QUESTÃO 2 (2 valores)

Dada a matriz de coeficientes geométricos **B** obtenha os vetores coeficientes algébricos $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ *e* $\boldsymbol{d}$ sendo que

$\boldsymbol{a} = [a_x\ a_y\ a_z]$, $\boldsymbol{b} = [b_x\ b_y\ b_z]$, $\boldsymbol{c} = [c_x\ c_y\ c_z]$, $\boldsymbol{d} = [d_x\ d_y\ d_z]$.

Apresente todos os cálculos e matrizes utilizadas.

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & -2 \\ 0 & 1 & 1 \end{bmatrix}$$

R:

2)

Temos que A = MB sendo que M é a Matriz de Transformação Universal.

Tendo ainda que:

$$A = [a\ b\ c\ d]^T = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}$$

Logo basta resolver por cálculo vetorial a expressão (sendo M conhecida):

$$A = MB = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & -2 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & (2+1) & (2-2+1) \\ -2 & (-3-1) & (-3+4-1) \\ 1 & 0 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 3 & 1 \\ -2 & -4 & 0 \\ 1 & 0 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$a = [1 \quad 3 \quad 1];\ b = [-2 \quad -4 \quad 0];\ c = [1 \quad 0 \quad -2];\ d = [0 \quad 0 \quad 1]$

QUESTÃO 3 (2 valores)

Admita que temos as seguintes funções de mistura Bézier.

$B_0 = 4u - 1$

$B_1 = u^2 + u + 1$

$B_2 = -u^2 - 2u + 5$ com $u \in [0,1]$.

Sendo os três pontos de controlo $\boldsymbol{P_0} = (1,0,1)$, $\boldsymbol{P_1} = (-1,0,1)$, $\boldsymbol{P_2} = (-2,\ 1,\ 0)$ calcule o ponto $\boldsymbol{P} = (p_x, p_y, p_z)$ da curva Bézier em $u = \frac{1}{2} = 0.5$

R:

3)

Vamos recorrer ao cálculo matricial referente a Bézier de grau 2 (quadráticas), que são controladas por 3 pontos, tal que:

$p\,(u) = UMP$ em que $U = [u^2 \quad u \quad 1]$; $M = \begin{bmatrix} 0 & 1 & -1 \\ 4 & 1 & -2 \\ -1 & 1 & 5 \end{bmatrix}$ e $P = [P_0\ P_1\ P_2]^T$

Assim vem que:

$$p\,(u) = [u^2 \quad u \quad 1].\begin{bmatrix} 0 & 1 & -1 \\ 4 & 1 & -2 \\ -1 & 1 & 5 \end{bmatrix}.\begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix} = [u^2 \quad u \quad 1].\begin{bmatrix} 0 & 1 & -1 \\ 4 & 1 & -2 \\ -1 & 1 & 5 \end{bmatrix}.\begin{bmatrix} 1 & 0 & 1 \\ -1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix}$$

Se substituirmos por *u* = 0.5 vem:

Veja-se que esta expressão permite obter qualquer ponto da curva com $u \in [0,1]$. Seja u = 0.5 $= \frac{1}{2}$

$$p\left(u = \tfrac{1}{2}\right) = \left[\left(\tfrac{1}{2}\right)^2 \quad \tfrac{1}{2} \quad 1\right].\begin{bmatrix} 0 & 1 & -1 \\ 4 & 1 & -2 \\ -1 & 1 & 5 \end{bmatrix}.\begin{bmatrix} 1 & 0 & 1 \\ -1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix} =$$

$$\left[\tfrac{1}{4} \quad \tfrac{1}{2} \quad 1\right].\begin{bmatrix} 1 & -1 & 1 \\ 7 & -2 & 5 \\ -12 & 5 & 0 \end{bmatrix} =$$

$$\left[\left(\left(\frac{1}{4}\right) + \left(\frac{7}{2}\right) - 12\right), \left(-\left(\frac{1}{4}\right) - \left(\frac{2}{2}\right) + 5\right), \left(\left(\frac{1}{4}\right) + \left(\frac{5}{2}\right)\right)\right]$$

=

$$\left[\left(\left(\frac{15}{4}\right) - 12\right), \left(-\left(\frac{5}{4}\right) + 4\right), \left(\frac{11}{4}\right)\right]$$

QUESTÃO 4 (1 valor)

Apresente e justifique uma técnica de representação de sólidos que recomendaria para modelação de uma família de peças mecânicas de automóvel.

R:

4)

As peças de automóvel assumem usualmente formas volumétricas regulares ainda que com reentrâncias, orifícios e cortes, entre outros. Ou seja, a partir de um volume inicial podem ser operadas funções booleanas, extrusões, cortes, etc.

A técnica que mais se adequa é a instanciação de primitivas que a partir de uma biblioteca inicial de formas volumétricas base permite operar sobre as mesmas um mesmo conjunto de funções booleanas, tais como os recortes regulares de acordo com parâmetros dados de entrada que permitem gerar famílias de rodas dentadas a partir de um disco inicial, por exemplo.

QUESTÃO 5 (4 valores)

Considere o código abaixo. Explique sucintamente o que ele executa e descreva/comente o que cada linha de código faz:

```
public void init(GLAutoDrawable drawable) {
    GL2 gl = drawable.getGL().getGL2();
    glu = new GLU();
    gl.glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    gl.glClearDepth(1.0f);
    gl.glEnable(GL_DEPTH_TEST);
  }
   @Override
  public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) {
    GL2 gl = drawable.getGL().getGL2();

    if (height == 0) height = 1;
    float aspect = (float)width / height;
    gl.glViewport(0, 0, width, height);

    gl.glMatrixMode(GL_PROJECTION);
    gl.glLoadIdentity;
    glu.gluPerspective(45.0, aspect, 0.1, 100.0);
     gl.glMatrixMode(GL_MODELVIEW);
     gl.glLoadIdentity();
  }
   @Override
  public void display(GLAutoDrawable drawable) {
    GL2 gl = drawable.getGL().getGL2();
    gl.glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    gl.glLoadIdentity();
```

```
     gl.glTranslatef(0.0f, 0.0f, -6.0f);
    gl.glBegin(GL_TRIANGLES);
      gl.glVertex3f(0.0f, 1.0f, 0.0f);
      gl.glVertex3f(-1.0f, -1.0f, 0.0f);
      gl.glVertex3f(1.0f, -1.0f, 0.0f);
    gl.glEnd();
  }
```

R:

```
public void init(GLAutoDrawable drawable) { // método para inicializar
      GL2 gl = drawable.getGL().getGL2(); // inicializa o contexto no Opengl
      glu = new GLU();                    // acede as funcionalidades da GLU
      gl.glClearColor(0.0f, 0.0f, 0.0f, 0.0f); // fundo da janela a negro
      gl.glClearDepth(1.0f); // define valor para limpeza com z=1.0 (o mais atrás
possível)
      gl.glEnable(GL_DEPTH_TEST); // ativa a utilização do depth buffer para
eliminação de superfícies escondidas

   }

   @Override
   public void reshape(GLAutoDrawable drawable, int x, int y, int width, int
height) { //invocada sempre que ocorre um redimensionar da janela
      GL2 gl = drawable.getGL().getGL2();  // referência para o contexto gráfico
do Opengl

      if (height == 0) height = 1;   // evita erros de divisão por zero
      float aspect = (float)width / height;

      // Define o visor com a dimensão total da janela de visualização
      gl.glViewport(0, 0, width, height);

      // Define projeção em perspectiva
      gl.glMatrixMode(GL_PROJECTION);  // escolhe a matriz de projeção
      gl.glLoadIdentity();             // reset da matriz de projeção
      glu.gluPerspective(45.0, aspect, 0.1, 100.0); // define os parâmetros fovy,
aspect, zNear, zFar

      // Ativa a transformação MODELVIEW
      gl.glMatrixMode(GL_MODELVIEW);
      gl.glLoadIdentity(); // reset da matriz model-view
   }

   @Override
   public void display(GLAutoDrawable drawable) { //invocada constantemente para o
render da janela de visualização
      GL2 gl = drawable.getGL().getGL2();  // get the OpenGL 2 graphics context
      gl.glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // limpa os buffers
para cor e profundidade
      gl.glLoadIdentity();  // reset da matriz model-view
```

```
      // ----- Troço de código onde é inserido o que se quer desenhar
efetivamente, neste caso, um simples triângulo

      gl.glTranslatef(0.0f, 0.0f, -6.0f); // move-o para uma posição na janela
      gl.glBegin(GL_TRIANGLES); // desenhar utilizando triângulos
         gl.glVertex3f(0.0f, 1.0f, 0.0f); // definição de vértices
         gl.glVertex3f(-1.0f, -1.0f, 0.0f);
         gl.glVertex3f(1.0f, -1.0f, 0.0f);
      gl.glEnd();
   }
```