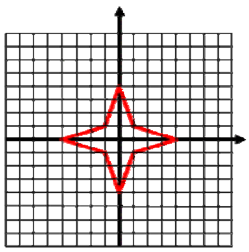
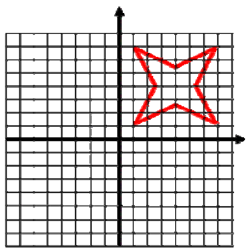
Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Computação Gráfica
AD2 - 1° semestre de 2009.

1) Escreva a matriz homogênea correspondente a composição de transformações abaixo (1.0 ponto):





As operações de transformação necessárias para levar a figura 1 na figura 2 requerem primeiramente a rotação da figura em torno da origem de um ângulo $\alpha = 45^\circ$ no sentido anti-horário, seguida de uma translação dada pelo vetor de translação (4,4). Em forma matricial as matrizes de translação Matriz_A e Matriz_B, em coordenadas homogêneas, conforme descrito abaixo

Dados do problema:

$$\boldsymbol{Matriz_A} = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \text{ e } Matriz_B = \begin{bmatrix} \cos 45^\circ & -sen45^\circ & 0 \\ sen45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A Matriz_C efetua a transformação final e é dada pelo produto Matriz_A x Matriz_B :

$$\boldsymbol{Matriz_C} = \begin{bmatrix} \cos 45^\circ & -\boldsymbol{sen}45^\circ & 4 \\ \boldsymbol{sen}45^\circ & \cos 45^\circ & 4 \\ 0 & 0 & 1 \end{bmatrix}$$

A figura é então finalmente transformada efetuando-se o produto da Matriz_C pelos vetores em coordenadas homogêneas correspondentes a cada vértice da estrela.

2) Seja a matriz

$$T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

representando um movimento rígido do espaço em coordenadas homogêneas. Mostre que a transformação inversa é dada por

$$T^{-1} = \begin{bmatrix} n_x & n_y & n_z & -\langle p, n \rangle \\ o_x & o_y & o_z & -\langle p, o \rangle \\ a_z & a_y & a_z & -\langle p, a \rangle \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dica: as três primeiras colunas de T são formadas por vetores ortonormais (1.0 ponto).

A matriz T pode ser escrita como o produto

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{n}_x & \boldsymbol{o}_x & \boldsymbol{a}_x & \boldsymbol{p}_x \\ \boldsymbol{n}_y & \boldsymbol{o}_y & \boldsymbol{a}_y & \boldsymbol{p}_y \\ \boldsymbol{n}_z & \boldsymbol{o}_z & \boldsymbol{a}_z & \boldsymbol{p}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \boldsymbol{p}_x \\ 0 & 1 & 0 & \boldsymbol{p}_y \\ 0 & 0 & 1 & \boldsymbol{p}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{n}_x & \boldsymbol{o}_x & \boldsymbol{a}_x & 0 \\ \boldsymbol{n}_y & \boldsymbol{o}_y & \boldsymbol{a}_y & 0 \\ \boldsymbol{n}_z & \boldsymbol{o}_z & \boldsymbol{a}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

logo,

$$T^{-1} = \left(\begin{bmatrix} 1 & 0 & 0 & \boldsymbol{p_x} \\ 0 & 1 & 0 & \boldsymbol{p_y} \\ 0 & 0 & 1 & \boldsymbol{p_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{n_x} & \boldsymbol{o_x} & \boldsymbol{a_x} & 0 \\ \boldsymbol{n_y} & \boldsymbol{o_y} & \boldsymbol{a_y} & 0 \\ \boldsymbol{n_z} & \boldsymbol{o_z} & \boldsymbol{a_z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right)^{-1} = \begin{bmatrix} \boldsymbol{n_x} & \boldsymbol{o_x} & \boldsymbol{a_x} & 0 \\ \boldsymbol{n_y} & \boldsymbol{o_y} & \boldsymbol{a_y} & 0 \\ \boldsymbol{n_z} & \boldsymbol{o_z} & \boldsymbol{a_z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & \boldsymbol{p_x} \\ 0 & 1 & 0 & \boldsymbol{p_y} \\ 0 & 0 & 1 & \boldsymbol{p_z} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} =$$

$$\begin{bmatrix} \boldsymbol{n_x} & \boldsymbol{n_y} & \boldsymbol{n_z} & 0 \\ \boldsymbol{o_x} & \boldsymbol{o_y} & \boldsymbol{o_z} & 0 \\ \boldsymbol{a_z} & \boldsymbol{a_y} & \boldsymbol{a_z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\boldsymbol{p_x} \\ 0 & 1 & 0 & -\boldsymbol{p_y} \\ 0 & 0 & 1 & -\boldsymbol{p_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{n_x} & \boldsymbol{n_y} & \boldsymbol{n_z} & -(\boldsymbol{n_x p_x} + \boldsymbol{n_y p_y} + \boldsymbol{n_z p_z}) \\ \boldsymbol{o_x} & \boldsymbol{o_y} & \boldsymbol{o_z} & -(\boldsymbol{o_x p_x} + \boldsymbol{o_y p_y} + \boldsymbol{o_z p_z}) \\ \boldsymbol{a_z} & \boldsymbol{a_y} & \boldsymbol{a_z} & -(\boldsymbol{a_x p_x} + \boldsymbol{a_y p_y} + \boldsymbol{a_z p_z}) \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} \boldsymbol{n_x} & \boldsymbol{n_y} & \boldsymbol{n_z} & -\langle \boldsymbol{p}, \boldsymbol{n} \rangle \\ \boldsymbol{o_x} & \boldsymbol{o_y} & \boldsymbol{o_z} & -\langle \boldsymbol{p}, \boldsymbol{o} \rangle \\ \boldsymbol{a_z} & \boldsymbol{a_y} & \boldsymbol{a_z} & -\langle \boldsymbol{p}, \boldsymbol{a} \rangle \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3) Descreva a principal deficiência da matriz de projeção abaixo. Sugira outra matriz de projeção que corrija tal deficiência (1.0 ponto).

$$T = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

A matriz de projeção

$$\boxed{\boldsymbol{P} = \begin{bmatrix} \boldsymbol{n} & 0 & 0 & 0 \\ 0 & \boldsymbol{n} & 0 & 0 \\ 0 & 0 & \boldsymbol{n} & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}}$$

leva as coordenadas de profundidade de todos os pontos visíveis dentro do volume de visualização no plano *near*, o que faz com que percamos a ordem relativa entre os elementos projetados.

Entretanto, é possível corrigir este problema determinando-se uma matriz tal que a profundidade z''= z'/w, resultante da coordenada projetada z', seja uma função monotônica da profundidade z original do ponto. Para isto precisamos determinar os coeficientes α e β na matriz abaixo, cujo produto da terceira linha pelo vetor de coordenadas especifica a equação à direita, que descreve uma função monotônica (verificar o gráfico da função ou se não há troca de sinais na primeira derivada).

$$\boldsymbol{P}' = \begin{bmatrix} \boldsymbol{n} & 0 & 0 & 0 \\ 0 & \boldsymbol{n} & 0 & 0 \\ 0 & 0 & \alpha & \beta \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

$$z'' = \frac{z'}{w'} = \frac{\alpha z + \beta}{-z}$$

Para determinar α e β lembremos que os valores de profundidade dos pontos no plano *near* e *far* devem ser preservados (-n e –f, respectivamente), assim chegamos ao seguinte sistema:

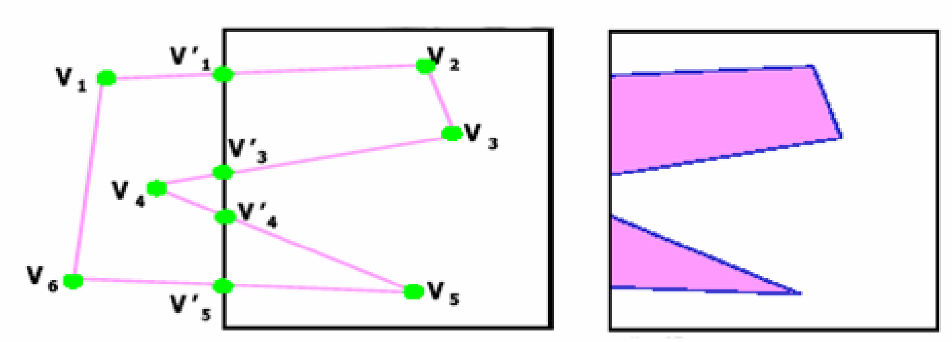
$$\frac{\alpha(-\boldsymbol{n}) + \beta}{-(-\boldsymbol{n})} = -\boldsymbol{n}, \text{ valor de z'' para o plano near}$$
$$\frac{\alpha(-\boldsymbol{f}) + \beta}{-(-\boldsymbol{f})} = -\boldsymbol{f}, \text{ valor de z'' para o plano far}$$

A solução do sistema acima indica que α= n+f e β = nf, donde chegamos à matriz:

$$\boldsymbol{P}' = \begin{bmatrix} \boldsymbol{n} & 0 & 0 & 0 \\ 0 & \boldsymbol{n} & 0 & 0 \\ 0 & 0 & \boldsymbol{n}+\boldsymbol{f} & \boldsymbol{nf} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

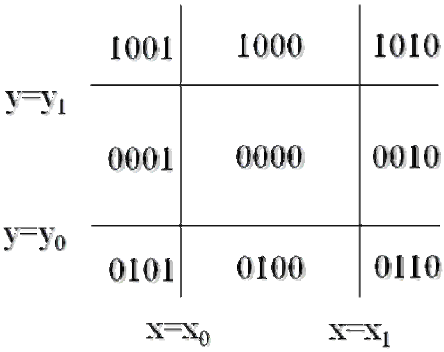
4) Faça uma pesquisa sobre o algoritmo de Weiler-Atherton para recorte de polígonos (1.0 ponto).

O algoritmo de Weiler-Atherton é o mais genérico algoritmo de recorte e se aplica a polígonos côncavos e com furos. Na terminologia do algoritmo de Weiler-Atherton, o polígono a ser recortado é chamado de *polígono-objeto* e a região de recorte é chamada de *polígono de recorte*. As novas arestas do *polígono-objeto*, criadas por seu recorte são idênticas a partes do polígono de recorte. O *clipping* de polígonos de Weiler-Atherton foi desenvolvido para identificação de superfícies visíveis.



A implementação do algoritmo de W-A utiliza uma lista circular de vértices para descrever o *polígono de recorte* e todos os *polígonos-objeto*. Adota-se como convenção, percorrer tais polígonos no sentido horário e os limites internos de objetos com furos, no sentido anti-horário. Desta forma, a superfície ou área cheia do objeto delimitado pelo polígono permanece sempre à direita. As arestas do *polígono de recorte* e dos *polígonos-objeto* podem ou não intersectar. Caso o façam, isto ocorre sempre aos pares, uma de entrada e outra de saída. O algoritmo inicia o processamento em alguma intersecção inicial e segue a borda externa do *polígono-objeto* em sentido horário até encontrar outra interseção. Na interseção, dobra-se à direita e segue-se o exterior do *polígono de recorte* em sentido horário até encontrar uma interseção com o *polígono-objeto*. Nesta interseção, dobra-se novamente à direita e segue-se o *polígono-objeto*. O processo é realizado até que se volte ao ponto de partida. Limites interiores do *polígono-objeto* são seguidos em sentido anti-horário.

5) Escreva um algoritmo que determina a classificação de um ponto, através de um código de 4 bits, com relação às nove regiões determinadas por uma janela retangular (0.5 ponto).




```
unsigned char code(double x, double y,
          double xmin, double xmax, double ymin, double ymax)
{
  unsigned char code=0;

  if (y > ymax) code += 8;
  if (y < ymin) code += 4;
  if (x > xmax) code += 2;
  if (x < xmin) code += 1;

  return code;
}
```

6) Explique detalhadamente a constante especular presente no modelo de iluminação Phong (0.5 ponto).

O coeficiente especular consiste numa constante que servirá de expoente para elevar o produto escalar entre o vetor de reflexo e do observador. Como este produto escalar corresponde ao cosseno do ângulo formado, é um valor que varia entre 0 e 1. Ao elevar por um expoente tendendo a infinito, faz-se que a especularidade apenas exista quando o ângulo tender a zero. Ao elevar-se com expoentes pequenos, esta área tende a ser grande. Assim, pode-se afirmar que é um controle da área de especularidade.

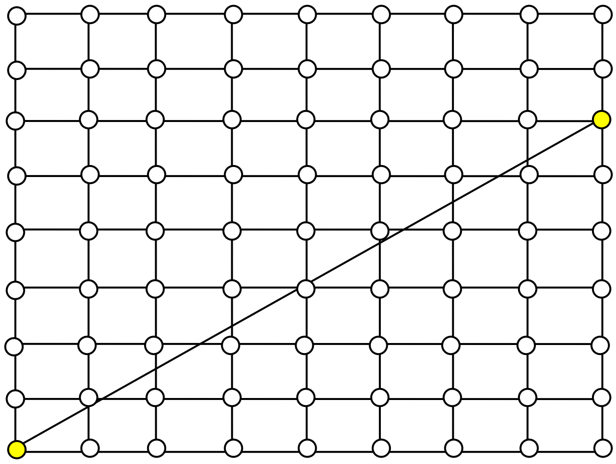
7) Como se poderia resolver de forma mais eficiente que a aproximação do Phong a iluminação ambiente? Porque é inviável para tempo real? (0.5 ponto)

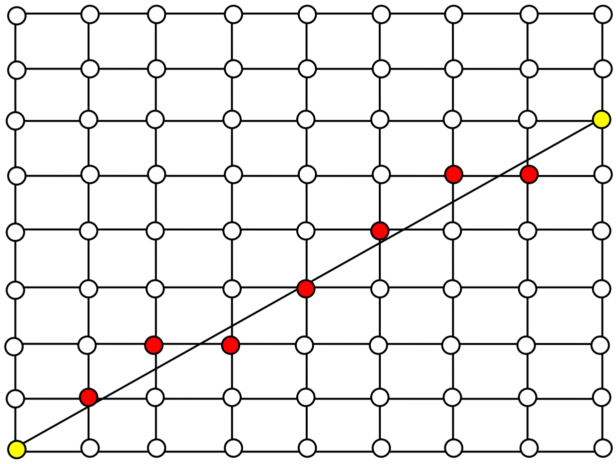
Há várias maneiras de resolver, Uma delas seria utilizar juntamente com o modelo de iluminação Phong o algoritmo de radiosidade, próprio para tratar iluminação ambiente com precisão. Outra possibilidade, muito usada pelos artistas 3D, consiste em utilizar várias luzes auxiliares, com intensidades pequenas e espalhadas pela cena. O problema para tempo real é que o número de fontes de luz encarece o pipeline gráfico.

8) Como o modelo Gouroud trata as normais de uma superficie (1.0 ponto).

O modelo Gouroud realiza uma interpolação entre as normais de polígonos vizinhos. A normal estará presente nos vértices, sendo que ao invés de ter a mesma direção de um polígono, irá possuir uma direção que é a média de todas as normais dos polígonos adjacentes a este vértice.

9) Considerando o algoritmo do ponto-médio para rasterização de retas, determine quais *pixels* devem ser acessos no reticulado abaixo. Considere os *pixels* como sendo os elementos na interseção das retas do reticulado conforme ilustrado na figura abaixo (0.5 ponto).





10) Como se faz para resolver o problema de profundidade na rasterização de polígonos? (Expliquem detalhadamente) (0.5 ponto).

Utiliza-se o algoritmo de Z-Buffer. Este algoritmo consiste em criar uma memória de tamanho equivalente ao frame-buffer. Sempre que um pixel referente a um triângulo for pintado no frame-buffer, será escrito no frame-buffer a profundidade do mesmo. Caso já haja uma profundidade escrita anteriormente no Z-Buffer, antes de pintar o pixel será feita uma consulta se este novo pixel possui profundidade maior ou menor. Caso seja menor, permite-se a sua escrita, em cima do anterior. Neste caso atualiza-se o valor do Z-Buffer com o valor deste novo pixel. Caso o valor de profundidade seja menor, impede-se a escrita deste pixel no frame-buffer, pois já há um pixel mais próximo pintado previamente.

11) Quais são os problemas de aliasing de texturas? (1.0 ponto).

São problemas referentes à amostragem:

- Amostragem de um mesmo texel para vários pixels da imagem
- Amostragem em um pixel da imagem para uma área grande da textura, correspondente a dois ou mais texels.

12) Porque na equação de Ray-tracing, em cada etapa a contribuição de cor vinda da recursão é menor? Porque depois de algumas recursões o efeito passa a ser praticamente imperceptível? (0.5 ponto)

A equação do Ray-tracing é dada por:

Cor = Iluminacao (P) + Kr* RayTracing(Reflexo) + Kt*RayTracing(Transmissao)

Kr é o coeficiente de reflexo, que varia entre 0 (sem reflexo) a 1 (espelho). Assim, em cada etapa o resultado da recrusão é multiplicado por um coeficiente menor ou igual a um, fazendo com que diminua de intensidade. Ao fazer isto recursivamente este coeficiente é espalhado e tem-se que o valor fica pequeno, podendo num determinado momento chegar a ser menor do que o valor de representatividade do RGB.

13) Como a equação Phong se relaciona com o Ray-tracing? (0.5 ponto).

A equação de iluminação do Ponto P, conforme visto na questão anterior, pode usar o modelo Phong.

14) Como é o processo padrão de criar animações em softwares de computação gráfica? Em que consiste a geração procedural de animação? (0.5 ponto).

O processo Padrão consiste em criar key-frames, que são quadros chaves: alguns quadros gravam dados sobre a informação que se quer animar. Nos quadros intermediários o software irá realizar uma interpolação entre estas chaves. A animação procedural consiste em equações ou formulações matemáticas que irão calcular a alteração de algum parâmetro da cena ou do objeto em função do tempo.

15) O que é quantização e quais são suas aplicações. Descreva de forma sucinta o Algoritmo da Populosidade (0.5 ponto).

Quantização é uma transformação que visa discretizar o gamute de cores de uma imagem, além de reduzir o número de bits necessário para armazenar a informação de cor. As principais aplicações estão na compressão e transmissão de imagens.

Algoritmo da Populosidade:

1. Calcule o histograma da imagem.
2. Escolha os k níveis de quantização como sendo as k cores mais freqüentes (mais populosas).
3. A função de quantização q(c) atribui a cada cor c o nível de quantização mais próximo de c segundo o quadrado da métrica euclidiana.

Obs.: Em caso de empate fazer uma escolha aleatória ou considerar a vizinhança do pixel.