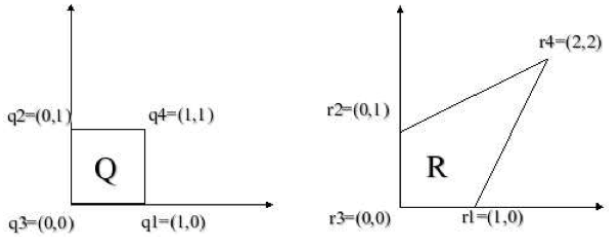
1) Ao programar um jogo, sabe-se que texturas com alta resolução podem trazer problemas quando o objeto em questão está longe da câmera. Qual é este problema? Como corrigir? (1.0 ponto)

São problemas referentes a aliasing: Ocorre a amostragem em um pixel da imagem em uma área grande da textura, correspondente a dois ou mais texels. Podem ocorrer ruídos e pedaços da textura aparecendo e desaparecendo aleatoriamente. Para corrigir é necessário aplicar filtros na textura.

2) Escreva a matriz de projeção perspectiva e demonstre porque ela é assim. (1.0 ponto).

A matriz de projeção perspectiva é uma matriz que representa uma transformação: $T : RP^2 \rightarrow RP^2$ no plano projetivo. Assim, uma transformação projetiva P em RP^2 fica caracterizada quando são conhecidas as imagens por P de 4 pontos em "posição geral" (3 quaisquer não estão em linha reta). Portanto, para transforma o quadrilátero A no quadrilátero B.



Neste caso, o que se deseja é achar a matriz que faz a transformação da Q em R na forma:
$R = QT$

$$T = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ -1 & -1 & 3 \end{bmatrix}$$

A prova pode ser observada por:

$$T(1,0)=T(1,0,1)=\begin{pmatrix}2 & 0 & 0\\ 0 & 2 & 0\\ -1 & -1 & 3\end{pmatrix}\begin{pmatrix}1\\0\\1\end{pmatrix}=\begin{pmatrix}2\\0\\2\end{pmatrix}=\begin{pmatrix}1\\0\\1\end{pmatrix}=\begin{pmatrix}1\\0\end{pmatrix}$$

$$T(0,1)=T(0,1,1)=\begin{pmatrix}2 & 0 & 0\\ 0 & 2 & 0\\ -1 & -1 & 3\end{pmatrix}\begin{pmatrix}0\\1\\1\end{pmatrix}=\begin{pmatrix}0\\2\\2\end{pmatrix}=\begin{pmatrix}0\\1\\1\end{pmatrix}=\begin{pmatrix}0\\1\end{pmatrix}$$

$$T(0,0)=T(0,0,1)=\begin{pmatrix}2 & 0 & 0\\ 0 & 2 & 0\\ -1 & -1 & 3\end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix}=\begin{pmatrix}0\\0\\3\end{pmatrix}=\begin{pmatrix}0\\0\\1\end{pmatrix}=\begin{pmatrix}0\\0\end{pmatrix}$$

$$T(1,1)=T(1,1,1)=\begin{pmatrix}2 & 0 & 0\\ 0 & 2 & 0\\ -1 & -1 & 3\end{pmatrix}\begin{pmatrix}1\\1\\1\end{pmatrix}=\begin{pmatrix}2\\2\\1\end{pmatrix}=\begin{pmatrix}2\\2\end{pmatrix}$$

3) Escreva uma proposta de call-back para desenhar um cubo sem fundo no OpenGL, aplicando uma textura nas faces. (1 ponto)

```
void desenhar(void)
{
//Define os parâmetros da câmera que serão usados no estágio de culling
glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(-300.0f, 420.0f, -80.0f, 15.0f, -1.0f, 2000.0f);
        gluLookAt(100.0f, 100.0f, -100.0f,  0.0f, 0.0f, 0.0f,  0.0f, 1.0f,0.0f);

        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();


glPushMatrix();
glScale(2.0f, 2.0f, 2.0f); //Aplica a escala para dobrar o tamanhos do quads. Este comando é processado pelo estágio de processamento de vértice

glTranslatef(72.0f , 0.0f, 72.0f );  //Desloca os quads para a posição 72, 0, 72. Este comando é processado pelo estágio de processamento de vértice

glColor3f(1.0f, 1.0f, 1.0f);   //Define cor branca para não afetar a cor da textura.

glBindTexture(GL_TEXTURE_2D, 1);  //Efetua o bind da textura – a textura é processada no estágio de processamento de pixel

glBegin(GL_QUADS);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(0.0f, 0.0f, 0.0f);

glTexCoord2f(0.0f, 1.0f);
glVertex3f(0.0f, fHeight, 0.0f);
```

```
glTexCoord2f(1.0f, 1.0f);
glVertex3f(fWidth, fHeight,0.0f);


glTexCoord2f(1.0f, 0.0f);
glVertex3f(fWidth, 0.0f, 0.0f);//Aplica a textura que é processada no estágio de
processamento de pixel


glutSwapBuffers();  //Troca o framebuvver enviando o buffer construído para a
saída de vídeo
}
```

4) Explique a diferença entre ilumiunação por pixel e iluminação por vértice (1.0 ponto)

Na iluminação por pixel, aplica-se a equação de iluminação (por exemplo, Phong) para cada ponto de superficie de um objeto que corresponde sua projeção num dos pixels da tela. Já na iluminação por vértice aplica-se o cálculo de iluminação para cada vértice, independente de sua projeção na tela. No estágio de rasterização será feita uma interpolação das cores dos vértices, preenchendo o interior do polígono.

5) Escreva em alto nível um programa de Ray-tracing, destacando a sua chamada recursiva (1.0 pontos)

```
Ray_Tracing(VETOR)
Para cada Pixel da Imagem faça
    Objeto_mais_próximo = Nenhum
    Distância_mínima = infinito
  Crie um raio do observador ao pixel
Para cada objeto da cena faça
            Se o raio tem interseção com este objeto
                    Se Distancia_mínima < distancia (câmera até este objeto)
                            Objeto_mais_próximo = este objeto
                    Fim se
            Se Objeto_mais_próximo = Nenhum
                    Pixel = Cor_de_fundo
            Senão
Reflexo  = Calcula_reflexo(Objeto_mais_próximo, luz)
Transmissão = Calcula_transmissão (Objeto_mais_próximo, N)

Pixel = Phong(Objeto) + Ray_Tracing(Reflexo) + Ray_Tracing(Transmissão)
            Fim Se
    Fim se
 Fim faça
Fim faça
```

As chamadas recursivas estão em negrito.

6) Como se faz para resolver o problema de profundidade na rasterização de polígonos? (Expliquem detalhadamente) (1.0 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 triangulo 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.

7) O Ray tracing é um ótimo método para renderizar funções implícitas. Explique (1.0 ponto).

O Ray tracing, por realizar um cálculo exaustivo de interseções com a geometria da cena, pode usar neste cálculo equações que não sejam necessariamente de polígonos, mas de outras superfícies. Na rasterização isto é praticamente impossível, devido ao pipeline baseado em vértices e interpolação de polígonos.

8) Como é o mapeamento de texturas esférico? (1.0 ponto).

Toma-se o ponto que se deseja mapear e projeta-se o mesmo sobre uma esfera hipotética, que o envolve. Pega-se então o ponto que intersepta esta esfera e converte-se para coordenadas esféricas, i.e., toma-se a latitude e longitude do ponto, usando-o como coordenada bidimensional da imagem.

9) O que é Environment Mapping? Porque a usamos? (1.0 pontos)

O reflexo real consiste em traçar um raio que simule a reflexão de um raio de luz para um determinado ponto da superfície, sendo necessário calcular a interação deste raio de reflexo com a cena, para ver qual a luz que chega. Para isto, deve-se calcular a iluminação outra vez, apenas para detectar a contribuição de reflexo. O Environment mapping consiste em substituir este cálculo de reflexo por uma textura conhecida: ao traçar o raio de reflexo, ao invés de calcular outra vez a iluminação e a interação deste raio com a cena toda, calcula-se apenas uma projeção de textura para ele.

10) Compare a animação procedural com a animação baseada em keyframes (1.0 ponto)

O processo Padrão consiste em criar key-frames, que são quadros chaves: alguns quadros grava 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.