COMPUTER SCIENCE TRIPOS Part IA – 2026 – Paper 3

3 Introduction to Graphics (rkm38)

(*a*) Explain the purpose of near- and far-clipping planes in Z-buffer algorithm and how their choice can increase or reduce the possibility of Z-fighting artifacts. [5 marks]

(*b*) Given the chromaticity coordinates x and y and luminance Y, give the equations for trichromatic coordinates X and Z. [5 marks]

(*c*) You used the latest Large Language Model to generate shader code using the following prompt:

"Generate both vertex and fragment shaders in GLSL. The shaders should transform vertices and normals using the model matrix M, the view matrix V, and the projection matrix P. Each pixel is shaded using the Phong reflection model, assuming a single point light source at fixed coordinates."

However, the resulting code does not work as expected. Fix all 4 bugs in the code, rewritting affected lines of code and explaining the nature of the bug. You may need to modify more than 4 lines. You can use line numbers and the name of the shader (vs/fs) to refer to the lines, e.g., `vs:3` for the vertex shader, line 3. The generated code does not contain any syntax errors. [10 marks]

```
1: #version 330 core

2: in vec3 aPos;
3: in vec3 aNormal;

4: uniform mat4 uModel, uView, uProj; // M, V, P
5: out vec3 vNormal, vFragPos;

6: void main()
7: {
8:    vec4 worldPos = uView * uModel * vec4(aPos, 1.0);

9:    vNormal = normalize((uModel *
                            vec4(aNormal, 1.0)).xyz);
10:   vFragPos = worldPos.xyz;

11:   gl_Position = uProj * uView * worldPos;
12: }
```

```
1: #version 330 core
// Normal and position in world space
2: in vec3 N, vFragPos;
3: out vec4 FragColor;

// light and camera in world space
4: uniform vec3 uLightPos, uCameraPos;

5: uniform vec3 uAmbientColor, uDiffuseColor,
                uSpecularColor;
6: uniform float uShininess;

7: void main()
8: {
9:    vec3 L = normalize(uLightPos - vFragPos);
10:   vec3 V = normalize(uCameraPos - vFragPos);
11:   vec3 R = reflect(-L, N);

12:   float diff = dot(N, L);
13:   float spec = pow(dot(R, V), uShininess);

14:   vec3 ambient  = uAmbientColor;
15:   vec3 diffuse  = uDiffuseColor  * diff;
16:   vec3 specular = uSpecularColor * spec;

17:   FragColor = vec4(ambient + diffuse +
                       specular, 1.0);
18:}
```