Engineering Tripos Parts IIA and IIB THIRD AND FOURTH YEARS

Module 4F9: Medical Imaging & 3D Computer Graphics

Solutions to 2007 Tripos Paper

1. **Ultrasonic imaging**

(a) Ultrasound echos from structures deep in the body have to travel through more tissue and so are of lower amplitude. The deeper a scatterer, the longer it takes for the echo to get back to the probe. We can thus correct for the attenuation of deep structures by applying a gain that increases with time. This is called time-gain compensation. In order to work out the appropriate rate of increase we need to know the speed of sound in the material, c, in m/s and the rate of attenuation, a, in dB/m. [20%]

(b) We want to know the rate at which sound is attenuated with time in dB/s. This is given by $a \times c$ because dB/m $\times$ m/s = dB/s. The required rate of time-gain compensation will be minus this value, i.e. a positive gain rather than a negative rate of attenuation. [15%]

(c) (i) Distances will be overestimated because the sound is slower in the water than the calibration of the machine.

$$1.6 \times \frac{1540}{1425} = 1.73$$

Hence the total depth will appear to be $1.73 + 1.4 = 3.13$ cm. [15%]

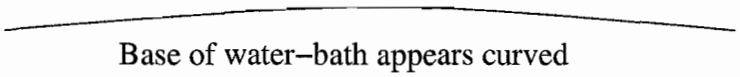
(ii) The true distance, d, from the centre of curvature of the probe to point B is given by Pythagoras.

$$4^2 + 7.6^2 = d^2 \Rightarrow d = 8.5884\,\text{cm}$$

2.5884 cm of this is in the cold water and will be overestimated by a factor of $\frac{1540}{1425}$ giving a measured distance of 2.797 cm. The measured distance of point B from the centre of curvature of the probe is thus 8.797 cm.

The ultrasound pulse that is backscattered at B is fired at an angle θ to the vertical, where $\tan(\theta) = 4/7.6 \Rightarrow \theta = 27.7585°$. Hence the vertical component of the distance from the centre of curvature to B is given by $8.797\cos(27.7585) = 7.785$ which is 0.185 cm more than it should be. The depth of the water-bath is thus measured as 3.185 cm. [30%]

(iii) The base of the water bath appears to be bent up at the sides. An exaggerated sketch is shown below. Although the absolute error is just under 2 mm, the change in error between the centre and the side is only 0.5 mm. The wavelength at 3 MHz is about 0.51 mm, so we would not expect to see the curvature in the ultrasound image. [20%]



Assessors' remarks: This question was about ultrasonic imaging of soft tissue. It tested the candidates understanding of time-gain compensation and length distortion due to variations in sound speed in different materials. Candidates showed a good understanding of both these concepts and the quality of the answers was high. A surprisingly large number of candidates even managed to correctly solve the more difficult numerical problem in part (c)(ii). The comments offered in answer to the last part of the question indicated a satisfactory understanding of the ultrasonic imaging process.

2. **CT and nuclear medicine imaging**

(a) The 2D Radon transform maps a function $f(x, y)$ to the set of its integrals over lines at perpendicular angles, ϕ, and distances, s, from the origin.

A projection is a set of values of the radon transform that all have the same angle, ϕ, but different offsets, s, such that they cover the whole of the object being scanned.

The value of the projection at angle ϕ and offset s is proportional to minus the log of the transmitted X-ray intensity over the incident X-ray intensity at angle ϕ and offset s. [20%]

$$p_\phi(s) = -\ln \frac{I_\phi(s)}{I_0}$$

(b) Direct Fourier reconstruction

- Take n projections $p_\phi(s)$ of the object.
- Find 1D Fourier transforms of the projections.
- Use the set of 1D Fourier transforms to tile the spatial frequency plane. Each transform contributes a radial strip (ω, ϕ), for a particular angle ϕ, passing through the origin.
- Resample this data to produce a regular sampling of the spatial frequency plane in Cartesian coordinates (ω_x, ω_y). This involves interpolation to fill the gaps between the radial strips.
- Take the 2D inverse Fourier transform.

Filtered backprojection

- Take n projections $p_\phi(s)$ of the object.
- Convolve each projection with the inverse 1D Fourier transform of $|\omega|$. Where ω is the distance from the origin in the spatial frequency plane.
- Backproject each filtered projection across the area to be reconstructed and accumulate the values into the image. [20%]

(c) Low energy X-rays (gamma photons) are attenuated more than high energy ones, it therefore follows that the deeper an X-ray beam penetrates into matter, the more

its spectrum bunches up at the high energy end of the scale. This is known as beam hardening.

In clinical radiology, the superficial absorption of softer photons gives a radiation dose to the patient and serves only to distort the output of the imaging system, creating streak artifacts and cupping. Cupping is the name for the effect where the attenuation appears to be reduced towards the centre of the object.

It is therefore normal to pre-harden the beam using a beam hardening filter. This is a piece of aluminium or copper that the beam has to pass through before reaching the patient. [20%]

(d) Collimation is the process of ensuring that the direction of a gamma photon is known when it is incident on a detector.

It is required for X-ray computed tomography in order to eliminate gamma photons that have been subject to Compton scattering. It is achieved by using blocks of lead with holes drilled through, such that only photons that are travelling in the correct direction can get through.

It is required for SPECT and PET imaging in order know the line on which the radiation was emitted. For SPECT, a lead block with holes is used as described above. For PET, electronic collimation is used. This relies on the fact that when a positron is emitted it combines with an electron to produce two gamma photon travelling in opposite directions. The PET system is designed to detect coincident arrivals at opposite sides of the subject (within 10 ns). From this information, we can infer that the positron was emitted somewhere along the line joining the two detectors which were activated coincidentally. Electronic collimation is much more efficient than physical collimation with lead. [20%]

(e) A gamma photon from a particular radioactive decay has a known energy value. If a photon is subject to Compton scattering then it's direction and energy are changed. Because the direction has been changed, the photon is now no longer any use in the imaging process and only contributes distortion if detected. Gamma cameras are designed to detect the energy as well as the number of received photons so we can exclude low energy photons that have been subject to Compton scattering. This is called energy filtering. [20%]

Assessors' remarks: This was a multi-part question asking for brief descriptions of a variety of concepts related to X-ray computed tomography, SPECT and PET. The standard of the answers was generally excellent. The candidates seemed to have a clear grasp of this part of the course. They were able to present an impressive level of detail in their descriptive answers which resulted in a high average mark for the question. All the parts of the question were answered well and there was no area of discernible weakness.

3. **Bézier surface patches and radial basis functions**

(a) The full definition for the z coordinate of a surface patch in terms of parameters s, t is:

$$z(s,t) = \mathbf{S}\mathrm{M}_B\mathrm{Q}_z\mathrm{M}_B^T\mathbf{T}^T$$

where Q_z is the measured data, $\mathbf{T} = [t^3\ t^2\ t\ 1]$, $\mathbf{S} = [s^3\ s^2\ s\ 1]$, and t and s both vary from 0 to 1. However, the question only asks for the intersections of this surface with the $x = 0$ and $y = 0$ planes. These are simply Bézier curves in one parameter, so for instance the intersection with the $y = 0$ plane is given by:

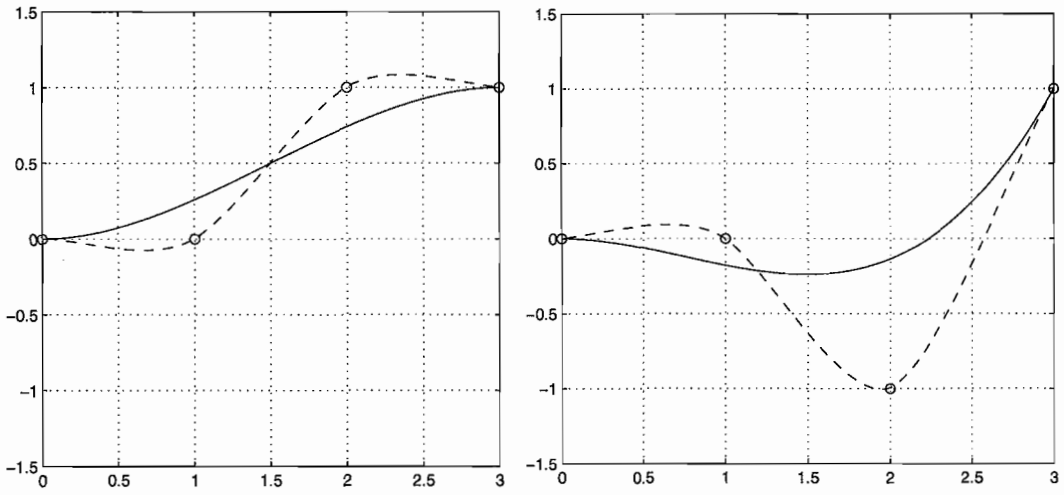
$$z(t) = \mathbf{T}\mathrm{M}_B\,[0\ 0\ 1\ 1]^T$$

In addition, the t parameter can be directly substituted for a scaled x parameter, since:

$$x(t) = \mathbf{T}\mathrm{M}_B\,[0\ 1\ 2\ 3]^T = 3t$$

Combining and evaluating these give:

$$z(x) = \frac{1}{27}(9 - 2x)\,x^2$$



(i) intersection with $y = 0$ (ii) intersection with $x = 0$

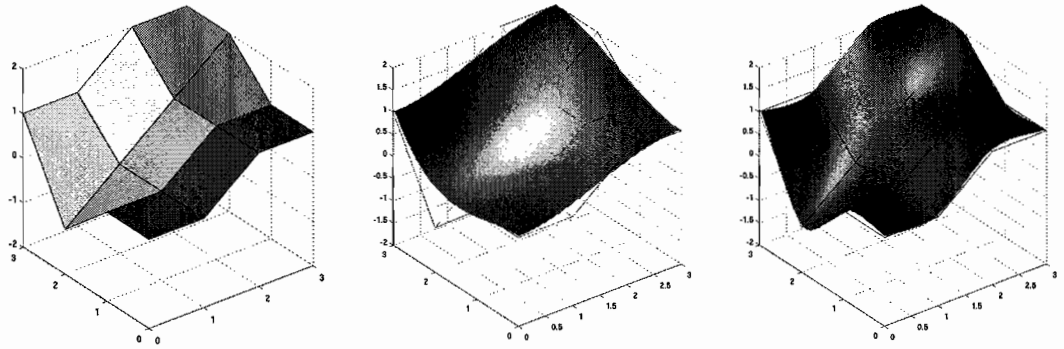
This curve is shown in bold in plot (i) above — note that it is not necessary to find the actual curve equation in order to sketch it, we simply need to know how a Bézier curve depends on the control points. It intersects the $z = 0$ plane only at $x = 0$ (the other intersection at $x = \frac{9}{2}$ is outside the range of the surface patch).

By a similar argument, the intersection with the $x = 0$ plane is given by:

$$z(y) = \frac{1}{27}(4y - 9)\,y^2$$

This curve is shown in bold in plot (ii) above. It intersects the $z = 0$ plane at $y = 0$ and at $y = \frac{9}{4}$. [40%]

(b) It is not necessary to calculate the RBF itself. All that is required is to note that the RBF must interpolate the original data (since it is directly inverted rather than iterated), and that, since we are using the thin-plate spline, it will fit a smooth function through the measured z data. This is enough to produce good curve sketches. These are shown dashed in the plots above (here calculated from the actual RBF, sketches approaching this form are acceptable). For the $y = 0$ plane, it should be evident that the intersections with $z = 0$ are exactly at $x = 0$ and $x = 1$, since the curve must interpolate the data. For the $x = 0$ plane, there are intersections at $y = 0$, $y = 1$ and $y \approx 2.57$. The latter has been calculated numerically — it is enough to note that this intersection must exist, and that it will be at a greater value of y than for the Bézier curve.



control mesh Bézier surface patch RBF interpolation

For completeness, the plots above show the actual surfaces using a Bézier patch and an RBF. These sketches are not required in answering the question. [30%]

(c) The Bézier patch is simple to calculate, but it is not really appropriate for this use. It only interpolates the corner points, not the others in the control mesh, so use of the original data is biased. It cannot be extended to larger data sets, since there is no way of overlapping the control meshes to correctly align the surface patches. A B-spline patch would do the job much better. It does exhibit the convex hull property, so that the entire surface will be limited by the range of the measured data.

The RBF fits the smoothest surface (in some sense) which will interpolate the original data. It is hence arguably the best interpolant of the data. However, its use is limited for very large data sets, due to the need to invert a very large matrix. For noisy data it can be made to approximate rather than interpolate by inverting the matrix iteratively rather than directly. [30%]

Assessors' remarks: There was a lot of variability in the answers, with some excellent and some very poor. Several candidates were unable to sketch the Bezier

curves, either not realising that the full surface patch was not required, or thinking that a Bezier curve should interpolate the points. Several also missed that it was only required to note that the RBF should interpolate the points smoothly for a good sketch. Part (c) was generally answered well, though only a couple of candidates noted the inconsistency in interpolation when using a Bezier and suggested the much more appropriate B-spline instead.

4. **Triangle lists and point normals**

(a) (i) The following are examples of consistency checks which could be carried out on this mesh (noting that it consists only of triangles, and that it represents an isosurface contained in a 3D data set):

- Basic reference checks, i.e. all the triangles refer to points which exist (do not contain numbers beyond the length of the point list) and all of the points are used by at least one triangle. The former is trivial, for the latter we need to go through each triangle, keeping a record of what points we have used, then at the end check if any have not been used.
- Each triangle refers to three points which are all distinct. Here we must check both that the references are distinct, and that the points that these refer to are in different locations.
- The vertex ordering is consistent in all the triangles. To check this, for each triangle, find all the triangles which share an edge with it. In each of these other triangles, the ordering of the vertices in this common edge should be reversed.
- The surface is watertight. We can check this by using the algorithm for calculating volume given in lectures (project each triangle onto a plane and sum the subsequent triangular columns). If this gives us a different answer when we use the x, y and z planes for our projection, the surface is not watertight. (Not required for the answer, but note that we have to repeat this with three planes, since this will only check for gaps in surface regions which are not orthogonal to the projection plane. If we have a gap in the surface which is aligned with the x plane, it will be orthogonal to the other two, so the apparent volumes for these will be the same). Alternatively, we can follow the method for the next check.
- No edges are shared by more than two triangles. We can check this by determining that each edge on each triangle is only involved in one other triangle.
- Triangles do not intersect each other. This is actually quite hard to check for. We have to compare each triangle with every other triangle. The intersection test involves finding the line of intersection of the *planes* containing each triangle, then determining if this line of intersection passes through each triangle, by looking at intersections between the triangle edges and this line, or alternatively checking whether all three points making up the triangle are on one side of this

line. Even if this line does intersect both triangles, we still need to check that the intersections overlap each other. [50%]

(ii) Other important issues for mesh storage formats are compactness (how much memory is required for storage to a given accuracy) and ease of manipulation (how difficult it is to perform common processing tasks on the storage format). [10%]

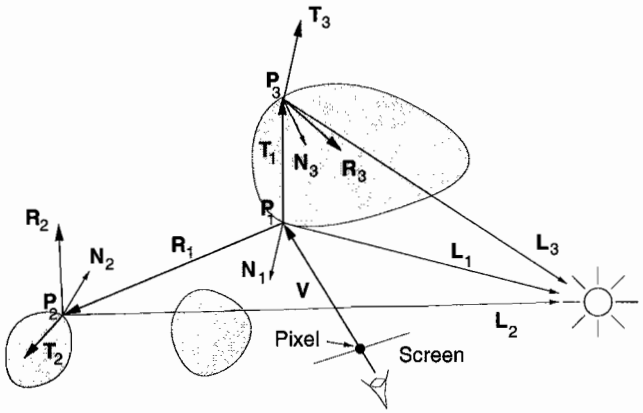
(b) (i) Given that the surface is an isosurface through 3D data, then the normal to this isosurface at each point is simply the normalised vector gradient of the 3D data. A simple way to estimate the vector gradient is to take central differences in all three directions, centred on each point on the surface. A more accurate technique would be to fit a B-spline in three parameters to the data and take derivatives of this spline function numerically. [20%]

(ii) Surface normals at each point can be worked out from the triangles containing that point. The normal to a triangle is easily calculated from the cross-product of two of the edges. However, since the point will be involved in more than one triangle, we need to take an average normal of all these triangles to get a reasonable estimate at the point. This estimate can be improved by weighting this average by the angle each triangle subtends at the point. [20%]

Assessors' remarks: A fairly straightforward, generally well answered question. It was common, however, for candidates to lose marks by simply missing that part (a) asked for a suggestion as to how the consistency check might be performed as well as what consistency checks were useful. Several candidates were confused by (b)(i) and wrote about triangle normals here, rather than noting that all we require is the gradient of the original scalar function.

5. **Ray tracing and intersection tests**

(a) Recursive ray tracing is a rendering technique which accounts for a degree of indirect illumination. It elegantly combines hidden surface removal, transparency effects and shadow computation into a single model.



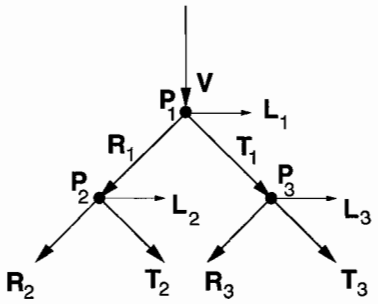
Simple ray tracing algorithms work in world coordinates. A ray $\mathbf{V}$ is fired from the centre of projection through every pixel on the screen. When the ray strikes an object at $\mathbf{P}_1$, further rays are spawned. One ray, $\mathbf{L}_1$, heads for the ith light source. If $\mathbf{L}_1$ passes through other objects on the way, then the illumination intensity is attenuated by a shadow factor S_i, depending on the number and opacity of objects in the way.

As well as these **shadow feelers**, the algorithm also spawns a **reflection ray** $\mathbf{R}_1$ and a **refraction ray** $\mathbf{T}_1$. The direction of the refraction ray is determined from the refraction indices using Snell's law. The intensity of the pixel is then

$$I_\lambda = c_\lambda I_a k_a + \sum_i S_i f_{att} I_{pi} (c_\lambda k_d \mathbf{L}_i.\mathbf{N} + k_s (\mathbf{R}_i.\mathbf{V})^n) + k_s I_{r\lambda} + k_t I_{t\lambda}$$

where $I_{r\lambda}$ is the intensity of the reflection ray and $I_{t\lambda}$ is the intensity of the refraction ray. k_t is a **transmission coefficient** in the range 0 to 1.

Values for $I_{r\lambda}$ and $I_{t\lambda}$ are found by recursively evaluating the equation at the surfaces $\mathbf{R}_1$ and $\mathbf{T}_1$ next intersect ($\mathbf{P}_2$ and $\mathbf{P}_3$), with $\mathbf{P}_1$ as the new viewpoint. So the progress of the algorithm follows a **ray tree**:



The tree is *constructed* top down, until either a ray fails to intersect an object or some predetermined maximum depth is reached. The tree is then *evaluated* bottom-up, as each node's intensity is computed from its children's intensities. Care must be taken to prevent aliasing in the ray-traced image.

Ray tracing produces results with an unmistakable signature, far from perfect photorealism. In addition to the aforementioned maximum tree depth, the two main approximations are (a) only specular reflection and refraction are considered, whereas in reality there are diffuse interactions too, and (b) the Phong model, itself a gross approximation, is employed each time a ray strikes a surface. There are a plethora of other effects not accounted for by ray tracing, such as scattering and dispersal in air. [45%]

(b) (i) The object should occupy as much of the bounding volume as possible. Otherwise, there is a high chance that a ray will intersect the bounding volume — triggering a large number of individual polygon tests — but not intersect the object. An elephant, being fairly spherical, would be a good candidate for a spherical bounding

volume. A giraffe, on the other hand, would occupy a small proportion of a bounding sphere, so an elongated cuboid would be a better choice. [15%]

(ii) A cuboid is defined by three pairs of parallel infinite planes, which we will denote $\mathcal{P}_{x1}$, $\mathcal{P}_{x2}$, $\mathcal{P}_{y1}$, $\mathcal{P}_{y2}$, $\mathcal{P}_{z1}$ and $\mathcal{P}_{z2}$. A ray can be defined parametrically as $\mathbf{r} = \mathbf{a} + \lambda\mathbf{b}$. We can trivially solve for the intersection of the ray with each infinite plane, obtaining the six solutions λ_{x1}, λ_{x2}, λ_{y1}, λ_{y2}, λ_{z1} and λ_{z2}. The ray therefore lies between $\mathcal{P}_{x1}$ and $\mathcal{P}_{x2}$ when λ is in the range $\lambda_{x1} \rightarrow \lambda_{x2}$, between $\mathcal{P}_{y1}$ and $\mathcal{P}_{y2}$ when λ is in the range $\lambda_{y1} \rightarrow \lambda_{y2}$, and between $\mathcal{P}_{z1}$ and $\mathcal{P}_{z2}$ when λ is in the range $\lambda_{z1} \rightarrow \lambda_{z2}$. To be inside the cuboid, λ must satisfy all three intervals simultaneously. If any two of the λ ranges are disjoint, the ray does not intersect the cuboid. An intersection algorithm would therefore calculate the λ ranges and check they all overlap. If the λ_y range is disjoint from the λ_x range, there is no need to calculate the λ_z range. [20%]

(c) We would need to expand the z-buffer to record not only the z 3D screen coordinate, but also the identity of the polygon whose rasterisation resulted in the z-buffer entry, and where on the polygon the rasterised pixel lies. If we were to do this, and run an otherwise standard z-buffer algorithm, the final z-buffer would contain, for each pixel (and therefore each ray), the required information about the closest polygon to the viewpoint. This is potentially much faster than a naive ray tracer: instead of calculating the intersection of n polygons with m rays, we are simply passing n polygons through a standard surface rendering pipeline, with a slightly more sophisticated z-buffer, but without the expensive illumination calculations. [20%]

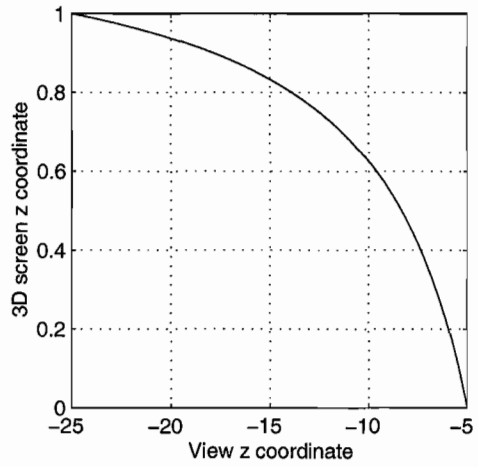
Assessors' remarks: This question tested the candidates' understanding of recursive ray tracing. It was mostly book work, with some extensions concerning bounding boxes and efficient intersection tests. The few candidates who attempted the question described the basic ray-tracing algorithm well in (a), and grasped the requirements for an efficient bounding box in (b)(i). There were two plausible algorithms for the cuboid/ray intersection test in (b)(ii). Only one student stated the need to store the polygon identity in (c).

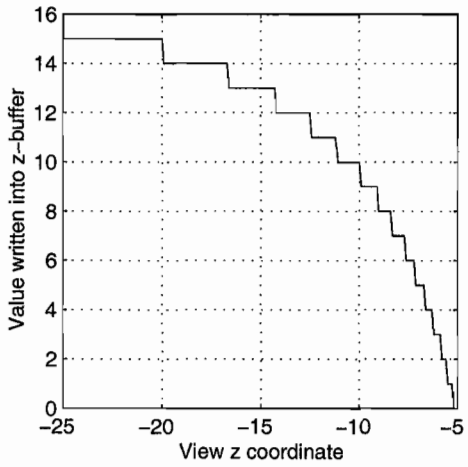
6. **z-buffers and 3D screen coordinates**

(a) The z-buffer algorithm is used for hidden surface removal in most practical implementations of the surface rendering pipeline. Earlier stages of the pipeline express all polygon vertices in 3D screen coordinates (x_s, y_s, z_s). z_s is a normalised representation of a point's depth, in the range 0 to 1. Nearby vertices have small z_s, distant ones have larger z_s. Depth values for individual pixels are derived by bilinear interpolation of the vertex values.

The z-buffer is an area of memory with the same dimensions as the frame buffer. When we write a pixel into the frame buffer, we also write its z_s value into the z-buffer. If a subsequent polygon attempts to shade the same pixel as an earlier one, we compare the new z_s with the value currently in the z-buffer, and write over the existing pixel in the frame buffer and the z-buffer only if the new point is nearer to the viewer. Initially, all entries in the z-buffer are set to $z_s = 1$. [20%]

(b) (i) The diagram below (left) shows a typical relationship between z_v and z_s, in this case for $n = 5$ and $f = 25$. Below right, we see the same relationship quantised to 4-bit precision: this is not required by the question, but illustrates a point we'll return to in part (c).





[10%]

(ii) The mapping from view to 3D screen coordinates can be written as

$$\begin{bmatrix} wx_s \\ wy_s \\ wz_s \\ w \end{bmatrix} = \begin{bmatrix} d/x_{\max} & 0 & 0 & 0 \\ 0 & d/y_{\max} & 0 & 0 \\ 0 & 0 & -f/(f-n) & -fn/(f-n) \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_v \\ y_v \\ z_v \\ 1 \end{bmatrix}$$

It is important that the mapping takes this form, since it means that the transformation between view and 3D screen coordinates is a projective one. This, in turn, guarantees that lines map to lines and planes map to planes, which is essential if we are to use any sort of linear interpolation at later stages of the rendering pipeline. [25%]

(c) With reference to the quantised sketch in (b)(i) above, z-buffer precision is going to be more of a problem at the back of the view volume (where a broader range of z_v values map to the same quantised z-buffer value) than at the front. We therefore need to ensure that points with z_v values of $-(200 - 0.1)$ map to the penultimate quantised z-buffer value of $2^k - 2$, and not to $2^k - 1$. The largest value of z_s that will be stored as $2^k - 2$ is infinitesimally less than $1 - 2^{-k}$. The limiting value of k should therefore map $z_v = -199.9$ to $z_s = 1 - 2^{-k}$. We therefore obtain the following equation:

$$\begin{aligned} 1 - 2^{-k} &= \frac{200(1 - 100/199.9)}{200 - 100} \\ \Leftrightarrow 2^{-k} &= 5.0025 \times 10^{-4} \\ \Leftrightarrow -k \log 2 &= \log(5.0025 \times 10^{-4}) \\ \Leftrightarrow k &= 10.965 \end{aligned}$$

The minimum value of k to guarantee correct depth discrimination between objects whose z_v values differ by 0.1 is therefore 11. [30%]

(d) When $n = 1$ and $f = 101$, even though the depth range is the same, there is more perspective compression since the ratio f/n is larger. We would therefore require more bits in the z-buffer to discriminate between objects whose z_v values differ by 0.1. [15%]

Assessors' remarks: This question tested the candidates' understanding of 3D screen coordinates and the z-buffer algorithm for hidden surface removal. Almost all candidates provided a good description of the z-buffer algorithm in (a), though remarkably few could sketch the relationship between z_v and z_s in (b)(i). Almost all candidates showed a good grasp of the mechanics and significance of homogeneous coordinates in (b)(ii). Parts (c) and (d) were closely related to an examples paper question that candidates should have been familiar with. However, only around half the candidates realised that quantisation effects were most significant at the far clipping plane and went on to derive the correct answer in (c), and only around a third of the candidates knew what they were doing in (d).

Andrew Gee, Richard Prager & Graham Treece
May 2007