Introduction to Graphics
Supervision 1

Andrej Ivašković (a1294)
Compiled on: 20th October 2018

You may email me your work or leave it in my pigeonhole in the Trinity College Great Court mail room.
Please submit the assigned work at least 24 hours before the supervision!

1 Before attempting the problems

This exercise sheet only contains theoretical and conceptual exercises – writing OpenGL and GLSL code is covered in detail in the ticks.

Lectures 1–4 are covered here: theory of vision, the basic ray tracing algorithm, as well as an overview of the rendering pipeline and different coordinate systems used in computer graphics. Note how 4-vectors and matrix multiplication are ubiquitous here: if you are uncomfortable with these ideas, it might be a good idea to review them.

To get a feel of the work done by 3d artists, download Blender. Its user interface is not the most intuitive, but you will see that you will see that 3d artists use the same language and ideas as OpenGL programmers and computer scientists. Try importing or exporting some .obj files, then open them in a text editor. You should be able to see how 3d objects are represented in a format that is easy to parse both by humans and by programs.

2 Problems

1. Using blue lines on a black background in a CAD system would be a poor choice for the interface colours for designing an object. Why?

2. Calculate the ultimate monitor resolution (in a unit that makes sense) beyond which better resolution will be unnecessary. You should clearly state your assumptions. [Hint: you may find it useful to refer to the notes to get the relevant numbers that describe human vision]

3. Ray tracing and the rendering pipeline (somewhat resembling the OpenGL pipeline) are two methods for creating an image based on a scene. In which scenarios would you use one over the other?

4. (a) Explain the concepts of diffuse and specular reflection.
(b) Why do we care about approximating the specular component as opposed to performing the exact computation?

If the intensity of the incoming ray is \( I_{\text{in}} \), the proportion of light specularly reflected is \( k_{\text{spec}} \), the unit vector of the normal is \( \vec{n} \), the unit vector of the incoming ray in \( \vec{r}_{\text{in}} \), the unit vector of the direction of perfect reflection is \( \vec{r}_{\text{perf}} \), and the unit vector pointing towards the viewer is \( \vec{v} \), what is the formula for computing \( I \), the intensity of the specular component of reflected light in the direction of the viewer?

If \( n \) is Phong’s roughness coefficient, what is Phong’s approximation of specular reflection? Show that these two expressions are equal when \( n \to \infty \).

(c) Typically, surfaces have different types of components and their reflectance behaviour may vary in different parts – this is specified with reflectance maps. Reflectance maps contain the values of \( s \) and \( n \) in the following expression for \( \phi(i, e, g) \), for different parts of the surface:

\[
\phi(i, e, g) = \frac{s(n + 1)(2 \cos i \cos e - \cos g)^n}{2} + (1 - s) \cos i
\]

Here \( \phi(i, e, g) \) specifies the fraction of incident light reflected per unit surface area, per unit solid angle, in the direction of the camera. The angle \( i \) is the angle of the light source, relative to the surface normal \( \vec{n} \), \( e \) is the angle of a ray of light re-emitted from the surface, also relative to \( \vec{n} \), and \( g \) is the angle between the emitted ray the light source.

Discuss this formula (in particular, how it combines specular and diffuse reflection).

5. (a) Describe in your own words what aliasing is (in the context of computer graphics).

(b) Explain three methods of anti-aliasing in ray tracing. Describe their strengths and weaknesses.

(c) Is there ever need to perform anti-aliasing in the rendering pipeline?

6. What is distributed ray tracing?

7. Describe the steps of the rendering pipeline. In each stage explain why it is performed and what kind of coordinates are used in which step.

8. Explain why matrix operations are ubiquitous in 3d graphics rendering. If everything is happening in three dimensions, why don’t we stick to 3-vectors and \( 3 \times 3 \) matrices?

9. (2010 Paper 4 Question 4, (b)) Give a matrix that transforms the square \( ABCD \) into \( A'B'C'D' \) (see figure).

10. Given a finite set of weighted points \( S = \{(A_1, m_1), \ldots, (A_n, m_n)\} \), where the masses are real numbers such that \( \sum_{i=1}^{n} m_i \neq 0 \), the centre of mass of \( S \)
is the unique point $C$ such that, for all points $O$ in space:

$$\overrightarrow{OC} = \frac{\sum_{i=1}^{n} m_i \overrightarrow{OA_i}}{\sum_{i=1}^{n} m_i}$$

Given a triangle $\triangle ABC$ and a point $P$ in its plane, its barycentric coordinates relative to $\triangle ABC$ is the tuple $(m_A, m_B, m_C)$ such that $m_A + m_B + m_C = 1$ and $P$ is the centre of mass of $\{(A, m_A), (B, m_B), (C, m_C)\}$.

(a) How can you compute the barycentric coordinates of a point inside a triangle?

(b) How can these coordinates be used to implement Gouraud and Phong shading?

Note that the uniqueness of barycentric coordinates and the centre of mass can be proved with a bit of vector manipulation – do so if you want to review vector operations.

Some of the exercises have been borrowed or adapted from question in old Computer Graphics and Image Processing exercise sheets. Credit for those is due to their authors.