Resit Paper: Computer Graphics (INBCG-08) 2016

Date: May 17, 2016 Time: 5:30pm – 8:30pm

Instructions:

- Fill in your name, student number, and date (top right corner) on each of the answer sheets that you hand in.
- Write clearly and in English. Illegible parts will not be graded.
- In derivations, describe *all* the steps needed to arrive at your result.
- Use pseudo-code and sketches/diagrams where appropriate.
- It is recommended that you read all questions in full first.
- The use of books, lecture/lab slides, notes, other similar material, and electronic devices is *not allowed* during the exam.
- Zero tolerance cheating policy applies.

Structure:

- This exam paper consists of 9 questions on 3 pages.
- You have 3 hours to answer all the questions.
- Some questions are split into sub-questions, each with its own point maximum (shown in bold).
- The total number of available points P_{max} is 100.
- The exam grade E is arrived at as follows: $E = P/10$, where P is your exam score.
- Your final course grade depends also on your grades for practical assignments.

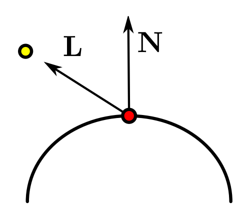
Question 1: Illumination

Local illumination models typically determine the final intensity (colour) based on several components.

(a) (**2 points**) What does the ambient component model from the real world?

(b) (**3 points**) What does the diffuse component model from the real world and why is it independent from the viewer/camera position?

(c) (**4 points**) Given the normal $\mathbf{N}$ and the direction $\mathbf{L}$ to a point light source (both normalised) at a point on a surface, give the formula for evaluating the diffusion component $\mathbf{I}_d$ in the Phong model, using the included sketch. How does your answer account for points with normals facing away from the light source?



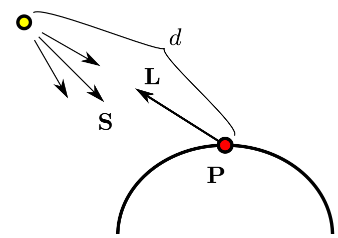
Question 2: Light sources

One of the simplest light sources in computer graphics is the point light source. A spot light source with attenuation improves on the point light source in several regards.

(a) (**2 points**) Describe what a point light source is, what parameters are needed to describe it in a scene, and what it (approximately) models from the real world.

(b) (**3 points**) Describe what a spot light source with attenuation is and what it (approximately) models from the real world. Note that there are two types of attenuation. [Hint: One is based on distances, the other on angles.]

(c) (**5 points**) Assume that at a point $\mathbf{P}$, the computed colour according to a certain illumination model with a single point light source is $\mathbf{I}_{point} = \mathbf{I}_a + \mathbf{I}_d$, where $\mathbf{I}_a$ is the ambient component and $\mathbf{I}_d$ is the diffuse component (the specular component is ignored). The light source is now replaced with a spot light source with attenuation with the same intensity and position, and its unit direction vector is $\mathbf{S}$ and its distance to $\mathbf{P}$ is d. The unit vector from $\mathbf{P}$ to the light is $\mathbf{L}$. Starting from $\mathbf{I}_{point}$, give a formula for computing $\mathbf{I}_{spot}$ at $\mathbf{P}$, i.e., the new colour at $\mathbf{P}$ when lit by the spot light, as a function of $\mathbf{I}_a$, $\mathbf{I}_d$, $\mathbf{S}$, $\mathbf{L}$, and d. Explain your formula.



(d) (**2 points each**) How are the two types of attenuation controlled by the user? Explain this in terms of the parameters in your formula from (c).

Question 3: Implicit surfaces

(a) (**2 point**) Given a point $\mathbf{P} = (p_x, p_y, p_z)$, how do we check whether the point is on the implicit surface described by $f(x, y, z) = 0$?

(b) (**2 points**) Assuming that $\mathbf{P}$ is on the surface, how do we compute the normal of the surface at $\mathbf{P}$?

(c) (**7 points**) Marching cubes is an algorithm for converting implicitly given surfaces to polygonal meshes. Describe the algorithm using annotated pseudo-code and sketches. You can assume that we want to turn the surface given by $f(x, y, z) = l$, for some value of l, into a mesh over a grid of n^3 cubes. It is not necessary to explicitly enumerate all the possible cases for a single cube arising in the algorithm.

Question 4: Z-buffer

(a) (**3 points**) Explain the role of the z-buffer (also called depth-buffer) in the graphics pipeline.

(b) (**3 points**) Can the z-buffer handle semi-transparent objects? If so, explain how. If not, give an example when it fails to produce the correct result.

(c) (**4 points**) The a-buffer can produce better results than the z-buffer. How is this achieved, and at what expense compared to the z-buffer? [Hint: Note that global super-sampling is inefficient. Consider which pixels need to be super-sampled and which do not, and why.]

Question 5: Texturing

To achieve more realistic results, the diffuse component in an illumination model is often replaced by or combined with intensity (colour) values from a texture.

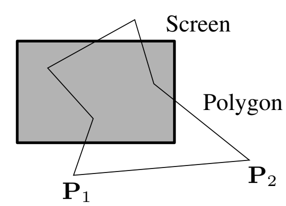
(a) (**2 points each**) Give three more examples of how textures are used to replace certain quantities or parameters in illumination models and to what effect.

(b) (**3 points**) A pixel on the screen typically does not correspond to a single pixel in a texture (texel), and thus up- or down-sampling is needed. How is this handled and how does it improve on the naive point sampling?

(c) (**3 points**) Explain what a MIP map is (use a sketch), what are its advantages over a standard texture, and how does tri-linear averaging on MIP maps work.

Question 6: Polygon clipping

Polygon clipping is one of the operations performed in the graphics (OpenGL) pipeline.



(a) (**3 points**) Describe a situation (using a sketch) where skipping polygon clipping would produce the wrong rendered result. For what other reason is clipping performed?

(b) (**6 points**) Describe the Sutherland-Hodgman polygon-clipping algorithm in 2D screen space, including all transition types that arise in it. Assume that the polygon has n vertices $\mathbf{P}_1$ through $\mathbf{P}_n$, and that the screen is rectangular.

(c) (**2 points**) What changes need to be made to (b) to extend the algorithm to handle clipping in 3D against the viewing frustum?

Question 7: Ray-tracing

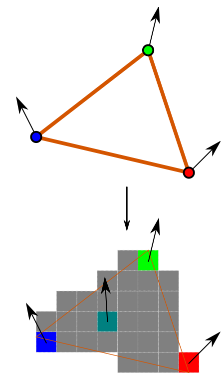
(a) (**6 points**) Describe, in pseudocode, the basic ray-tracing algorithm (no bounces) with only ambient illumination that produces the final rendered image. Assume that the screen has $a \times b$ pixels with centres $\mathbf{P}_{a,b}$, the centre of projection is at $\mathbf{E}$, and that there are n objects in the scene. Each object comes with its own (ambient) colour and an intersection routine which returns, given a ray, all existing ray-object intersections.

(b) (**2 points**) Shadows add more realism to a rendered scene. In a ray-tracer, how does one check whether a visible point is lit or in shadow with respect to a single spot light in the scene?

(c) (**2 points each**) Describe two effects which cannot be captured by (distributed) ray-tracing: one that is well modelled by radiosity and one by photon maps. Include a sketch and justify your answer in both cases.

Question 8: Rasterisation

Rasterisation (or the rasteriser) is an important step in the graphics pipeline.



(a) (**3 points**) What is rasterisation and where does it fit into the graphics pipeline? Include a diagram of the whole graphics pipeline (or relevant parts thereof).

(b) (**4 points**) Assume that you have a line-drawing algorithm that rasterises line segments. How can this be integrated into an efficient algorithm for rasterising triangles? Flood-fill is not considered efficient in this context.

(c) (**2 points**) What problems can arise when triangle rasterisation is not well aligned with line-segment rasterisation? Give an example.

(d) (**2 points each**) Describe, in brief, two shading models: One in which colours are interpolated and one in which normals are interpolated, in the rasterisation step, over a given triangle. Where in the graphics pipeline are these shading models implemented? Which one is more efficient (for a typical scene) and why?

Question 9: Transformations

In graphics, 3D transformations are typically described by 4×4 matrices in homogeneous coordinates, and these are cocatenations of atomic transformations comprising scaling (both uniform and non-uniform), rotation, shearing, and translation.

(a) (**3 points each**) Decompose the transformation given by

$$\mathbf{M} = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

into exactly two atomic transformations, i.e., in the form $\mathbf{M} = \mathbf{M}_1 \cdot \mathbf{M}_2$. Give the individual matrices $\mathbf{M}_1$ and $\mathbf{M}_2$, describe the corresponding transformations geometrically in one sentence, and include a sketch.

(b) (**2 points**) What is the resulting matrix $\mathbf{M}'$ if the two atomic transformations from (a) are applied in reverse order, i.e., $\mathbf{M}' = \mathbf{M}_2 \cdot \mathbf{M}_1$?

(c) (**2 points**) Give an example of a pair of atomic transformations, not both of the same atomic type, that produce the same transformation when concatenated in either of the two possible orders. Explain your answer.