CPSC 314, Written Homework 4

Out: Wed 30 Mar 2016
Due: Wed 06 Apr 2016 2pm (hand in at start of lecture)
No homework accepted after Fri 08 Apr 2pm
Value: 4% of final grade
Total Points: 100

Name: ____________________________

Student Number: ____________________________

$Q1$	$/8$
$Q2$	$/6$
$Q3$	$/4$
$Q4$	$/20$
$Q5$	$/24$
$Q6$	$/22$
$Q7$	$/16$
$Total$	$/100$

Please check one of the following:

☐ I did not collaborate with anyone in the completion of this homework.
☐ I collaborated with people named below in the completion of this problem set:

Name: ______________________ Student Number: ______________________

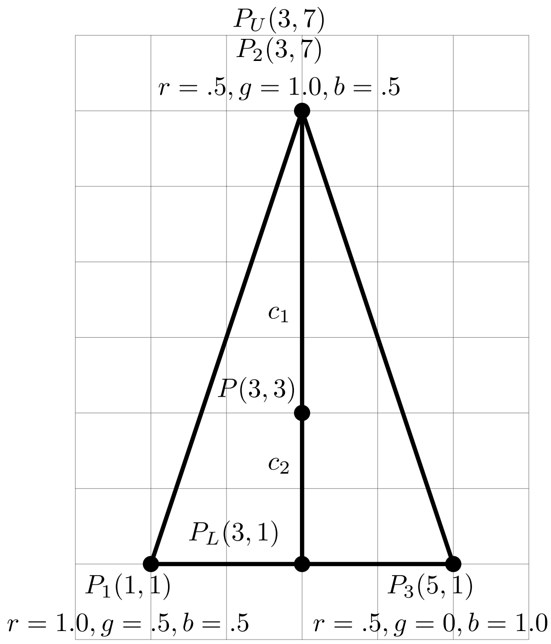
Name: ______________________ Student Number: ______________________

Name: ______________________ Student Number: ______________________

Interpolation (14 pts)

1. (8 pts) For the following triangle, given coordinates and color components of P_1, P_2 and P_3, use bi-linear interpolation to find the (r, g, b) color components at that point P. Show your work.

   Marking Accept answers for both vertical scan line and horizontal scan line, though the later one is normally implemented. Three points for each intermediate point (P_L/P_R or P_U/P_L). Two points for final result.



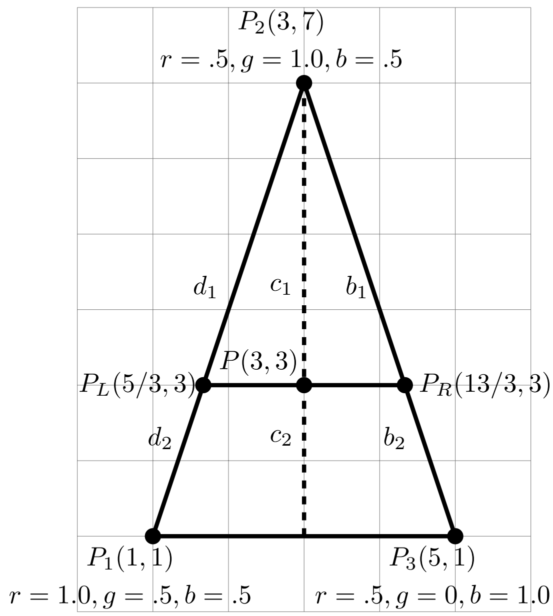
Solution

Using a vertical scan line of bi-linear interpolation, we have:

$P_{rgb} = \frac{c_2}{c_1+c_2} \cdot P_{Urgb} + \frac{c_1}{c_1+c_2} \cdot P_{Lrgb}$

Since P_L lies on the center between P_1 and P_2, it is simple to its compute color component using $P_{Lrgb} = (P_{1rgb} + P_{3rgb})/2$. P_U has the same color compoment as P_2.

$$
\begin{aligned}
P_{rgb} &= \frac{1}{3} \cdot P_{Urgb} + \frac{2}{3} \cdot P_{Lrgb} \\
&= \frac{1}{3} \cdot (.5, 1, .5) + \frac{2}{3} \cdot (.75, .25, .75) \\
&= (\frac{2}{3}, \frac{1}{2}, \frac{2}{3}) \\
&= (.667, .5, .667)
\end{aligned}
$$



Or using a horizontal scan line of bi-linear interpolation, we compute (r,g,b) for point P. By inspection, we see that P is the center point of $P_L P_R$, so we have:

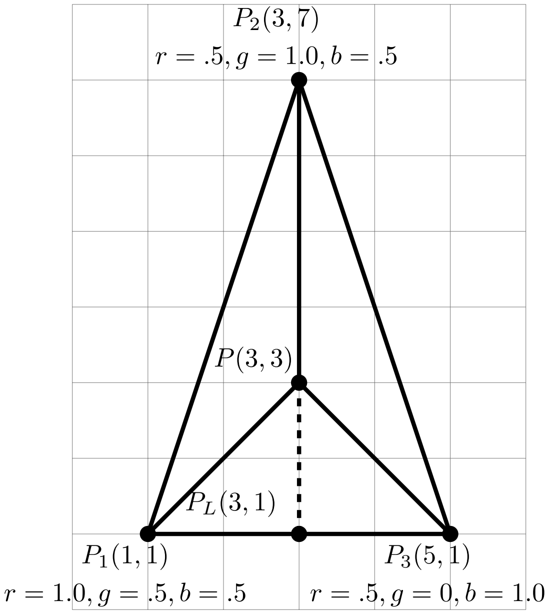
$P_{rgb} = \frac{1}{2} \cdot P_{Lrgb} + \frac{1}{2} \cdot P_{Rrgb}$

To compute P_{Lrgb} and P_{Rrgb}, we need to know the ratio $d1 : d2$ and $b1 : b2$. Due to the similarity of triangles, we see $d1 : d2 = b1 : b2 = c1 : c2 = 2 : 1$, so we have:

$$
\begin{aligned}
P_{Lrgb} &= \frac{d2}{d1+d2} \cdot P_{2rgb} + \frac{d1}{d1+d2} \cdot P_{2rgb} \\
&= \frac{1}{3} \cdot (.5, 1, .5) + \frac{2}{3} \cdot (1, .5, .5) \\
&= (5/6, 2/3, 1/2) \\
&= (.83, .67, .5) \\
P_{Rrgb} &= \frac{b2}{b1+b2} \cdot P_{2rgb} + \frac{b1}{b1+b2} \cdot P_{3rgb} \\
&= \frac{1}{3} \cdot (.5, 1, .5) + \frac{2}{3} \cdot (.5, 0, 1) \\
&= (1/2, 1/3, 5/6) \\
&= (.5, .33, .83) \\
P_{rgb} &= \frac{1}{2} \cdot P_{Lrgb} + \frac{1}{2} \cdot P_{Rrgb} \\
&= \frac{1}{2} \cdot (5/6, 2/3, 1/2) + \frac{1}{2} \cdot (1/2, 1/3, 5/6) \\
&= (2/3, 1/2, 2/3) \\
&= (.677, .5, .677)
\end{aligned}
$$

2. (6 pts) For the same triangle, find the barycentric coordinates α, β, and γ for P, and use them to interpolate the (r, g, b) color components at that point. Show your work.

Marking 1.5 points for each α, β, and γ. 1.5 points for final result.



Solution

The line between P_2 and P_L cuts the triangle into two pieces with same area. By inspection, we see that $P_2P : PP_L = 2 : 1$. Therefore, $A_{P1} = A_{P3} = \frac{1}{2}A \cdot \frac{2}{3}$, $A_{P2} = \frac{1}{2}A \cdot \frac{1}{3} \cdot 2$, where A is the area of the triangle formed by P_1, P_2, P_3. We calculate the barycentric coordinates as follows:

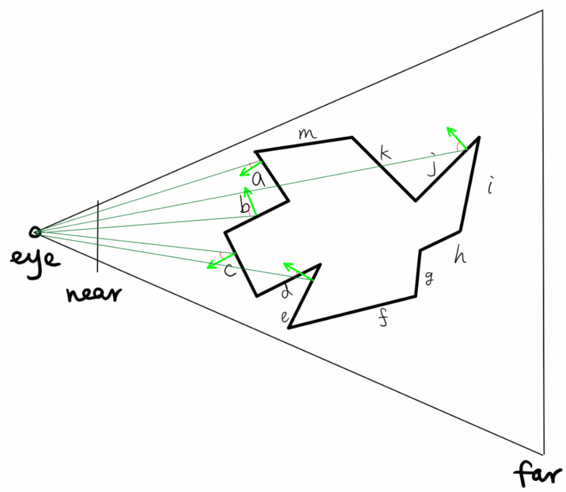
$$\alpha = A_{P1}/A = 1/3$$
$$\beta = A_{P2}/A = 1/3$$
$$\gamma = A_{P3}/A = 1/3$$

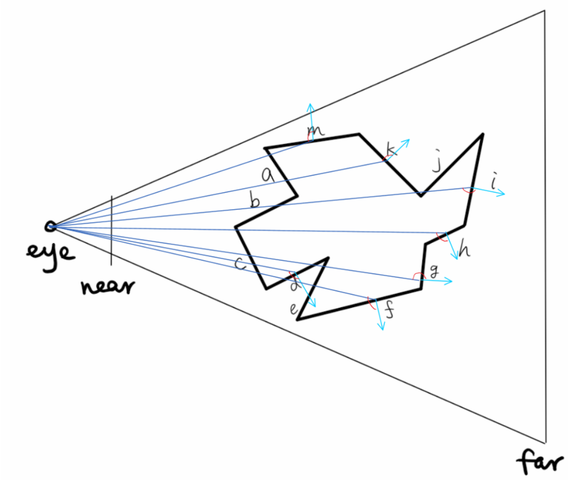
Using these, we calculate the colour components at P:

$$\begin{aligned} P_{rgb} &= \alpha \cdot P1_{rgb} + \beta \cdot P2_{rgb} + \gamma \cdot P3_{rgb} \\ &= 1/3 \cdot (1, .5, .5) + 1/3 \cdot (.5, 1, .5) + 1/3 \cdot (.5, 0, 1) \\ &= (2/3, 1/2, 2/3) \\ &= (.667, .5, .667) \end{aligned}$$

Visibility

3. (4 pts) For the following 2D scene, an eye point is given with respect to an object formed by line segments. Which faces would be removed for the given eyepoint if backface culling were used? Show your work by drawing in the normals for each face.





Solution

VCS checks not as simple as $N_v > 0$, must take eyepoint into account given perspective warping. Blue: angle between normal and eyepoint-to-plane line greater than 90 degree. Green: angle between normal and eyepoint-to-plane line less than 90 degree. Cull if angle is greater than 90 degree.
Cull: d, f, g, h, i, k and m. 0.5 point deduction for each missing/incorrect face.

4. (20 pts) You have bought a very cheap graphics card, which has a Z buffer of only 3 bits. You can thus only determine the visibility relationships of objects in your scene at a very coarse resolution: there are only $2^3 = 8$ bins available. These bins are represented as the base-10 integers 0 through 7. You should assume that the general GL perspective matrix was used for projection, with the near plane set to -0.1 and the far plane set to -80.

a) (8 pts) Give the z-values of the planes forming the boundaries of these bins in DCS, the display coordinate system (which ranges from 0.0 at the near plane to 1.0 at the far plane). That is, what is the value of the plane between bin 0 and bin 1, between bin 1 and bin 2, and so on.

b) (8 pts) Give the z-values of the planes in the camera coordinate system.

c) (4 pts) Explain why this graphic card might be very bad in terms of depth test.

marking
a) 1 pt deduction for each incorrect number; 4 pts deduction if set far to be 8 rather than 1;

b) 2 pt deduction for each incorrect derivation for equations; 1 pt deduction for sign error; 1 pt deduction for each incorrect number in final result, up to 3 pts.

Solution

a)

Plane	z **value**
near plane	0
between bin 0 and 1	.125
between bin 1 and 2	.25
between bin 2 and 3	.375
between bin 3 and 4	.5
between bin 4 and 5	.625
between bin 5 and 6	.75
between bin 6 and 7	.875
far plane	1

b) To solve this problem, we will need the following equations:

$$Z_{DCS} = 1 \ll N \cdot (a + b/z)$$
$$a = \frac{z_{far}}{z_{far} - z_{near}}$$
$$b = \frac{z_{far} \cdot z_{near}}{z_{near} - z_{far}}$$

Here N, the number of bits, is 3, so $1 \ll N = 8$. We can rearrange to solve for z.

$$z = \frac{b}{(Z_{DCS}/8) - a} \tag{1}$$

We need to solve equation 1 for each value of the Z-buffer given $z_{near} = -.1$ and $z_{far} = -80$.

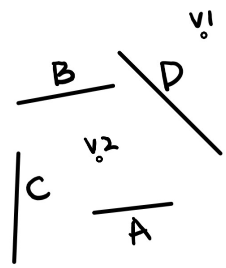
$$a = \frac{-80}{-80 + .1} = 1.0013$$
$$b = \frac{80 \cdot .1}{-.1 + 80} = 0.1001$$
$$z = \frac{0.1001}{(Z_{DCS}/8) - 1.0013}$$

Plugging in values from 0 to 8 for Z_{DCS} produces the following table:

Plane	z **value**
near plane	-0.1
between bin 0 and 1	-.1143
between bin 1 and 2	-.1333
between bin 2 and 3	-.1599
between bin 3 and 4	-.1998
between bin 4 and 5	-.2661
between bin 5 and 6	-.3985
between bin 6 and 7	-.7931
far plane	-80

c) Due to the nonlinearity and lack of precision in the range from -.7931 to -80 in VCS, it can lead to depth fighting for far object.

5. (24 pts) Build a BSP tree for the following scene using the polygons (shown as line segments). The cutting plane induced by a polygon should just extend along the line itself. The labelled side of the polygon should be the right child in the tree, and the unlabelled side should be the left child. If a polygon X gets split, name the split polygons as X_1 and X_2 with X_1 vertically on top of X_2 in the scene.



a) (3 pts) Give the BSP tree with the single root node of polygon A, and sketch the entire scene with the addition of the new cutting plane.

b) (3 pts) Same as above, building on the previous answer, after adding polygon B.

c) (2 pts) Same as above, building on the previous answer, after adding polygon C.

d) (2 pts) Same as above, building on the previous answer, after adding polygon D.

e) (7 pts) Traverse your BSP tree to produce a painter's algorithm ordering from eye point V1. Show your work at each step in the traversal, starting from the root of the BSP tree.

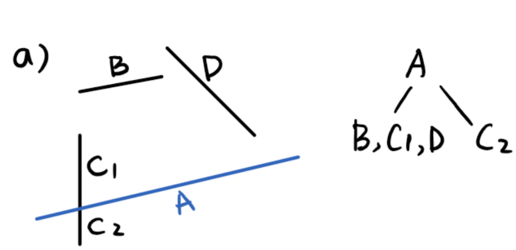
f) (7 pts) Same as above, instead using eye point V2.

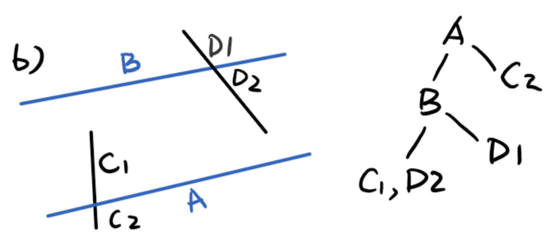
marking

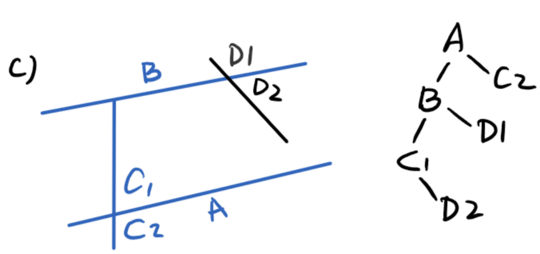
a) - d): One point deduction for each left/right-child swapped. One point deduction for not including entire scene in each tree. Two points deduction if incorrectly separating polygons in each tree. Three points deduction if only showing the final tree.

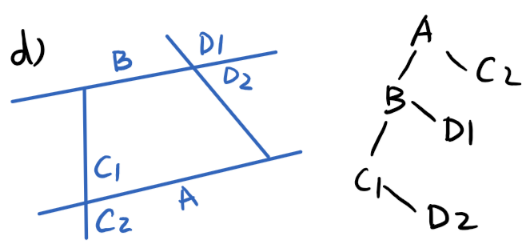
e), f): One point deduction if no order of drawing shown. For solution without intermediate steps, marking just based on how the solution match the expected one.

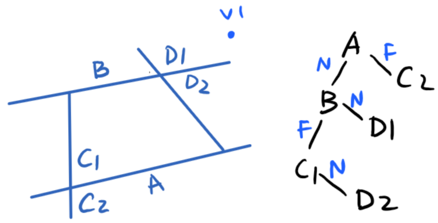
Solution

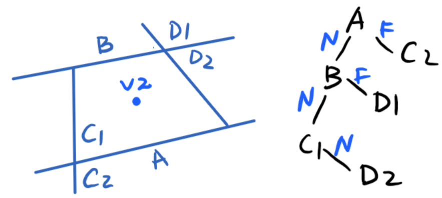












e)

```
BSPtree(A)
        v1 is on left-hand side of A plane:
                near = A->left; far = A->right;
        BSPtree(far): C2
                v1 is on the left-hand side of C2:
                        near = C2->left; far = C2->right;
                BSPtree(far): empty, back
                draw C2
                BSPtree(near): empty, back
        draw A
        BSPtree(A->left): B
                v1 is on right-hand side of B:
                        near = B->right; far = B->left;
                BSPtree(far): C1
                        v1 is on right-hand side of C1:
                                near = C1->right; far = C1->left;
                        BSPtree(far): empty, back
                        draw C1
                        BSPtree(near): D2
                                v1 is on right-hand side of D2:
                                        near = D2->right; far = D2->left;
                                BSPtree(far): empty, back
                                draw D2
                                BSPtree(near): empty, back
                draw B
                BSPtree(near): D1
                        v1 is on right-hand side of D1:
                                near = D1->right; far = D1->left;
                        BSPtree(far): empty, back
                        draw D1
                        BSPtree(near): empty, back
```

The order of drawing is C2, A, C1, D2, B, D1

f)

```
BSPtree(A)
        v2 is on left-hand side of A:
                near = A->left; far = A->right;
        BSPtree(far): C2
                v2 is on the left-hand side of C2:
                        near = C2->left; far = C2->right;
                BSPtree(far): empty, back
                draw C2
                BSPtree(near): empty, back
        draw A
        BSPtree(near): B
                v2 is on left-hand side of B:
                        near = B->left; far = B->right;
                BSPtree(far): D1
                        v2 is on left-hand side of D1:
                                near = D1->left; far = D1->right;
                        BSPtree(far): empty, back
                        draw D1
                        BSPtree(near): empty, back
```

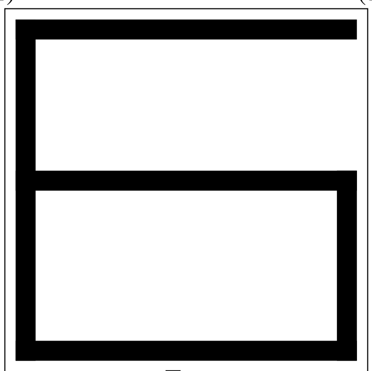
```
draw B
BSPtree(near): C1
        v2 is on right-hand side of C1:
                near = C1->right; far = C1->left;
        BSPtree(far): empty, back
        draw C1
        BSPtree(near): D2
                v2 is on the left-hand side of D2:
                        near = D2->left; far = D2->right;
                BSPtree(far): empty, back
                draw D2
                BSPtree(near): empty, back
```

The order of drawing is C2, A, D1, B, C1, D2

Textures (22 pts)

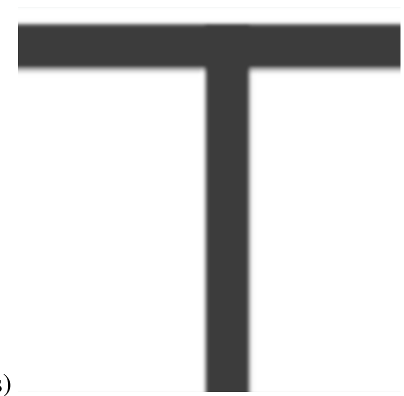
6. Texture Mapping

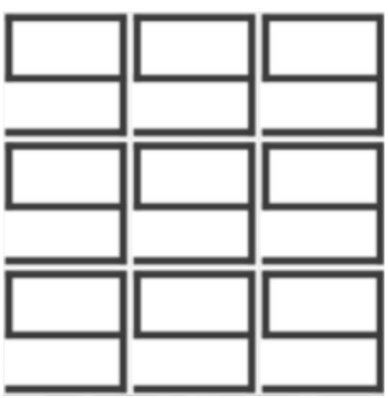
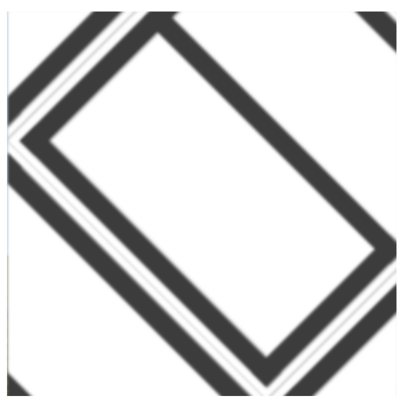
a) (20 pts) In the following figure, sketch the texture (top) as it would appear in each of the rectangles with the specified texture coordinates. Assume the texture mode is gl.REPEAT.



Texture

Marking based on how your drawing match the solution: 1-2 pts deduction if minor mismatch; 3 pts deduction if repeat mode is not considered.

a.(4 pts)  b.(6 pts) 

c.(4 pts)  d.(6 pts) 

b) (2 pts) Storing texture MIPMAP requires 33% additional space.

Ray Tracing (16 pts)

7. A ray $R(t)$ begins at a known eyepoint $E = (1, 2, -1)$, and travels in a direction $V = (0, 0, -1)$ towards a given screen pixel. In the scene, there is a sphere of radius r = 5 centered at the point $(1, -1, -10)$.

   a) (2 pts) Give the parametric equation of the ray $R(t)$ in terms of parameter t.

   b) (8 pts) Decide whether this ray will intersect with the sphere in the scene or not. If intersect, compute the intersection point(s). Show your work.

   c) (6 pts) Suppose we have another ray $R'(t)$ begins at the same eyepoint E, and travels towards another screen pixel in a direction $V' = (3, -3, -13)$. After calculation, you find one of the intersection points with sphere to be $P' = (4, -1, -14)$. Compute the normal N' for the intersection point P'. Then decide whether the point P' is on the backside of the sphere from the eyepoint E.

   Solution

   a)

$$\vec{R}(t) = \vec{E} + \vec{V} \cdot t = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} t \tag{2}$$

No partial credits if incorrect.

b)

Sphere equation (2 pts):

$$(x-1)^2 + (y+1)^2 + (z+10)^2 = 25$$

Plug eqn(2) into the sphere equation and solve the quadratic equation in t to decide whether the ray intersects with the sphere or not (4 pts):

$$\begin{aligned}
(R_x - 1)^2 + (R_y + 1)^2 + (R_z + 10)^2 &= 25 \\
(1-1)^2 + (2+1)^2 + (-1-t+10)^2 &= 25 \\
9 + (9-t)^2 &= 25 \\
(9-t)^2 &= 16 \\
9 - t &= \pm 4 \\
t &= 9 \pm 4
\end{aligned}$$

since t = 5 or 13, there are two intersection points (2 pts):

$$\begin{aligned}
P_1 &= (1, 2, -6) \\
P_2 &= (1, 2, -14)
\end{aligned}$$

c)

First we need to compute the normal of the intersection points (3 pts):

$$\begin{aligned}
F(x, y, z) &= (x-1)^2 + (y+1)^2 + (z+10)^2 - 25 \\
\vec{n}(x, y, z) &= \begin{pmatrix} 2(x-1) \\ 2(y+1) \\ 2(z+10) \end{pmatrix} \\
\vec{N}'(4, -1, -14) &= \begin{pmatrix} 6 \\ 0 \\ -8 \end{pmatrix}
\end{aligned}$$

Normalizing the normals we get (though not necessary in this question):

$$\vec{N'} = \frac{(6, 0, -8)}{\|(6, 0, -8)\|} = \begin{pmatrix} .6 \\ 0 \\ -.8 \end{pmatrix}$$

To decide whether it is on the back (2 pts):

$$\vec{EP} = \frac{E - P'}{\|E - P'\|} = \frac{(1, 2, -1) - (4, -1, -14)}{\|(1, 2, -1) - (4, -1, -14)\|} = \begin{pmatrix} -3/\sqrt{187} \\ 3/\sqrt{187} \\ 13/\sqrt{187} \end{pmatrix} = \begin{pmatrix} -.2194 \\ .2194 \\ .9507 \end{pmatrix}$$

$$\vec{EP} \cdot \vec{N'} = -.8922 < 0$$

The dot product is less than 0 implies that the P' is on the back (1 pt).