CPSC 314
Midterm 2

March 19, 2018

Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page.

Name: Solutions

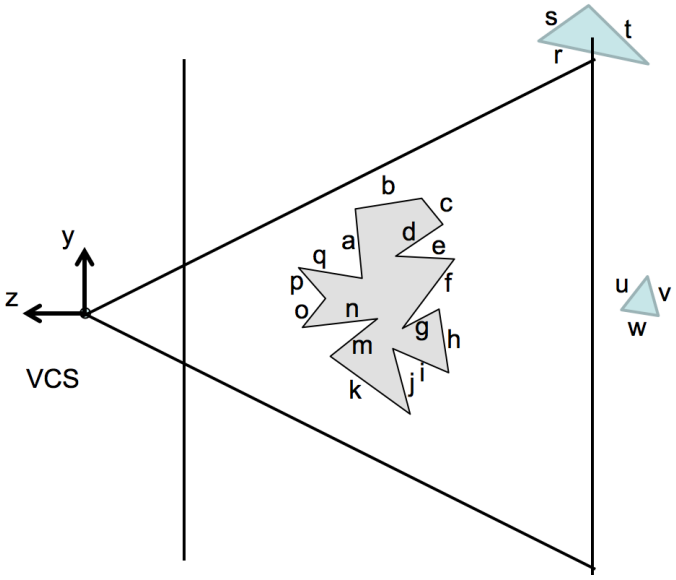
Student Number: _____

Question 1	/ 6
Question 2	/ 6
Question 3	/ 6
Question 4	/ 6
Question 5	/ 8
TOTAL	/ 33

This midterm has 5 questions, for a total of 33 points.

1. Culling

   Assume that the object below is solid, and has a set of labeled faces, as shown below.



   (a) (2 points) In alphabetical order, list the faces that would be removed by back-face culling. Do not take possible view-frustum culling into account.

   b, c, d, e, f, h, j, n, q, t, v

   (b) (2 points) In alphabetical order, list the faces that would be removed by view-frustum culling. Do not take possible back-face culling into account.

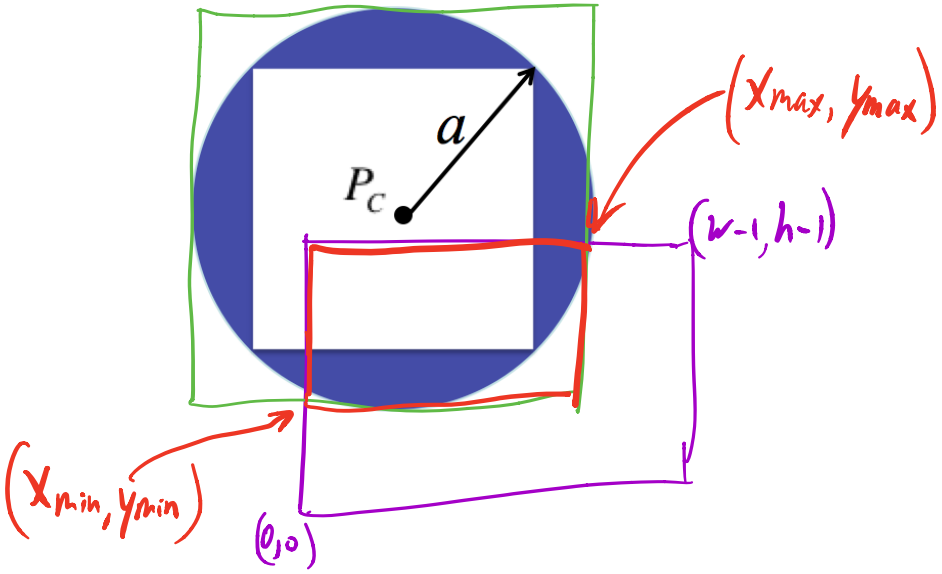
   s, u, v, w

   (c) (2 points) After view-frustum culling and back-face culling, which remaining faces would be completely removed by z-buffer tests? List these in alphabetical order.

   g, i

2. (6 points) Scan Conversion

Write pseudocode for scan converting the shape as illustrated below, i.e., a circle with a square taken out of it. The shape is centred at location $P_C(x_C, y_C)$. Use implicit equations to develop your solution. Assume that a pixel is set on using `Image[x][y]=1`, and that the image spans from `(0,0)` in the bottom left to `(w-1,h-1)` in the top right, giving an image of $w \times h$ pixels. If part of the shape is located off-screen, the remainder should still be properly rendered.

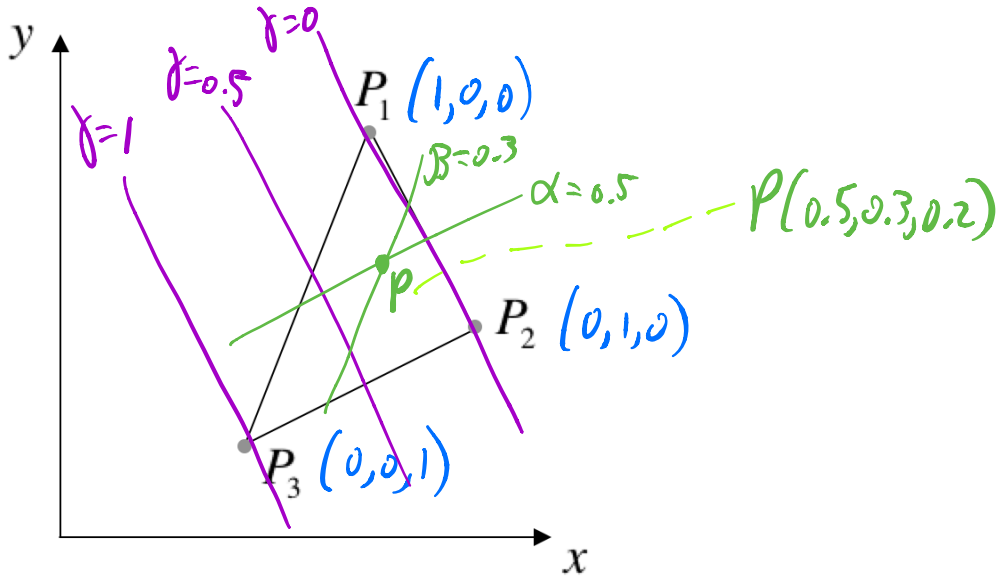


$x_{min} = \max(0, x_c - a)$
$x_{max} = \min(w-1, x_c + a)$
$y_{min} = \max(0, y_c - a)$
$y_{max} = \min(h-1, y_c + a)$

```
for (x = x_min; x ≤ x_max; x++)
    for (y = y_min; y ≤ y_max; y++)
        C1 = a^2 - (x - xc)^2 - (y - yc)^2   // inside circle
        C2 = |x - xc| - a/√2                 // outside square in x
        C3 = |y - yc| - a/√2                 // outside square in y
        if (C1 ≥ 0 and (C2 ≥ 0 or C3 ≥ 0))
            Image[x][y] = 1
```

3. Barycentric Coordinates

Assume that the barycentric coordinates are defined according to $P = \alpha P_1 + \beta P_2 + \gamma P_3$.



(a) (4 points) On the diagram above, label the vertices with their barycentric coordinates, (α, β, γ). Then sketch the three lines that correspond to $\gamma = 0, \gamma = 0.5, \gamma = 1$.

(b) (1 point) Sketch the point P that corresponds to $\alpha = 0.5, \beta = 0.3$. $\gamma = 0.2$

(c) (1 point) Use barycentric interpolation to compute a value v at the point above, i.e., defined by $\alpha = 0.5, \beta = 0.3$, if the value of this quantity at the three vertices is given by: $v_1 = 1$, $v_2 = 2$, $v_3 = 3$.

$$v = \alpha v_1 + \beta v_2 + \gamma v_3 = (0.5)1 + (0.3)2 + (0.2)3$$
$$= 0.5 + 0.6 + 0.6 = \boxed{1.7}$$

(d) (1 point) Now compute the *perspective correct* interpolated value of v for the same point above. Assume the following h-values for the vertices: $h_1 = 1, h_2 = 1, h_3 = 2$. Note: Perspective-correct interpolation is determined by $v = \frac{B(v/h)}{B(1/h)}$, where $B()$ performs standard barycentric interpolation of its argument. You need not do the math to simplify your final answer.

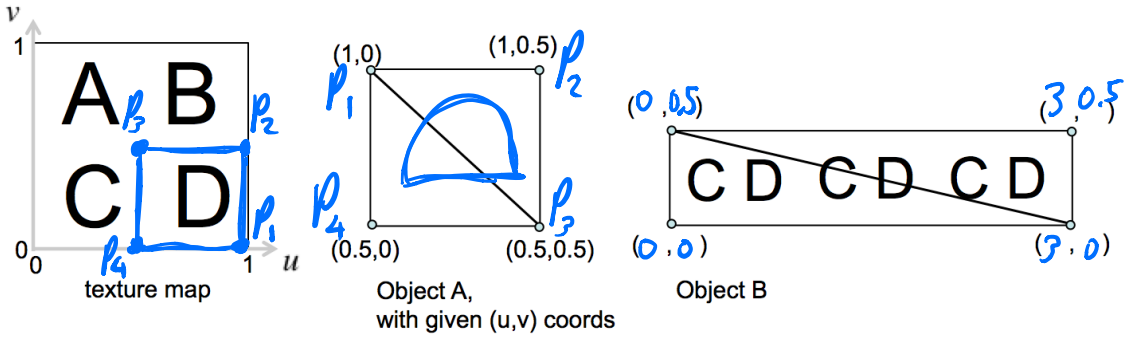
$$\frac{v}{h} = \alpha \frac{v_1}{h_1} + \beta \frac{v_2}{h_2} + \gamma \frac{v_3}{h_3} = (0.5)\frac{1}{1} + (0.3)\frac{2}{1} + (0.2)\frac{3}{2} = 0.5 + 0.6 + 0.3 = 1.4$$

$$\frac{1}{h} = \alpha \frac{1}{h_1} + \beta \frac{1}{h_2} + \gamma \frac{1}{h_3} = \frac{0.5}{1} + \frac{0.3}{1} + \frac{0.2}{2} = 0.9$$

$$v = \frac{1.4}{0.9} = \boxed{1.556}$$

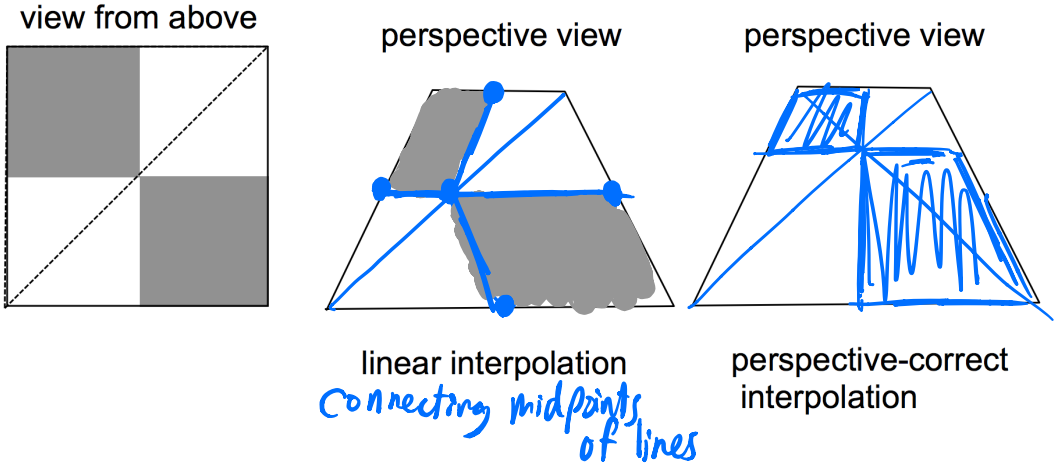
4. Texture Mapping

(a) (4 points) Consider the texture map below, which is to be mapped to Objects A and B. Assume that the RepeatWrapping texture mode is being used.



(i) In the Object A diagram above, sketch the image that would appear for Object A for the assigned texture coordinates.
(ii) In the Object B diagram above, assign texture coordinates to Object B so that it would yield the given image.

(b) (2 points) A square is texture mapped with a checkerboard texture, and rendered using two triangles, as shown below. Sketch what the texure would look like for (i) linearly-interpolated texture coordinates, and (ii) perspective-correct interpolated texture coordinates.



5. Short Answer

(a) *(2 points)* *Overdraw* refers to the fact that the same display pixel may be redrawn multiple times when multiple different triangles, at different distances, may cover a given pixel. If the cost of shading a fragment is high, i.e., because of a complex calculation required in the fragment shader, then overdraw can be bad because a resulting colour will be computed at high cost, only to later be overwritten by a closer object that covers the same pixel. Describe what might be done in order to minimize overdraw for geometry that has costly shading computations.

Either: (a) Render scene from front-to-back as much as possible, to allow z-buffer test to fail as often as possible, thereby avoiding fragment shader calls.
(b) Do a z-buffer-only pre-render, then the real render.

(b) (1 point) Guest lecture: State machines are commonly used to develop and control animated characters in games. Why can these become problematic to use?

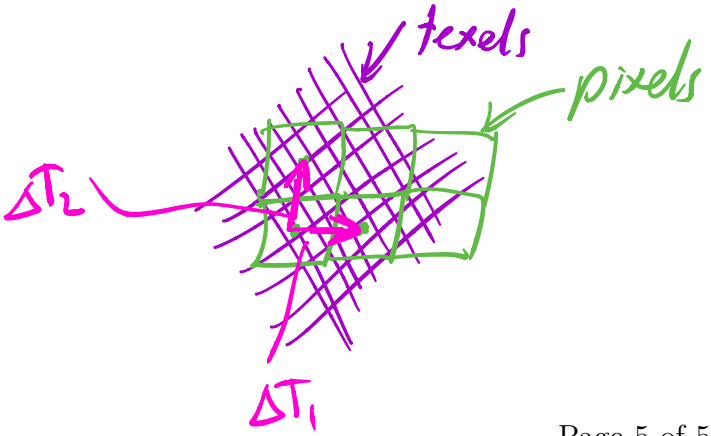
They quickly become very complex and need to be constructed and maintained by hand, as compared to other more recent methods, e.g. motion matching

(c) (1 point) Clipping can be computed in NDCS. Circle one: `True` `False`

(d) (1 point) Back-face culling can be computed in NDCS. Circle one: `True` `False`

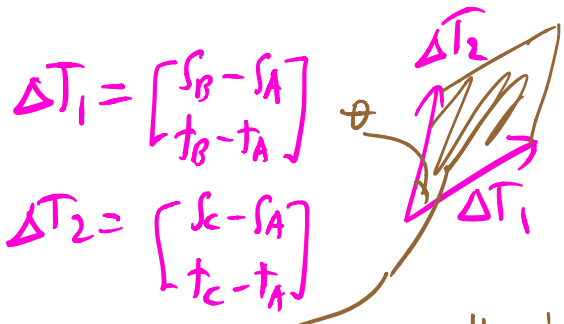
(e) (1 point) Given the (s,t) texture coordinates, a colour-lookup from a MIPMAP can require accessing up to N texels in the pyramid, e.g., using a LinearMipMapLinear lookup. N=__8__

(f) (2 points) Suppose that you have access to the texture coordinates, $T(s,t)$, for the pixel $P_A(x,y)$ and its neighbors, $P_B(x+1,y), P_C(x,y+1)$. How might this be used to compute an estimate of the texel coverage for the pixel $P_A(x,y)$?



$\Delta T_1 = \begin{bmatrix} s_B - s_A \\ t_B - t_A \end{bmatrix}$

$\Delta T_2 = \begin{bmatrix} s_C - s_A \\ t_C - t_A \end{bmatrix}$



$\text{Area} = |\Delta T_1||\Delta T_2| \sin\theta = |T_1 \times T_2|$