Module 3G4: Medical Imaging & 3D Computer Graphics

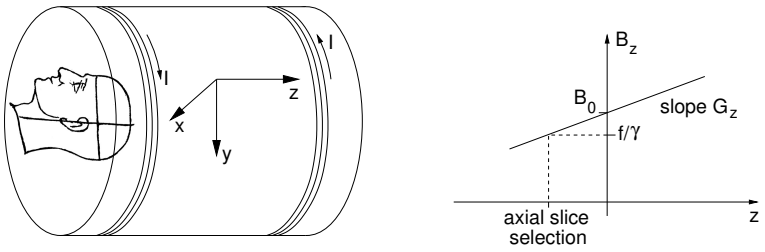
Solutions to 2023 Tripos Paper

1. **MRI imaging and slice selection**

(a) An MRI scanner incorporates three gradient coils that can apply an arbitrary linear variation to the field in each of the principal scanning directions:

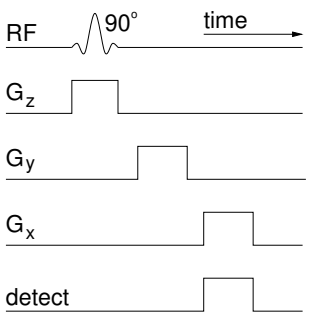
$$\frac{\partial B_z}{\partial z} = G_z \ , \quad \frac{\partial B_z}{\partial x} = G_x \ , \quad \frac{\partial B_z}{\partial y} = G_y$$

For example, the z-gradient might look like this:



If we apply G_z at the same time as the RF pulse, we can restrict imaging to a particular axial slice by tuning the frequency of the pulse, since only protons exposed to a field f/γ are excited. Or we could use x- or y-gradients to select sagittal or coronal slices respectively.

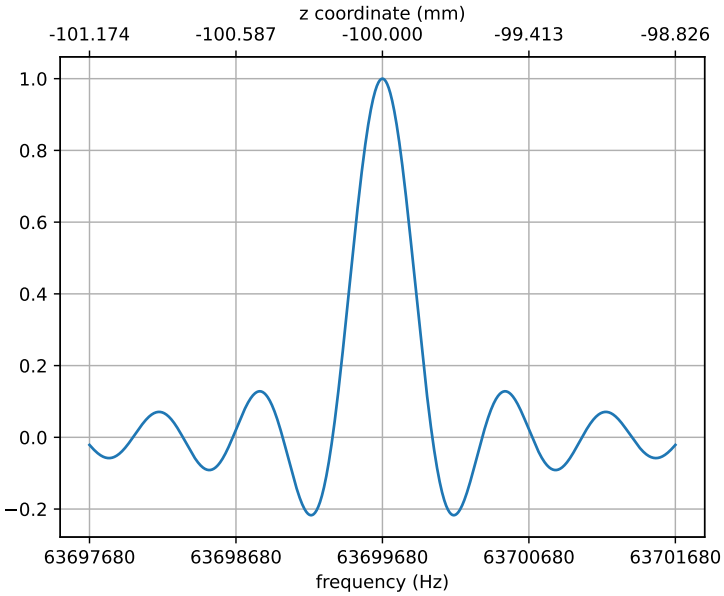
Apart from slice selection, we can also use the gradient fields for phase and frequency encoding, allowing us to infer where in a slice a particular NMR signal comes from. Continuing the example of an axial slice, by applying a y-gradient after the RF pulse, the precessions will go at different speeds according to their y-coordinates. We then switch off the y-gradient, so the precessions return to their original speeds, but their phases now encode the y-coordinate. We then apply an x-gradient at the same time as we detect the NMR signal. The x-gradient causes protons to precess at different speeds according to their x-coordinates. So, when we detect the signal, phase encodes y and frequency encodes x.



We cannot decompose the detected signal into distinct (x, y) signals from a single imaging sequence like the one above. But, by repeating the sequence with different values of G_y, we can. This is known as *k-space encoding*. [30%]

(b) (i) Substituting values into the given equation, we find $t_p = (\pi/2)/(2\pi \times 42.58 \times 10^6 \times 2 \times 10^{-6}) = 2.936$ ms. [5%]

(ii) From the table of Fourier transforms in the *Information Data Book*, an unmodulated pulse of duration $b = 2.936$ ms has a spectrum $\sim \mathrm{sinc}(\pi b f)$ with zeros at n/b Hz for $n = 1, 2, \ldots$ Modulating with the carrier shifts this spectrum up to 63699680 Hz.



The spectrum is therefore as shown above, where the horizontal axis is labelled both in terms of frequency f and also the z coordinates of excited spins, calculating according to $B_z = f/\gamma = 1.5 + 0.04z$. [30%]

(iii) The rectangular RF pulse is not bandlimited, so the resolution of the imaged axial slice is poor: its main lobe is 0.4 mm wide while the side lobes decay slowly and extend indefinitely. Ideally, the RF pulse would have a rectangular spectrum to excite spins in a narrow axial slice, but this would require a sinc-shaped pulse of infinite duration. Practically, we would compromise with some sort of windowed sinc pulse, giving a fairly uniform spectrum over a narrow range of frequencies, a rapid roll-off and then little energy outside of the narrow frequency range. Care would need to be taken to balance the amplitude of the pulse with its duration, to ensure that the flip angle remains $90°$.

The result would be fairly uniform imaging of a selected axial slice of a certain thickness, with good suppression of slices outside this range. The slice thickness is proportional to the bandwidth of the RF pulse and inversely proportional to the gradient field strength. For better axial resolution, we would require a thinner slice but this is difficult to achieve

in practice since: for safety reasons, there is an upper limit of around 80 mT/m for the gradient field strength; an RF pulse with a narrow bandwidth implies a wide main lobe of the windowed sinc function, which requires a long on-time and hence a long examination; a very thin slice would imply few excited spins and therefore a low SNR, unless the main field strength were very high, though B_0 is currently limited to 8 T for safety reasons. [35%]

Assessors' remarks: This question tested candidates' understanding of signal localisation in MRI imaging. In (a), most candidates offered solid, textbook accounts of slice selection, phase and frequency encoding. The most common weakness was neglecting to explain the need to repeat the sequence with different phase gradients. In (b), although many candidates located the slice correctly at $z = -10$ cm, very few understood that the rectangular pulse implied a sinc-shaped spectrum, and hence a sinc-shaped slice selection region. Of those who did understand this, even fewer were able to apply basic Fourier theory (with the necessary formulae in the data book) to determine the width of the sinc's main lobe. Consequently, most candidates did not understand the cause of the poor axial resolution in (b)(iii), and hence struggled to suggest how the resolution might be improved.

2. **Triangle meshes and vertex normals**

(a) Possibly the most obvious way to store this mesh is to have a single list of triangles, for each of these triangles to store the three vertices, and for each of these vertices to store the (presumably 3D) locations of that vertex. An alternative is to store the vertices as a separate list, where the location of that vertex is stored for each vertex. Then a separate list of triangles stores, for each triangle, the three vertex numbers (but not the actual locations) that are associated with that triangle.

Various possible features or operations (of which only two are required) include:

Storage size / inefficiency — the single list of triangles is much less efficient in terms of memory, typically taking up about three times as much space as the separate lists of vertices and triangles, though that depends on what type is used for locations and vertex numbers.

Floating point accuracy / rounding errors — in the single list of triangles, the same vertex is stored multiple times (generally up to six, since that is typically the number of triangles referencing each vertex). Any operations on that mesh may result in rounding errors which mean the same vertex ends up with very slightly different locations.

Manually editing the mesh — this generally requires searching the mesh for which part is closest to where a user clicked, and it is easier to do this directly with the vertex list. For the single triangle list, all identical vertices would have to be found and moved simultaneously. In addition, moving a vertex then only means replacing the location of that vertex, since the triangle list can stay exactly the same. Removing a triangle would be the same with each scheme.

Checking for consistency — there are many consistency checks, for instance checking whether a mesh is watertight. These are generally much easier with the second (or more complex) schemes. [20%]

(b) For the normal to the triangle, note that the cross product of two vectors always generates a vector perpendicular to both, so given **a** and **b** are both in the plane of the triangle, and (for a non-degenerate triangle) are not parallel, then the cross product must represent the triangle normal. The normalised surface normal would then be given by $\hat{\mathbf{n}}$.

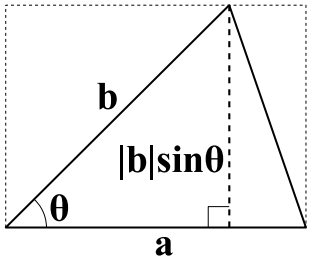


Fig. 1

For the area of the triangle it is easier to think of the magnitude of the cross product, defined as $|\mathbf{a}||\mathbf{b}| \sin\theta$ where θ is the angle between the vectors. If **a** is considered the base of the triangle, this can easily be interpreted as the well known 'half base times height' formula, for instance as in Fig. 1. The area is therefore given by $|\mathbf{n}|$. [10%]

(c) (i) Any operation, for instance lighting, which requires interpolation across the mesh structure, needs the interpolated variable to be defined at the vertices rather than over the triangle. Hence if the normals are involved in any way in an interpolated operation, they must be defined at the vertices. Note that this is the case for both Gouraud and Phong shading, for instance. [10%]

(c) (ii) If the calculated vertex normal is $\mathbf{n}_v$, then (where $\oplus$ represents circular addition, so that, in this case, $5 \oplus 1 = 0$):

$$\begin{aligned}
\mathbf{n}_v &= \frac{1}{2}\sum_i \mathbf{a}_i \times \mathbf{b}_i \\
&= \frac{1}{2}\sum_i (\mathbf{o} - \mathbf{v}_i) \times (\mathbf{o} - \mathbf{v}_{i\oplus 1}) \\
&= \frac{1}{2}\sum_i (\mathbf{o} \times \mathbf{o}) - (\mathbf{o} \times \mathbf{v}_{i\oplus 1}) - (\mathbf{v}_i \times \mathbf{o}) + (\mathbf{v}_i \times \mathbf{v}_{i\oplus 1}) \\
&= \frac{1}{2}\sum_i (\mathbf{v}_{i\oplus 1} \times \mathbf{o}) - (\mathbf{v}_i \times \mathbf{o}) + (\mathbf{v}_i \times \mathbf{v}_{i\oplus 1})
\end{aligned}$$

since any vector cross-product with itself is zero, and reversing the order of the cross-product changes the sign. Splitting out the sums:

$$\mathbf{n}_v = \frac{1}{2}\sum_i (\mathbf{v}_{i\oplus 1} \times \mathbf{o}) - \frac{1}{2}\sum_i (\mathbf{v}_i \times \mathbf{o}) + \frac{1}{2}\sum_i (\mathbf{v}_i \times \mathbf{v}_{i\oplus 1})$$

However, the first two of these sums are identical: they both end up summing over all vectors $\mathbf{v}_i$. Hence these will cancel out, leaving the final result:

$$\mathbf{n}_v \;=\; \frac{1}{2}\sum_i (\mathbf{v}_i \times \mathbf{v}_{i\oplus 1})$$

Note that it would also be possible to demonstrate this by writing out $\mathbf{n}$ in full for each of the six triangles in the mesh, in which case the following result is also valid, though less concise:

$$\mathbf{n}_v \;=\; \frac{1}{2}\left((\mathbf{v}_0 \times \mathbf{v}_1) + (\mathbf{v}_1 \times \mathbf{v}_2) + (\mathbf{v}_2 \times \mathbf{v}_3) + (\mathbf{v}_3 \times \mathbf{v}_4) + (\mathbf{v}_4 \times \mathbf{v}_5) + (\mathbf{v}_5 \times \mathbf{v}_0)\right) \qquad [30\%]$$

(c) (iii) The result in (ii) does not depend on the central vertex $\mathbf{o}$ at all, and hence will not be changed as $\mathbf{o}$ varies. But the actual normal of $\mathbf{o}$ *is* likely to change as the location varies. Imagine moving the central point close to one of the edges: naturally one would expect the normal at that point to be much more like the normal at one of the closer outer vertices, and to be less affected by those that are further away.

Hence this is only a good way to calculate vertex normals if all the triangles are fairly similar sizes and shapes, in which case the central vertex will always be fairly equidistant from the surrounding vertices. [15%]

(c) (iv) We calculated $\mathbf{n}_v$ by summing the equation for the triangle normal and area: so the result represents a summed area as well as a summed surface normal. It would be most natural to see this as the total area of all the triangles containing the central point $\mathbf{o}$, but this cannot be the case, since it does not depend on $\mathbf{o}$. Imagine moving the central point a long way out of the plane of the surrounding vertices: this would greatly increase the area of all the triangles, but it does not change $\mathbf{n}_v$ at all.

Hence $\mathbf{n}_v$ must actually represent an area of the polygon defined by the vertices $\mathbf{v}_0$ to $\mathbf{v}_5$. These vertices do not have to lie in a plane, and in fact this area is the largest projected area of that polygon. So $|\mathbf{n}_v|$ can be used to give an estimate of the area of any 3D polygon defined by vertices $\mathbf{v}_i$. [15%]

Assessors' remarks: This questioned tested candidates' knowledge of polygon storage and the calculation of vertex normals. The fairly standard question in (a) was surprisingly poorly answered, with marks often lost due to vagueness or lack of precision in the definition of each storage scheme. (b) was a basic question on vector areas: some candidates again lost marks here for lack of precision, for instance omitting that it is the magnitude which represents the area. (c)(i) was answered well be nearly everyone, but (ii) was clearly the hardest part of the question (which is why it was also worth the most marks). There were some perfect answers, but many candidates stopped at an early stage before noting that much of the cross product could be cancelled, or were confused about vectors and labelled the triangle edges directly as $\mathbf{v}_i$. (iii) was well answered, with many candidates noting the expected loss in accuracy as the midpoint approached the outer polygon. Few spotted in (iv) that the calculated vertex normal is also the polygonal area.

3. **Multi-segment curves and repeated points**

(a) The continuity *within* each segment is always at least up to C_3 since all segments are defined using a cubic polynomial. However, the continuity at the joins of each curve (i.e. where the parameter is $t = 1$ for one curve, and $t = 0$ for the next) depends on the type of curve used. For Bezier curves, each segment shares only one control point, so C_0 continuity is guaranteed. Up to C_1 continuity is possible by ensuring that the neighbouring control points are in a straight line including the shared control point. For multi-segment curves, three join points are shared by each segment. Catmull-Rom curves only exhibit C_1 continuity at these joining points, whereas B-splines exhibit C_2 continuity. [15%]

(b) The convex hull property applied to a curve states that the whole curve must lie within the convex hull of the control points which define that curve, i.e. the curve is within any polygon formed by joining the control points. A more mathematical definition of the convex hull is that it contains any point which is a linear interpolant of the control points: i.e. a weighted sum of the control points where all weights are between zero and one, and the weights sum to one. The B-spline and Bezier both have this property but the Catmull-Rom does not.

The convex hull property is useful since it constrains the possible location of the curve. If we need to know whether a curve intersects a given region (for instance a display window), we can first check if the convex hull of the control points intersects this region. If it does not, then neither does the curve. [15%]

(c) (i) The easiest way to calculate this is to replace $\mathbf{p}_1$ with $\mathbf{p}_2$ in the curve definition, and then set t to 0:

$$\mathbf{p} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_2 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

$$\mathbf{p} = \frac{1}{6} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_2 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

$$\begin{aligned} \mathbf{p} &= \frac{1}{6}(\mathbf{p}_0 + 4\mathbf{p}_2 + \mathbf{p}_2) \\ &= \frac{1}{6}\mathbf{p}_0 + \frac{5}{6}\mathbf{p}_2 \end{aligned}$$

Hence the left-hand end point is one-sixth of the way on the line between $\mathbf{p}_2$ and $\mathbf{p}_0$. This is shown in Fig. 2 below. By symmetry, the right-hand end of A is on the line $\mathbf{p}_2$ to $\mathbf{p}_3$.

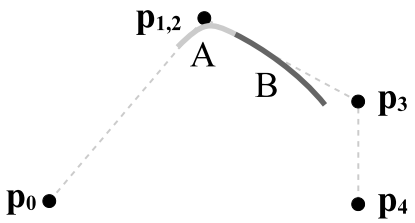


Fig. 2

[20%]

(c) (ii) Following the same reasoning as for (i) above:

$$\mathbf{p} = \frac{1}{6}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}\begin{bmatrix} \mathbf{p}_2 \\ \mathbf{p}_2 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

$$\begin{aligned} \mathbf{p} &= \frac{1}{6}(\mathbf{p}_2 + 4\mathbf{p}_2 + \mathbf{p}_2) \\ &= \mathbf{p}_2 \end{aligned}$$

Hence the left-hand point is exactly at $\mathbf{p}_2$. The right-hand point (needed to sketch the curve correctly) is one-sixth of the way on the line from $\mathbf{p}_2$ to $\mathbf{p}_3$, following similar logic to the result in (i). The sketch is shown in Fig. 3 below.

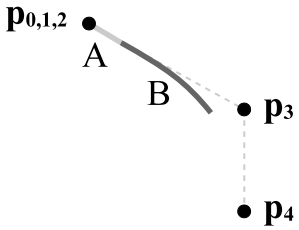


Fig. 3

[20%]

(c) (iii) The result can be deduced from part (i) and (ii) and the convex hull property. We know that A must start one-sixth of the way along the line from $\mathbf{p}_2$ to $\mathbf{p}_0$, and (by symmetry) B must stop one-sixth of the way along the line from $\mathbf{p}_2$ to $\mathbf{p}_4$. A must be in the convex hull of $\mathbf{p}_0 \ldots \mathbf{p}_3$, which is a straight line, and B must be in the convex hull of $\mathbf{p}_1 \ldots \mathbf{p}_4$, which is also a straight line. So both curves lie on these straight lines and they must therefore meet at the combined point. This is shown in Fig. 4 below.

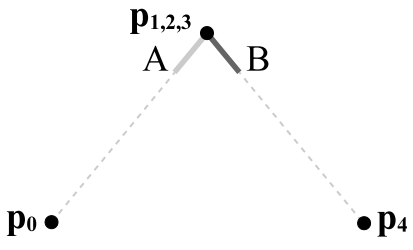


Fig. 4

This can also be deduced mathematically, for the start $t = 0$ of segment B, again following on from the previous logic:

$$\mathbf{p} = \frac{1}{6}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}\begin{bmatrix} \mathbf{p}_2 \\ \mathbf{p}_2 \\ \mathbf{p}_2 \\ \mathbf{p}_4 \end{bmatrix}$$

$$\begin{aligned} \mathbf{p} &= \frac{1}{6}\left(\mathbf{p}_2 + 4\mathbf{p}_2 + \mathbf{p}_2\right) \\ &= \mathbf{p}_2 \end{aligned}$$

[20%]

(c) (iv) In all cases, the segments themselves exhibit at least C_3 and G_3 continuity. The join between segments A and B is also always C_2 continuous, since that is a property of multi-segment B-spline curves. However, the geometric continuity at the join varies. For (i) and (ii) it is still G_2 but for (iii) it is only G_0, i.e. only continuous in value.

This is helpful in the situation where we need to use the B-spline to get a smooth curve but want to manipulate certain points: (iii) allows us to force the curve to contain a sharp bend at a particular point, or change of direction, whereas (i) allows us to start in a specified direction. Note that if these are motion paths, the acceleration is still continuous, so the motion in (iii) must have zero velocity and acceleration at the G_0 point. [10%]

Assessors' remarks: This question tested candidates' knowledge of cubic parametric splines, continuity and the convex hull property. Despite some very good answers to the introductory work in (a) and (b), many candidates received few marks due to simply stating what they could remember about splines rather than answering the specific questions. The more quantitative work in (c) was well answered by most, with some good reasoning and some good sketches, though several candidates dropped marks due to not wanting to sketch the cubic curves as straight lines in (ii) and (iii). In (iv), not many candidates noted that the curves still all had C2 continuity even when they were reduced to G0.

4. **Shading and shadow z-buffers**

(a) I_λ is the intensity of the reflected light of colour λ, where $\lambda \in \{r, g, b\}$ for red, green and blue.

I_λ depends on several terms. First, there is the ambient reflection term, $c_\lambda I_a k_a$, which models indirect illumination of the surface. c_λ, where $0 \leq c_\lambda \leq 1$, specifies the colour of the surface. I_a is the intensity of the general background illumination, and k_a is the surface's ambient reflection coefficient.

The next two terms in the model are summed over each point light source i with intensity I_{pi}. First there is the diffuse reflection term, $c_\lambda k_d \mathbf{L}_i.\mathbf{N}$, which models even reflection of the light source in all directions. Diffuse reflection is greatest when the surface is pointing directly towards the light source, and tails away to zero when the surface is side-on to the light source. $\mathbf{L}_i$ is the unit vector from the surface point towards the light source, $\mathbf{N}$ is the unit surface normal and k_d is the surface's diffuse reflection coefficient (small for dark surfaces, high for bright surfaces).

Finally, there is the specular reflection term, $k_s \left(\mathbf{R}_i.\mathbf{V}\right)^n$, which models directional reflection of the light source along the unit mirror vector $\mathbf{R}_i$. $\mathbf{V}$ is the unit vector from the surface point towards the viewer. The viewer only perceives the specular highlight (or glint) when looking along the mirror direction, or at least close to it. k_s is the surface's specular reflection coefficient (small for matte surfaces, high for shiny surfaces), and n is the specular exponent that determines the tightness of the glint. n is high for a tight highlight (e.g. a perfect mirror) and small for a more blurred highlight (e.g. aluminium).

The diffuse and specular terms are attenuated by a shadow factor S_i, where $0 \leq S_i \leq 1$. S_i is the fraction of the pixel shaded from the light source, often calculated using a shadow z-buffer algorithm. There is also a depth attenuation factor f_{att}, usually of the form

$$f_{\text{att}} = \min\left(\frac{1}{a_1 + a_2 d + a_3 d^2}\ ,\ 1\right)$$

where a_1, a_2 and a_3 are constants, and d is the distance from the light source to the surface point. The depth cueing ensures that surfaces with the same orientation, but at different distances from the viewer, are not assigned the same intensity. [20%]

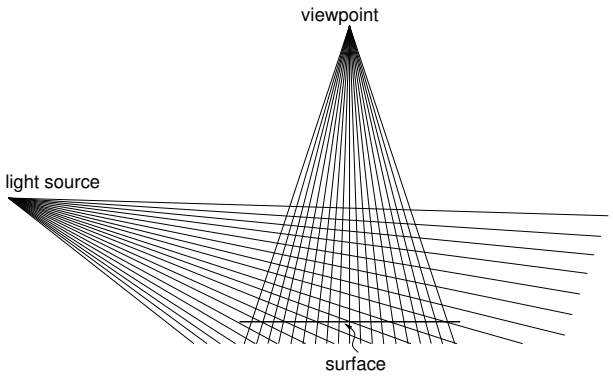
(b) The finite precision of the shadow z-buffer will affect the critical $z'_s > z_b$ test when $z'_s \approx z_b$. We can imagine two types of artifact that might arise.

First, consider a point A at the foot of a wall casting a shadow onto a nearby point B on the floor. Since A and B are nearby each other, $z'_s \approx z_b$. With infinite precision arithmetic we expect that $z'_s > z_b$, but this might not be the case given the finite precision of the z_b values. The shadow might therefore not be calculated correctly for points on the floor near the wall, causing the shadow to detach from the base of the wall, as if the wall were flying. This phenomenon is known as *Peter Panning*.

Next, consider a point A on a surface that is not in shadow. With infinite precision arithmetic we expect that $z'_s = z_b$, but this might not be the case given the finite precision of the z_b values. Point A might therefore be judged to be in its own shadow! A moiré pattern of self-shadowing might be observed across the surface. This phenomenon is known as *shadow acne*.

To minimize these artifacts, the limited precision of the shadow z-buffer should not be squandered: the near and far clipping planes should be positioned as tightly as possible around the relevant part of the scene. Defining "relevant" in this context is not straightforward, depending on the view frustum (i.e. the part of the scene that is visible) and also any objects that might cast shadows into the view frustum. [20%]

(c) (i)



The frame buffer pixel spacing is isotropic, consistent with viewing from directly above. The shadow z-buffer pixel spacing and z_b values are consistent with the surface being nearer to the light source on the left and further away on the right. [20%]

(ii) Pixel B coincides with the centre of a shadow z-buffer pixel. Given that the surface is not shadowed, we would expect $z'_s = z_b$ though, given the discussion in (b), there is a possibility that $z'_s > z_b$ and the pixel is erroneously self-shadowed. Pixel A maps to the same shadow z-buffer pixel and hence the same value of z_b. However, z'_s will be less than for pixel B, and we would therefore expect no shadowing at A. Pixel C also maps to the same shadow z-buffer pixel and hence the same value of z_b. However, z'_s will be greater than for pixel B, and we would therefore expect erroneous self-shadowing at C. The resulting shadow acne is quite different from the phenomenon in (b). The problem here is not the k-bit storage precision of the shadow z-buffer, but its limited *spatial* precision. The problem is particularly acute with oblique and/or distant illumination where several frame buffer pixels might map to the same z-buffer pixel. [20%]

(iii) The most straightforward way to suppress shadow acne is to add a bias into the shadow depth test, i.e. cast a shadow only when $z'_s > z_b + \epsilon$, where ϵ is the bias. A problem with this simple fix is that it tends to exacerbate Peter Panning. A more sophisticated approach adjusts the bias according to the angle between the surface and the light rays, so that the minimal amount of bias is used at each point: this is known as *slope-scaled* depth bias. [20%]

Assessors' remarks: This question tested candidates' understanding of the Phong illumination and reflection model, and the shadow z-buffer algorithm. Almost all candidates demonstrated excellent knowledge of the Phong model in (a). In (b), there was a general appreciation of the difficulties posed by finite-precision z-buffers, but many candidates

framed their responses in terms of hidden surface removal instead answering the specific questions about shadowing. Responses to (c) were better, with many candidates correctly identifying the cause of the self-shadowing artefacts, though there was little appreciation of the distinct roles played by the spatial and numerical precision of the shadow z-buffer.

Andrew Gee & Graham Treece
May 2023