1. (a) What is a *mesh model* and state briefly one advantage and one disadvantage of its use for surface modelling. [3]

A data structure representing a polygonal mesh (usually a triangular mesh) which is a piecewise planer approximation to the surface of the 3D shape being modelled.

Advantage: a near-universal representation, with hardware support for fast rendering.

Disadvantage: inherently approximate, so requiring a large number of polygons to reduce the error.

(b) What is *hidden surface removal* and why is it necessary? Explain the *z-buffer* method of hidden surface removal. Under what conditions is the *scan-line* method more appropriate? [9]

HSR: the removal of those parts of a model which cannot be seen in the view being rendered. It is necessary to reduce the rendering time, which would otherwise be taken up with parts of the model which will not be seen because they are obscured by other parts.

Z-buffer: two buffers of identical spatial resolution are used, one for the picture (colour values) and one for the depth (z values). The latter is initialised to its largest value ('infinity'). Each model facet (e.g. triangle) can be rendered independently. When a facet is rendered, produce a depth value and a colour value for each pixel. Check the depth value against the value in the z-buffer. If it is nearer than the z-buffer value, then update both the z-buffer value and the colour value in the two buffers, with the values just calculated. Otherwise discard both and move on to the next pixel. Easily implemented in hardware.

Scan-line: works well at very high resolution or when there is high overlap. Also works well if memory is limited, though this is less of an issue these days.

(c) Derive the simple perspective transform, as used in most rendering applications. Indicate why *3D clipping* is needed in a practical renderer. [4]

Perspective transform: direct from notes. Essentially divided-by-depth.

3D clipping: removal of those parts of the model which are outside the view frustum, including behind the viewer. Desirable because most of the model may in fact be outside the viewing frustum, especially for large VR applications. In turn this allows faster rendering, possibly in real-time.

Explain carefully why we use:

(i) matrices (ii) homogeneous coordinates

when representing transforms. [4]

(i) Matrices provide a uniform representation of a class of useful transforms, which can then be combined by matrix multiplication.

(ii) Homogeneous coordinates permit additive transforms (translation), as well as multiplicative ones (such as rotation). Moreover they permit projection (perspective).

2. What is the *Nyquist limit* and what does it tell us about adequately representing a picture by pixels? How does it apply in time, when representing a moving scene? [6]

The sampling theorem tells us that we must sample at twice the frequency (2f) of the highest frequency Fourier component (f) we wish to retain. f is called the Nyquist limit.

This tells us that the pixel spacing determines the highest spatial frequency we can reproduce; and that we should remove any higher frequencies before sampling a continuous image; or we should ensure that we do not generate any higher frequencies when synthesising a picture.

In time, the frame-rate is the temporal sampling rate. We should therefore ensure that there are no temporal frequencies which exceed half of this value.

What is *aliasing* and how does it relate to the Nyquist limit? [4]

Aliasing is the appearance of energy from high frequencies as low frequencies. That is, the high frequencies appear disguised as low frequencies, hence under the alias.

If we fail to sample correctly in space, we may see Moire patterns or similar problems as spatial aliasing.

If we fail to do sample correctly in time, then we may see a stationary wheel, or one revolving backwards, when in fact it is revolving forwards. These are temporal aliases.

Describe in detail Cook's method of stochastic sampling a moving scene, carefully explaining both what it achieves and the trade-offs it makes. [10]

[Detailed explanation, as given in the notes.]

3. Briefly explain the *ray-tracing* method of rendering. [2]

Trace a ray backwards from the eye, through each pixel on the screen to determine which part of the model it hits. Calculate the surface normal at that point. Apply the lighting calculation to get the erquired pixel value.

With the aid of a diagram, derive the intersections of a ray from vector $\mathbf{s}$ in the direction of unit vector $\mathbf{d}$ with a sphere of radius r, centre at vector $\mathbf{c}$. [5]

[Derivation as in the notes.]

What is *constructive solid geometry* (CSG)? Carefully explain how a CSG tree is ray-traced. [8]

CSG: use mathematically-rigorous definitions of solid primitives (e.g. sphere, cone, torus, cube, cylinder). Use Boolean combinations to mae moer complex objects, hierarchically (typically using a binary tree structure to represent the object).

CSG ray-trace: [from notes]

If the model contains mirrors, how is their effect ray-traced, what complications do they add and how are these complications controlled in practice? [5]

Method: use the surface normal to determine the direction of reflection. Spawn a new ray in that direction and treat its result (colour) as the colour for the original ray. Can dimish the contribution slightly (allowing for physical losses) and/or colour it (to allow for coloured mirrors).

Complication: potentially unlimited reflections. Bound these either arbitrarily (e.g. permit no moer than 10 reflections for a given ray); or physically (reduce the contribution at each reflection and give up when it gets too low to see).

4. Briefly describe how the eye's cone cells allow us to see colour. Why is it necessary to include negative values for RGB tri-stimulus coordinates? [4]

Three cones, with a spread of response centred on different parts of the spectrum. The brain combines these to give a full colour picture.

The three responses are not independent, they partially overlap. The RGB model assumes independent coordinates. Hence colour matching is not always possible because setting (for example) G to just the right value may also add some R and B to what is being viewed. This can be overcome by providing a colour bias to the sample, then removing that bias when a match has been achieved. This can result in a negative coordinate because the bias is greater than the match value.

With the aid of sketches of the CIE chromaticity diagram, explain each of the following facts.

(i) No colour monitor can display all the colours we can see.

Three CIE values define a triangle entirely within the CIE 'horseshoe'. There will always be values outside the triangle which cannot be display.

(ii) Even a good colour printer may not cope with all the colours on a monitor.

The triangle/polygon for a printer may not cover the triangle for the monitor.

(iii) There are visible colours which are not spectral colours or desaturated spectral colours.

Purples (the straight line edge of the horseshoe; and their desaturated variants.

[6]

Show how, given an RGB colour, its HSV equivalent is computed. Why is the HSV colour model thought to be more intuitive than RGB? [5]

Conversion: [notes]

HSV coordinates mean something to a viewer, because they relate to our everyday usage of colour much better than RGB.

You are given two distinct colours and asked to produce a range of colours by linearly interpolating between them. Carefully explain why this process produces a different range in RGB space to that in HSV space. Use diagrams if it assists your explanation. [5]

[Notes. Roughly, a straight line in one colour space does not cover the same colours as a straigh line in another space, even if the two end points are identical. Linear interpolation thus produces different results.]