# Computação Gráfica I - MAB122 (PLE-2020)
# Professor: João Vitor de Oliveira Silva

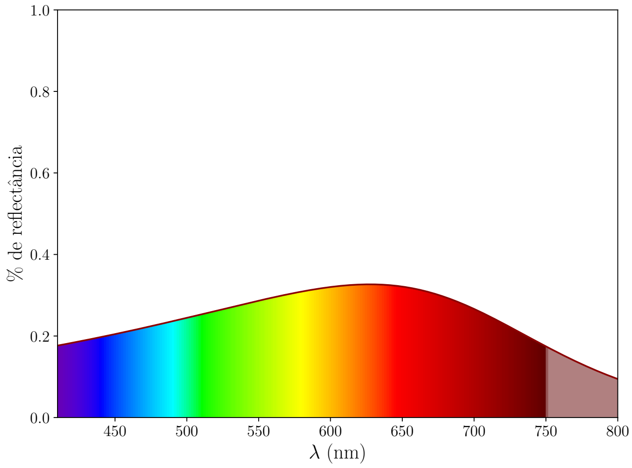
## Lista 1

### Cores

Escolha duas das três questões a seguir para responder:

1. Escreva, com suas palavras, o que é cor. Leve em consideração todos os aspectos envolvidos em sua resposta.

2. Considere a seguinte curva de reflectância:




   Qual das opções a seguir você considera que melhor se associa a curva? **Justifique sua resposta**.

   a) Céu azul
   b) Uma pera amarela
   c) Um papel A4 de cor branca
   d) Luz emitida por lâmpada branca

3. Dadas as seguintes cores no espaço HSL, escreva as representações correspondentes no espaço RGB (normalizado). **Indique os cálculos realizados.**

   a) $[318, 0.8, 0.9]$
   b) $[64, 0.25, 0.31]$
   c) $[159, 0.5, 0]$

Escolha duas das três questões a seguir para responder:

4. Considere os seguintes algoritmos:

**Algoritmo 1:**

```
Entrada: IMAGE imagem[width, height], SCENE cena, CAMERA cam
Variável: RAY raio, OBJECT3D objeto, OBJECT3D obj_closest, FLOAT t_hit,
          FLOAT t_closest, PIXEL pixel
início
    para cada pixel em imagem faça
        raio = cam.GENERATERAYTO(pixel.x, pixel.y)
        t_closest = +∞
        obj_closest = null
        para cada objeto em cena faça
            se raio.INTERSECT(objeto) então
                t_hit = raio.GETINTERSECTIONHITTIME()
                se t_hit < t_closest então
                    t_closest = t_hit
                    obj_closest = objeto
                fim
            fim
        fim
        se obj_closest ≠ null então
            imagem[pixel.x, pixel.y] = COMPUTECOLOR(objeto, t_closest)
        fim
    fim
fim
```

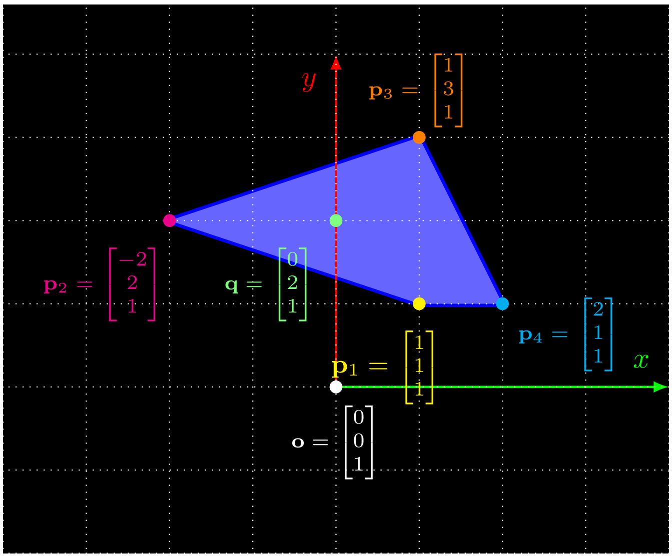
**Algoritmo 2:**

```
Entrada: IMAGE imagem[width, height], SCENE cena, CAMERA cam
Variável: OBJECT3D objeto, PIXEL pixel, OBJECT2D objeto_proj
início
    para cada objeto em cena faça
        objeto_proj = cam.PROJECT(objeto)
        para cada pixel em imagem faça
            se pixel.inside(objeto_proj) então
                imagem[pixel.x, pixel.y] = COMPUTERCOLOR(objeto_proj, pixel)
            fim
        fim
    fim
fim
```

Estes algoritmos são versões bastante simplificadas de algoritmos famosos de renderização. Nomeie cada um deles, explique o funcionamento de ambos e indique suas diferenças.

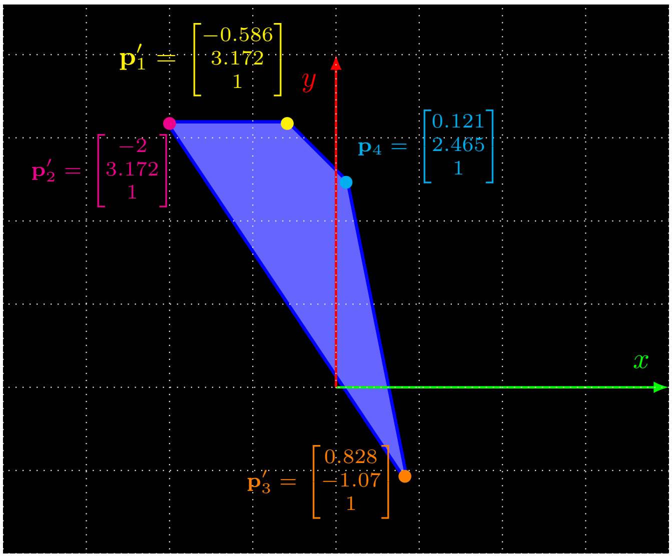
5. As figuras abaixo são exatamente as mesmas, certo? Mas perceba que, ao dar *zoom* neste documento, uma delas parece ficar “pixelada”. Qual a razão disso ocorrer?

6. Indique qual é a orientação do polígono abaixo e mostre que o ponto $\mathbf{q}$ está em seu interior.



Escolha duas das três questões a seguir para responder:

7. Considere que o polígono da questão 6 sofreu uma transformação afim $T$. Os pontos transformados $\mathbf{p}'_1, \mathbf{p}'_2, \mathbf{p}'_3$ e $\mathbf{p}'_4$ podem ser conferidos abaixo:




Informe:

a) A matriz $T$.

b) Uma composição de transformações geométricas (escalonamento, cisalhamento horizontal/vertical, rotação, reflexão e translação) que resulte na matriz $T$.

c) Duas formas de se calcular se um ponto $\mathbf{q}' = T\mathrm{q}$ está no interior do polígono transformado.

8. A curva implícita $5x^2 + 6xy + 5y^2 = 1$ em coordenadas cartesianas representa uma elipse. Encontre uma transformação afim $M$ que mapeie esta elipse em um círculo unitário centrado na origem. *Dica: comece escrevendo a curva implícita de forma vetorial.*

9. Para o problema de animar uma rotação 2D, foram propostas as duas seguintes soluções:

a) Aplicar $R(t) = (1-t)\mathrm{I} + tR_\theta$ sobre o objeto.

b) Aplicar $R(t) = R_{(t\theta)}$ sobre o objeto.

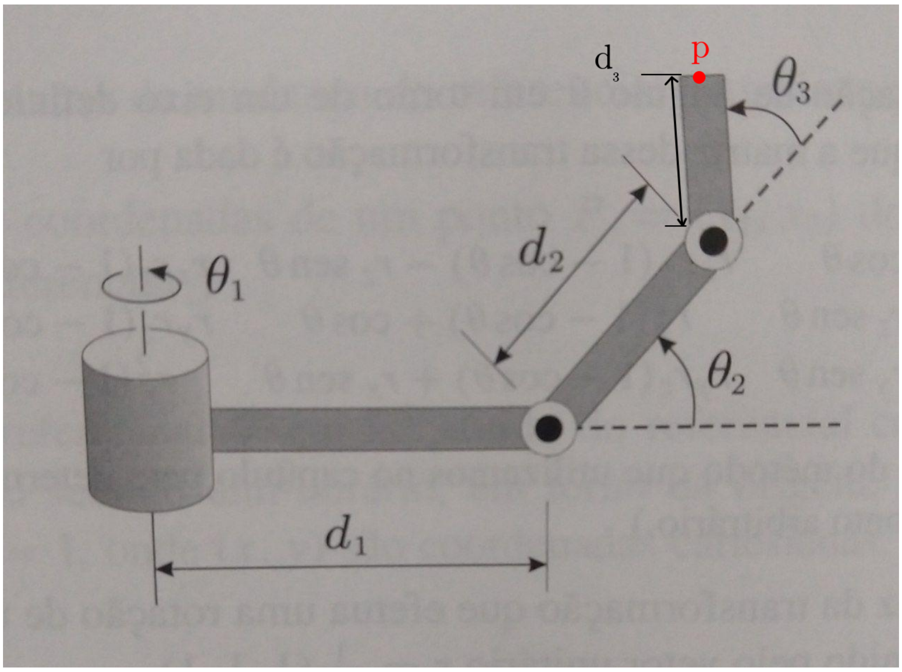
onde $t \in [0, 1]$ é o parâmetro que controla o tempo $t$ da animação.

Qual das duas soluções é a mais apropriada? **Justifique sua resposta**.

Escolha três das quatro questões a seguir para responder:

10. Determine a matriz de rotação de $120^o$ em torno do eixo definido pelo vetor unitário $\mathbf{u}_1 = \frac{1}{\sqrt{3}}[1,1,1]$, e depois uma rotação de ângulo de $-30^o$ em torno do eixo definido pelo vetor $\mathbf{u}_2 = [0,1,0]$. *Dica: construa dois quaternions $q_1$ e $q_2$, calcule o produto dos dois e depois obtenha a matriz de rotação associada.*

11. Considere o braço mecânico abaixo, composto de três partes: antebraço, braço e mão. Usando transformações locais e mudança de referenciais, determine as coordenadas do ponto $\mathbf{p}$ no referencial global.




12. A função *LookAt* serve para posicionar e rotacionar uma câmera virtual, de modo que se capture a região de interesse de uma cena 3D. A matriz que realiza a mudança do referencial global para o da câmera é chamada de *View* ou *World-to-Camera matrix*, a mesma é usada na etapa que antecede o processo de projeção. Mostre que a expressão da *View matrix* é dada por:

$$M_{view} = \begin{bmatrix} \mathbf{v}_{\text{right}} \cdot \mathbf{e}_1 & \mathbf{v}_{\text{right}} \cdot \mathbf{e}_2 & \mathbf{v}_{\text{right}} \cdot \mathbf{e}_3 & -\mathbf{p}_{\text{from}} \cdot \mathbf{v}_{\text{right}} \\ \mathbf{v}_{\text{up}} \cdot \mathbf{e}_1 & \mathbf{v}_{\text{up}} \cdot \mathbf{e}_2 & \mathbf{v}_{\text{up}} \cdot \mathbf{e}_3 & -\mathbf{p}_{\text{from}} \cdot \mathbf{v}_{\text{up}} \\ \mathbf{v}_{\text{forward}} \cdot \mathbf{e}_1 & \mathbf{v}_{\text{forward}} \cdot \mathbf{e}_2 & \mathbf{v}_{\text{forward}} \cdot \mathbf{e}_3 & -\mathbf{p}_{\text{from}} \cdot \mathbf{v}_{\text{forward}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

*Dica: de início, escreva a matriz $M_{view}^{-1} = M_{camera \to World}$, assumindo que* $\mathbf{v}_{\text{right}}, \mathbf{v}_{\text{up}}, \mathbf{v}_{\text{forward}}$ *e* $\mathbf{p}_{\text{from}}$ *são conhecidos. Em seguida, calcule a inversa desta matriz.*

13. No ThreeJS, todos os objetos são descritos por uma `position`, um `quaternion` e uma `scale`. A cada transformação que um objeto sofre, o mesmo executa um método `updateMatrix` que atualiza a matriz de transformação do objeto com um método chamado `compose`. Veja abaixo a implementação em código deste método:

```
compose( position, quaternion, scale ) {
        // te is an array containing the matrix values
        const te = this.elements;
        // getting some values
        const x = quaternion._x, y = quaternion._y, z = quaternion._z;
        const w = quaternion._w;
        const x2 = x + x, y2 = y + y, z2 = z + z;
        const xx = x * x2, xy = x * y2, xz = x * z2;
        const yy = y * y2, yz = y * z2, zz = z * z2;
        const wx = w * x2, wy = w * y2, wz = w * z2;
        const sx = scale.x, sy = scale.y, sz = scale.z;
        // Assigning the first column
        te[ 0 ] = ( 1 - ( yy + zz ) ) * sx;
        te[ 1 ] = ( xy + wz ) * sx;
        te[ 2 ] = ( xz - wy ) * sx;
        te[ 3 ] = 0;
        // Assigning the second column
        te[ 4 ] = ( xy - wz ) * sy;
        te[ 5 ] = ( 1 - ( xx + zz ) ) * sy;
        te[ 6 ] = ( yz + wx ) * sy;
        te[ 7 ] = 0;
        // Assigning the third column
        te[ 8 ] = ( xz + wy ) * sz;
        te[ 9 ] = ( yz - wx ) * sz;
        te[ 10 ] = ( 1 - ( xx + yy ) ) * sz;
        te[ 11 ] = 0;
        // Assigning the fourth column
        te[ 12 ] = position.x;
        te[ 13 ] = position.y;
        te[ 14 ] = position.z;
        te[ 15 ] = 1;
}
```

Explique o funcionamento dessa função, destacando o significado das atribuições realizadas. *Lembre-se que internamente o* ThreeJS *armazena as matrizes de forma column-major.*

## Câmera virtual e transformação projetiva

Resolva obrigatoriamente a questão a seguir:

14. Nessa questão, iremos realizar todas as etapas estudadas do pipeline gráfico. Suponha que temos o ponto $\mathbf{q}_{local} = [2, 1, 3, 1]^T$ em coordenadas locais.

    a) Obtenha as coordenadas globais $\mathbf{q}_{global}$, considerando que a *model matrix* é dada pela sequência das seguintes transformações:

       i. Escalonamento em $y$ de 3 unidades, e escalonamento de 2 unidades em $x$ e $z$.

       ii. Rotação em torno do eixo $x$ de $180^o$.

       iii. Translação de uma unidade em $x$, uma unidade em $y$ e uma unidade em $z$.

    b) Obtenha as coordenadas no referencial da câmera do ponto, ou seja, $\mathbf{q}_{camera}$. A câmera virtual está posicionada em $\mathbf{p}_{from} = [0, 10, 10, 1]^T$, apontada para $\mathbf{p}_{to} = [0, 0, 0, 1]^T$.

    c) Encontre as coordenadas no *clipping space* do ponto $\mathbf{q}_{clip}$, usando que a câmera virtual realiza projeção perspectiva com os seguintes parâmetros:

       - fov: $90^o$
       - aspect: 1
       - near: 2
       - far: 22

       Diga também se o ponto é visível ou não pela câmera.

    d) Realize a etapa de divisão perspectiva, encontrando as coordenadas do ponto no espaço normalizado $\mathbf{q}_{ndc}$.

    e) Usando que a largura $w$ é igual a 800 e que a altura $h$ é também igual a 800, obtenha as coordenas do ponto no espaço de tela (*screen space*). Para realizar este procedimento, faça

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} \frac{w}{2}(1 + x_{ndc}) \\ \frac{h}{2}(1 + y_{ndc}) \end{bmatrix}.$$