Section B

1)

1. Ways that we approximate :  
   First way : real values can only be stored to using a finite number of bits. A lot of algorithms in modelling and rendering work over rounding up floating points, they are so called float points algorithms. When rounding float points it loses valuable information which otherwise could be used. E.g. Z-buffer z-fighting, hidden surface removal, scan converting algorithms.   
   Second way : simplifying the real world for the sake of efficiency and practicality. Examples from rendering include : the effect of distance travelled on light intensity, and the effect of wavelength and refractive index of a material on the intensity of specularly reflected light.
2. Scan conversion: process of finding screen pixels to insert a polygon.   
   To efficiently scan-convert a triangle we use the sweep-line algorithms:  
   We start from top edge, working down to bottom, scanline by scanline, finding the start and end of each scanline inside the triangle and fill the pixel inside that scanline. It is a floating point algorithms that round pixel grid.
3. Z-buffer, the z-buffer keeps the depth of each pixel stored in its data structure.   
   Works by initialising every z-buffer entry to maxdepth.   
   Then for each pixel generated during the scan-conversion of an object  
   If its z-coordinates of it less than the corresponding entry in the z-buffer:  
   We update the z-buffer with the new pixel coordinates, also we compute the colour of that pixel and store it.  
   Otherwise we don’t do anything because that pixel is blocked by another pixel.
4. i) Ambient illumination: the general level of illumination caused by multiple reflections in the scene.  
   Can be computed with   
   Ambient illumination = ia\* ka   
   Ia is the intensity of ambient light  
   Ka is the ambient reflection coefficient, a number between 0 and 1  
   ii) Diffuse reflection: the absorption and uniform re-radiation of light. Some wavelengths are absorbed while other are reflected.   
   Idiffuse  = ipkd cos theta  
   i diffuse = ipkd (N.L)  
   Where ip is the light source intensity   
   And kd is the diffuse coefficient, a number between 0 and 1.  
   Theta is the the angle of incidence   
   N is the surface normal  
   And L is the direction of light source (normalised vector)  
   Can introduce the effect of distance too it so it becomes  
   Idiffuse = ip/d’ \* kd(N.L)   
   Where d’ distance the light traveled  
   And d’ = kc + kld + kqd2   
   All ks are lighting coefficient, constant, linear and quadratic coefficient  
   iii) specular reflection: reflection at the air-surface interface. This reflects the colour of light source. The Fresnel equation takes into account both the refractive index of the material and the wavelength of the light, but this is computer graphics … you can’t handle the truth! So approximate the result of the Fresnel equation with a single constant (the specular constant coefficient) Ks. The effect of the angle between the direction of maximum specular reflection R and the direction of of the viewer is captured using Phong’s specular function for observed specular reflection.   
   Specular reflection = ip/d \* ks(R.V)n   
   R is the reflect outbound angle   
   V is the viewer position   
   To incorporate lights we simply provide that light colour variables in the inputs
5. Local illumination: we treat each object in the scene separately. It’s limitation that it doesn’t model interaction between different objects.

2-

1. It will make a complex formula required for each type of matrix transformation.   
   Cannot capture translations ... try it!
2. By having a homogenous matrix system that is 4x4 we just need one set of operations.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 0 | tx |
| 0 | 1 | 0 | ty |
| 0 | 0 | 1 | tz |
| 0 | 0 | 0 | 1 |

1. The camera: are entities used to capture a scene, have a position, orientation and lens type. The position is determined by eye point, and orientation is determined by center of interest and the normalised vector.  
   Eye point matrix specify where the camera is located in 3d space, center matrix specify the 3d point where the center of interest is, the view matrix specify the up direction of the camera.   
   i) It is impossible to tell if the camera moved or the object has moved. (draw some diagram that shows a box moved 2 meters away from camera, and one time the camera moved 2 meters away, both cases are the same)   
   ii) Camera’s coordinate system: the position of the camera is determined by the eye point, orientation is determined by the center of interest and its up is determined by its view vector.   
   iii) perspective projection: The objects further away from the center of projections looks smaller and its angles and edges may look distorted, and closer one looks bigger. There are three types of classical projection, 1 point, 2 point and 3 point projections.  
   iv) the near and far clip plane: The planes that define the view volume for a projection. Anything in front of near clip plane or behind the far clip plane, is clipped out and therefore it is invisible.
2. 1- We translate B to pass through the origins, creating matrix M1.  
   2- We rotate matrix M1 about its X axis, mapping it into the XY plane, creating matrix M2.   
   3- We rotate Matrix M2 about its Z matrix, mapping it into the X plane, creating matrix M3.  
   3- We scale the matrix M3 by (s, t, u, 1)   
   sx , ty, uz. Creating matrix M4.   
   Then we apply the inverse matrix of M3, M2 and M1   
   Thus creating:   
   P’ = M1-1 . M2-1. M3-1 . M4. M3. M2.M1. P

==============

Section C

1. That means the features of my face could uniquely identify me in between numbers of people.   
   The eyes, especially the retina is unique between humans hence could be a very strong feature to have.  
   Other features such as combination of size of eyes, colour, nose size, jaw size, ears etc… can be used as extra features to further enhance the system.
2. Blob labelling.   
   By identifying the area and perimeter of those features and combination of those those could be used to uniquely identify the person.
3. By using the area, perimeter and colour distribution of those features.  
   E.g. can derive the area by using chain code analysis on the blobs:  
   Explain the chain code area and perimeter here.
4. Depending on the way they are facing could hide some important features and makes for example chain code hard to derive (since it is orientation dependent)   
   If the center of person’s face isn’t pointing toward the camera, and/or the eye radius wasn’t detected then alarm them to face you.
5. I would first train the system on various subjects, then do a cross validation test where subject of similar and different facial characteristic use the system. Having a confusion matrix will help identify where the problem lies.

4-

FK ME