COMP30019 Graphics and Interaction
Sample exam questions

Department of Computing and Information Systems,
The University of Melbourne

1 Conversion of image coordinates

Assume you have a digital image (or display) C columns wide and R rows high. A certain pixel has "matrix" coordinates (i, j), where i are rows and j are columns and the origin $(0, 0)$ is at the top-left corner pixel. What coordinates (x, y) does that same pixel have in the coordinate system used by Unity coordinate system? That is, how do you convert from matrix coordinates to Unity coordinates? How do you convert back the other way? What coordinates (x, y) does that same pixel have in a discrete Cartesian coordinate system, with the origin $(0, 0)$ at the lower-left corner pixel?

2 Image offset

Assume you have a file format for storing images with a header H bytes long, followed by the pixel data in ordinary left-to-right, top-to-bottom raster-scan order. Assume that pixels take up k bytes, and that the image is C columns across and R rows high. Consider the pixel in column j of row i. What is its byte offset from the beginning of the file? (Assume that columns are counted from zero at the left of the image, and that rows are counted from zero at the top of the image, but be aware how the formula would be different if one-origin indexing were used.)

3 Image rescale

Suppose you have an image c_1 pixels (columns) across and r_1 pixels (rows) high. You want to scale this image geometrically to be c_2 pixels across and r_2 pixels high (where c_2 and r_2 may be bigger or smaller than c_1 and r_1). (This might be the case if you had an already digitised image which you needed to stretch or shrink to fit some display layout.) Sketch the C code for performing this transformation.

4 Colour choice

You're creating a RoboCup monitor, which involves visualising soccer players running around the field. The teams colours are bright red on a bright green background of grass. What are potential problems with this? What alternatives would be reasonable?

5 Video images

Think about the problem of converting a non-interlaced T.V. video of 525 lines, 60 frames per second into a video of 625 lines, 50 frames per second (similar to that of converting U.S.A. T.V into Australian T.V but ignoring frame interlacing issues). Assume you have hardware that can digitise incoming lines to a sufficient resolution (number of pixels per line and number of intensity/colour levels), and then can re-convert the digital image-line data back into the appropriate analogue signal.

What simple or complex algorithms can you think of? What is the minimum number of lines of image memory required? What about the inverse problem of converting from Australian into U.S.A. video?

6 The robot's camera

Assume that the earth is a perfectly flat horizontal plain (and a plane too). Imagine a robot on the plain, with the optical centre of its (single) camera at height h above the ground. The camera is pointing straight ahead, with its optical axis horizontal. Assume that the camera can be treated as a pinhole camera with "focal length" f. See Figure 1.

(a) Show that that for a point on the ground in front of the robot, we can compute its 3D location from the 2D position of its projection in the image (so long as the point is visible within the limits of the image). Preferably give formulas for computing the point's 3D location (in some reasonable robot-centred coordinate system) from the 2D coordinates of its image projection.

(b) What do we need to assume about an object in terms of its position and shape in order to use the above method for computing the 3D location of an object from the position of its 2D projection in the image?

(c) What does the image line $y = 0$ correspond to in our everyday perceptions?

(d) What happens if the ground is not perfectly flat, but say has undulations in it? To what extent is it still possible to form some approximate or qualitative idea of the depth of objects away from the camera?

Note: For a 3D point with camera-centred coordinates (X, Y, Z) its image projection (x, y) under perspective projection is given by the equations:

$$\begin{aligned} \frac{x}{f} &= \frac{X}{Z} \\ \frac{y}{f} &= \frac{Y}{Z} \end{aligned}$$

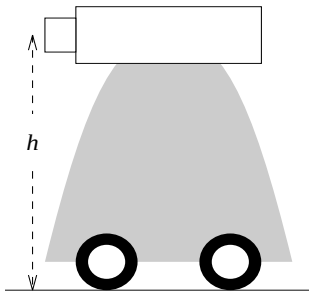

Figure 1: Geometry of the robot on the plain.

7 Composition of a scaling transformation

The scaling transformation described in lectures scales objects around the coordinate origin. That is, if you applied it to an object that wasn't at the origin, not only would it get bigger, but it would move further away from the origin (or smaller and closer if the scale factors were less than one).

Show how you can do a scaling by s_x horizontally and s_y vertically around any point $\begin{pmatrix} c_x & c_y \end{pmatrix}$ by composing together a translation, a scaling about the origin, and another translation.

Write these three separate transformations as 3×3 matrices operating in homogeneous coordinates, and multiply them all together (symbolically) in order to get a single matrix that performs the combined transformation.

Take the square defined by the four corner points $\begin{pmatrix} 0 & 0 \end{pmatrix}$, $\begin{pmatrix} 1 & 0 \end{pmatrix}$, $\begin{pmatrix} 1 & 1 \end{pmatrix}$, and $\begin{pmatrix} 0 & 1 \end{pmatrix}$. Draw this square on a piece of paper, preferably graph paper. What are these corner points in homogeneous coordinates?

We want to scale this square by a factor of 2 both horizontally and vertically about the centre of the square. Write down numerically the matrix that performs this transformation. Numerically apply this matrix to each of the corner points (in homogeneous coordinates) in order to obtain the homogeneous coordinates of the corners of the square after scaling. Convert the points back into ordinary 2D coordinates, and draw the resulting square on the same piece of paper. (These transformations all map straight line segments to straight line segments. Why?)

8 Transformation by composition

Give a sequence of transformation matrices required to build, including associated matrix algebra, a combined transformation matrix, $\mathbf{M}$, which transforms the object ABCDEFG into the object A'B'C'D'E'F'G' as shown below.

The (x, y) coordinates of each of the points above are as follows:
A $(2, 2)$, B $(4, 2)$, C $(4, 1)$, D $(5, 2.5)$, E $(4, 4)$, F $(4, 3)$, G $(2, 3)$,
A' $(2, 4)$, B' $(4, 6)$, C' $(3, 7)$, D' $(5.5, 6.5)$, E' $(6, 4)$, F' $(5, 5)$,
and G' $(3, 3)$.

How did you derive your transformation? How else could you derive such a transformation?

9 Computational properties of matrices

Show that the product of any two matrices of the form

$$\begin{pmatrix} . & . & . \\ . & . & . \\ 0 & 0 & 1 \end{pmatrix}$$

(where . can be anything) also has the same form. Of the 2D transformations we've seen, which ones have this form when expressed as 3×3 matrices operating in homogeneous coordinates? How can this property be put to use computationally?

10 Convex hulls and Bézier curves

The *convex hull* of a set of points is the least convex area (in 2D) or volume (in 3D) that includes that set of points. (Clearly it generalises to higher dimensions as well.) The convex hull of a set of 2D points is a convex polygon (as if you wrapped a piece of string around the outside of the points). The convex hull of a set of 3D points is a convex polyhedron.

Any point on a Bézier curve is a weighted combination of the control points, where the weights sum to one. This means that the curve must lie within the convex hull of the control points. Why is this a useful property?

Does the same hold true for a Bézier surface patch? That is, does the patch lie within the convex hull of the 16 3D control points of the patch? If it is true, why would it be a useful property?

11 Joining Bézier curves

Suppose you have two Bézier curves, one defined by control points P_1, P_2, P_3, P_4, and the other defined by P_4, P_5, P_6, P_7. What constraints are needed on these control points in order to ensure C^1 continuity at the join?

12 Piece-wise curves

How would you fill a region defined by a piece-wise curved boundary? That is, like a polygon, but with the boundary edges being curves instead of straight-line segments. Consider what restrictions it might be reasonable to impose.

13 Rotation by shear

Show how a rotation can be accomplished by combining a horizontal shear with a vertical shear. Specifically, show that

$$\begin{aligned} \mathbf{R}_\theta &= \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \\ &= \mathbf{H}_a \mathbf{V}_b \mathbf{H}_a \\ &= \begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ b & 1 \end{pmatrix} \begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix} \end{aligned}$$

for a and b dependent on θ. (Hint: Fully expand the matrix product, identify elements with $\mathbf{R}_\theta$, and do a bit of algebra and trigonometry.) Can you see any computational advantages to calculating rotations this way?

14 Computational efficiency of homogeneous coordinates

Consider the computational efficiency of homogeneous coordinates for transformations and rotations. How do homogeneous coordinates compare to non-homogeneous coordinates for these geometric transformations? What are the advantages/disadvantages of using homogeneous coordinates?

15 Perspective in homogeneous coordinates

Show how perspective projection can be expressed as a matrix operation in homogeneous coordinates, where the 3D point (X, Y, Z) is represented by the homogeneous coordinates (kX, kY, kZ, k), where $k \neq 0$, and is typically 1.

16 Perspective transformations using matrices

This exercise involves a question you may have seen previously, but this time you must formulate your solution in terms of transformation matrices instead of vector geometry.

Suppose you have a 3D world coordinate XYZ system, and suppose the camera (optical centre) is located at (U, V, W) in this system, and that the camera is pointing towards the origin. To fix the roll of the camera, assume that the positive y axis of the image surface points towards the positive Z axis (assume that the image surface is "in front of" the optical centre, so image is "right side up").

Show transformation matrices for the image projection (x, y) of a point (X, Y, Z) in world coordinates under perspective projection. What is it under orthographic projection?

17 Multiple light sources

You have a scene consisting of a single convex polyhedron. You have a single, *distant* point light source, and some ambient illumination. Consider only diffuse (Lambertian) reflection. For a given viewpoint, you want to be able to see quickly the effects of change of light source direction. Show how you could do this using a lookup table.

Could this lookup-table technique be extended to handle multiple light sources? To handle specular reflection? Would this make sense? Could it be extended to handle several objects, or non-convex objects?

18 Difference between Phong and Gouraud

Come up with an example for which Phong shading and Gouraud shading produce very different results.

19 Shading highlights

Describe how you could use *Gouraud shading* with specular reflection (instead of Lambertian reflection) to create shading (lighting) highlights on object surfaces.

Compare the approach to using *Phong shading* with specular reflection as a function of the number of polygons used to represent surfaces and as a function of the amount of computation required.

20 Computational cost of shading

What are the relative computational costs of Gouraud shading as compared to Phong shading? Outline the costs of each of the steps involved in each method and compare their relative computational complexity.

21 Intensity and RGB colour

Suppose the red/green/blue components of a raw colour image (digitised say from a colour video camera) are respectively R, G, and B (at every pixel). From this you can compute the combined intensity

$$I = R + G + B$$

which gives you a gray-scale version of the colour image. The raw RGB values depend on illumination and shading, so *normalised colour* is often used in image analysis:

$$r = R/I \qquad g = G/I \qquad b = B/I$$

the idea being that normalised colour represents the intrinsic surface colour of the objects in the environment, ignoring illumination and shading effects that are assumed to cancelled out on average.

What problems do you see in using normalised colour in a digital image, particularly in the darker parts?

22 Different shading approaches

One way of computing the appearance of coloured objects is to repeat the shading calculations for each primary colour, red, green, blue, separately. (Actually this is only an approximation—you might like to think about situations where it breaks down.) That is, we treat the light sources (point or ambient) as having three RGB components, and the surfaces as having three different reflectivities for each of red, green, blue. (These are the diffuse reflectivities—what about specular reflectivity?) We look at the red component of the incoming light and, using the red reflectivity of the surface, compute the red component of the surface's appearance to the viewer. We repeat this for green and blue to get the RGB components of the final display. Overall, about three times as much calculation as for the no-colour case.

What if we're only interested in producing a gray-shade display? We could add together (or average) the final RGB components at each pixel to get a lumped intensity value. But this is just as much work as the colour case (actually a tiny bit more). One method that suggests itself is to add up the RGB components of the incoming light to get a lumped incoming light intensity, and average together the RGB reflectivities to get a lumped reflectivity. We can then do the calculations as in the no-colour case—only one shading computation per pixel instead of three. Under what conditions will this shortcut produce the same results as the fuller RGB method described in the previous paragraph? All conditions? Or only some?

23 Refraction

Snell's Law says that

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{n_2}{n_1}$$

for a ray of light going from a medium with refractive index n_1 to a medium with refractive index n_2, where θ_1 and θ_2 are the respective angles the ray makes with the surface normal at the surface. (See Section 14.5.2 of textbook.) The refractive index of air (or vacuum) is 1.0; for most transparent materials (water, glass, crystal, gems) the refractive index is in the range 1.3 to 1.5, roughly speaking. (Diamond has a refractive index of 2.4, partly explaining its sparkle.)

Show that for a plate of refractive material with parallel faces (like glass in air), the ray of light leaving the plate has the same direction as the incoming ray. By how much is the outgoing light ray displaced perpendicular to the ray's direction?

Does computation of refraction necessarily involve calls on trigonometric functions? Explain.

24 Hidden surface removal

In the scanline method for hidden-surface removal, linear interpolation in depth on each polygonal face can be used between the x intercept points along each scan line. (Why is this a good thing?) However, this is only exactly true for parallel projection. Show a case, under perspective projection, for which equal steps in image x along a scan line do not lead to equal steps in depth Z along a polygonal face. Does this make any difference for hidden-surface elimination?

(a) Discuss and compare the advantages of the rendering pipeline approach using z-buffering over the real-time ray-tracing approach to computer graphics for scenes involving

- fewer polygons than pixels in the projected image, and
- more polygons than pixels in the displayed image.

(b) Can you think of reasonable or unreasonable cases in which Z buffering with inadequate depth resolution may give the "wrong" answer for hidden surfaces? Can you think of any rule of thumb for deciding what fineness of depth resolution is adequate for "reasonable" scenes? (You'll also need to think about what "reasonable" might mean.).

discuss them.

25 Animation and kinematics

(a) For animation describe the difference between *forward* kinematics and *inverse (reverse)* kinematics.

(b) For what kinds of scene graphs would you prefer to use *forward* kinematics over *inverse* kinematics for animation? When would you prefer *inverse* kinematics over *forward* kinematics? Explain you choices.

(c) Name one technique or algorithm that you would use to solve *forward* kinematics and another you would use to solve *inverse* kinematics. Under what conditions would use each of these techniques or algorithms and why?