPointer(colorAttributeLoc, 3, gl.FLOAT, false,
0, 0);
gl.enableVertexAttribArray(colorAttributeLoc);
gl.bindBuffer(gl.ARRAY_BUFFER, null);
```

c) *Draw the scene graph of the 3D car model as described in question 2.* **[6 marks]**



d) *Suppose drawBox(m) is a function to draw a transformed box according to the transformation matrix m. That is, if m is a rotation matrix, the function will draw a rotated box.*

i) *Explain the meaning of the following code segment and state the result obtained:* **[5 marks]**

```
m.setRotate(angle, 0.0, 1.0, 0.0);
m.translate(1.0, 3.0, -5.0);
drawBox(m);
```

The code rotates and translates a box – However since webGL statements are applied backwards, the box is first translated to the point (1,3,-5), and then the rotation is set to angle degrees around the Y axis. The rotation is performed around the point (0,0,0) and overwrites any prior rotation.

ii) *Explain whether you will get the same result if m.setRotate() has been replaced by m.rotate().* **[3 marks]**

No – If the box had a previous rotation applied to it, m.rotate compounds on that translation. For example, you could apply m.rotate(30,0,0,1) two times to achieve a 60 degree rotation in the z axis. However, applying m.setRotate multiple times will achieve no result as the rotation is set to a fixed value every time and does not rely on any previous translation.

2) *Using WebGL, implement a program to render a 3D car model and its movement. The car should comprise a body, two doors and four wheels.* **[70 Marks]**

The files are located in the folder "Car". Run the index.html file to view the animation. Press "l" to toggle the lighting, "c" to toggle the camera angle, the arrow keys to move and the space bar to toggle the opening and closing of the car doors.