

# Fundamentos Matemáticos para CG

## 1 Matrizes

Uma matriz  $A$   $m \times n$  é uma tabela retangular de elementos com  $m$  linhas e  $n$  colunas e que possui a seguinte notação:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

onde  $a_{jk}$  representa o elemento na  $j$ -ésima linha e  $k$ -ésima coluna.

- Cada elemento da matriz pode armazenar números, funções ou expressões numéricas.
- Em programação, os índices da matriz iniciam em 0.

Uma matriz com uma única linha ou coluna representa um vetor (vetor linha ou vetor coluna).

→ Por convenção, quando várias operações são expressas em formato matricial, a convenção matemática padrão é representar um vetor em formato de coluna.

$$V = \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix}$$

A transposta de uma matriz  $V$ , denominada  $V^T$ , é dada por

$$V = \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} \quad V^T = \begin{bmatrix} V_x & V_y & V_z \end{bmatrix}$$

- Uma matriz é dita **quadrada** quando  $m = n$  (dimensões a matriz).
- Uma matriz é diagonal quando é quadra e quando todos os elementos fora da diagonal principal são zeros.

$$A = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

- A matriz diagonal na qual todos os elementos da diagonal principal são uns é dita matriz **Identidade** **I**.
- A matriz na qual todos os elementos são zeros é denominada matriz nula.
- Duas matrizes A e B são iguais se possuem a mesma dimensão e  $a_{ij} = b_{ij}$  para todo i e j.

## Operações com matrizes.

Sejam **A** uma matriz e  $\alpha$  um número real, normalmente chamado de um **escalar**.

- O **múltiplo escalar** de  $\alpha$  e **A**, denotado por  $\alpha A$ , é obtido pela multiplicação de cada elemento de **A** por  $\alpha$ .
- Se  $\alpha = -1$ , o múltiplo escalar é chamado de negativo de **A** e denotado por **-A**.

$$A = \begin{bmatrix} 4 & 1 & 0 \\ 0 & 2 & 0 \\ -4 & 0 & 5 \end{bmatrix} \text{ e } \alpha = 2 \quad \alpha A = \begin{bmatrix} 8 & 2 & 0 \\ 0 & 4 & 0 \\ -8 & 0 & 10 \end{bmatrix}$$

A **soma** de duas matrizes **A** e **B** que têm as mesmas dimensões é a matriz **C** = **A** + **B** obtida pela soma dos seus respectivos elementos (isto é,  $c_{ij} = a_{ij} + b_{ij}$ ).

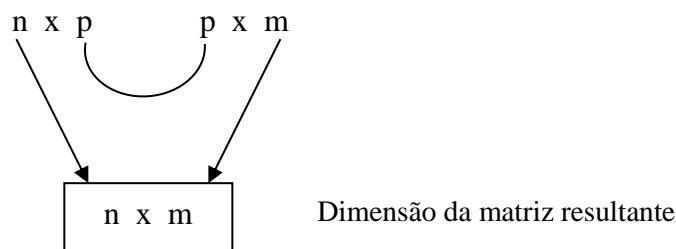
Similarmente, a **diferença** **C** = **A** - **B** é obtida por  $c_{ij} = a_{ij} - b_{ij}$ .

$$A = \begin{bmatrix} 1 & 0 \\ 5 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 4 & 6 \\ 1 & 1 \end{bmatrix}, \quad A + B = \begin{bmatrix} 5 & 6 \\ 6 & 2 \end{bmatrix}$$

O **produto** entre duas matrizes, denominado por **AB**, de uma matriz A (n x p) por uma matriz B (p x m) é uma matriz C (n x m) onde

$$c_{ij} = \sum_{s=1}^p a_{is} b_{sj} \text{ para } i = 1, 2, \dots, n \text{ e } j = 1, 2, \dots, m$$

Pela definição, somente pode-se multiplicar duas matrizes caso o número de colunas da primeira seja igual ao número de linhas da segunda.



Deve-se observar que  $AB \neq BA$  (**não comutativa**). Porém as seguintes propriedades são verdadeiras:

$$(aF)G = a(FG) \quad (\text{associativa})$$

$$(FG)H = F(GH) \quad (\text{associativa})$$

$$(FG)^T = G^T F^T \quad (\text{Propriedade usada na concatenação de matrizes de transformação})$$

## Propriedades

Dados dois escalares a e b quaisquer e quaisquer matrizes n x m F, G e H, as seguintes propriedades são verdadeiras:

$F + G = G + F$  (comutativa)  
 $(F + G) + H = F + (G + H)$  (associativa)  
 $a(bF) = (ab)F$  (associativa)  
 $a(F + G) = aF + aG$  (distributiva)  
 $(a+b)F = aF + bF$  (distributiva)

## Matriz inversa

Uma matriz  $M$   $n \times n$  é inversível se existir uma matriz, denotada por  $M^{-1}$ , de forma que

$$MM^{-1} = M^{-1}M = I.$$

Ou seja,

$$A = \begin{bmatrix} 1 & 0 \\ 5 & 1 \end{bmatrix}$$

A matriz  $M^{-1}$  é chamada inversa de  $M$ . Uma matriz é chamada **singular** quando não for inversível. Um exemplo de matriz singular é uma matriz que possui uma linha ou coluna com todos valores iguais a zero.

**Teorema:** Uma matriz  $M$  é inversível se e somente se  $M^T$  é inversível.

Um algoritmo para calcular a matriz inversa pode ser encontrado em [1].

## Matriz Ortogonal

Uma matriz quadrada  $A$  é **ortogonal** se  $MM^T = I$ . Portanto, se  $M$  é ortogonal então

$$M^{-1} = M^T$$

Uma outra consequência é que se  $M$  é ortogonal então  $\det M$  é igual a 1 ou -1.

Propriedades:

- Se  $A$  e  $B$  são ortogonais então  $C = AB$  também é ortogonal.
- Se  $A$  é ortogonal então  $A^{-1}$  também é ortogonal.

Uma matriz é dita **ortonormal** se o comprimento de cada linha ou coluna, tomados como vetores, tem comprimento 1 (normalizado). Esses vetores também são mutuamente **ortogonais** (formam um ângulo de  $90^\circ$  cada (perpendiculares) – ver definição de produto interno/escalar).

## Determinante

O determinante de uma **matriz quadrada** é um simples valor numérico que diz muito a respeito da matriz e tem várias utilidades.

O determinante de uma matriz  $A$   $m \times n$ , denotado por  $\det A$ , é dado por:

$$\det A = \sum a_{ij} (-1)^{i+j} M_{ij}$$

onde  $M_{ij}$ , chamado de **menor** da matriz  $A$ , é o determinante da matriz formada pela eliminação da linha  $i$  e da coluna  $j$  de  $A$ . O exemplo a seguir ilustra o processo:

$$\begin{vmatrix} 1 & 2 & 4 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix} = 1(-1)^{1+1} \begin{vmatrix} 5 & 6 \\ 8 & 9 \end{vmatrix} + 2(-1)^{1+2} \begin{vmatrix} 4 & 6 \\ 7 & 9 \end{vmatrix} + 4(-1)^{1+3} \begin{vmatrix} 4 & 5 \\ 7 & 8 \end{vmatrix}$$

O cálculo do determinante é, portanto, um processo recursivo, pois requer o valor do determinante de uma matriz de uma dimensão menor. O cálculo do determinante de uma matriz 2x2 é dado por:

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

Desta forma,

$$\begin{vmatrix} 1 & 2 & 4 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix} = 1(45 - 48) - 2(36 - 42) + 4(32 - 35) = -3$$

Para matrizes 3x3, pode-se também utilizar o seguinte algoritmo.

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = aei + bfg + cdh - gec - dbi - ahf$$

Caso tiver uma linha com valores = 0,  $\det = 0$ ;

O cálculo de determinante pode ser usado, por exemplo, para indicar:

- Determinar se um conjunto de vetores  $\{v_1, v_2, \dots, v_n\}$  forma uma base do  $\mathbb{R}^n$ , ou seja, se o sistema é LI ou LD (ver capítulo de Vetores). **Se  $\det \mathbf{0}$ , é LD;**
- O mesmo cálculo pode ser usado para determinar se 3 pontos em 2D são colineares

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = x_1 y_2 + y_1 x_3 + x_2 y_3 - x_3 y_2 - x_2 y_1 - y_3 x_1 = 0$$

Outra forma de determinar se 3 pontos são colineares consiste em primeiro criar 2 vetores a partir dos 3 pontos (Ver próxima seção). A partir dos vetores, determina-se o coeficiente angular dos dois vetores,  $y = mx + b$ . Pelo fato das retas estarem na origem, o termo  $b$  pode ser desconsiderado. Assim, temos  $m=y/x$ . As retas são colineares se ambas tiverem o mesmo coeficiente angular. Para evitar a divisão, passam-se os divisores para o outro lado da equação multiplicando.

```
Vetor v1 = i-j;
Vetor v2 = j-k;
if( v1.x * v2.y == v1.y * v2.x )
    true;
```

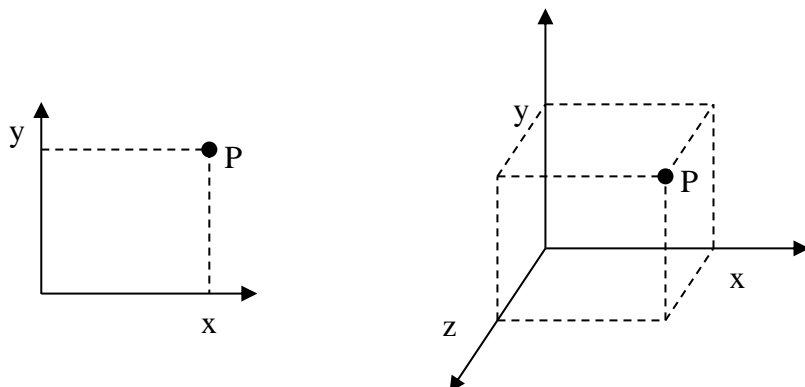
- Determinar se uma matriz é ortogonal.  $\det = 1$  ou  $-1$ .

## 2 Vetores – Conteúdo mais importante da disciplina

Esta seção introduz os principais conceitos matemáticos de vetores utilizados em aplicações gráficas e na implementação de personagens autônomos (NPCs) em jogos de computador. Este material pode ser utilizado tanto para aplicações 2D como para 3D.

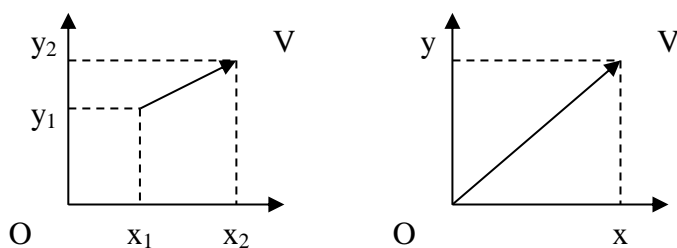
### Ponto

Todo personagem, imerso em um ambiente, possui uma localização. Essa localização pode ser dada por um ponto no espaço. As coordenadas deste espaço são  $(x,y)$  para o caso em duas dimensões e  $(x,y,z)$  para 3 dimensões (sistema cartesiano). Assim, um ponto  $P$  é representado por  $P = (x,y)$  ou  $P = (x,y,z)$ .



### Vetor

Para aplicação em CG (e IA), um vetor possui a mesma representação de um ponto  $V = (x,y)$ . Porém, seu significado pode ser diferente. Um vetor, como na matemática, possui um módulo (comprimento ou magnitude) e uma direção. Para isso, possui um ponto de origem e um ponto destino. Para que um vetor, representado por um único ponto, tenha toda essa informação, considera-se que a origem é sempre a origem do sistema de coordenadas. Ou seja, um vetor é definido pela diferença entre dois pontos:  $V = P_2 - P_1$ , ou seja, o ponto final menos o ponto inicial.



A seta do vetor indica o sentido e o comprimento o módulo, também chamado de magnitude. Vetores podem ser usados para representar qualquer quantidade que possui as propriedades de magnitude e direção. Dois exemplos comuns são força e velocidade. A força pode ser vista como puxar ou empurrar em certa intensidade em uma direção particular. Um vetor velocidade especifica quanto rápido um objeto está movendo em uma certa direção.

Neste texto, define-se vetor como uma n-upla de valores reais, onde  $n$  é tipicamente 2, 3 ou 4. Um vetor  $n$ -dimensional pode ser escrito como:

$$V = \langle V_1 \quad V_2 \quad \dots \quad V_n \rangle$$

Onde os números  $V_i$  são chamados componentes do vetor  $V$ . Ao invés de usar as componentes numeradas, geralmente usa-se o nome do eixo que cada componente corresponde. Como exemplo, as componentes de um vetor tridimensional podem ser  $P_x$ ,  $P_y$  e  $P_z$ . Os vetores também podem ser representados em formato matricial de uma coluna e  $n$  linhas:

$$V = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix}$$

Esta representação também pode ser escrita na forma de uma matriz com uma linha e várias colunas, ou seja, a matriz transposta de  $V$ :

$$V^T = [V_1 \quad V_2 \quad \dots \quad V_n]$$

Geralmente, em jogos utilizam-se somente vetores ao invés de pontos. Eles são usados para representar a posição de um personagem, sua direção, em cálculos de caminhos, determinação de linha de tiro, visibilidade, ângulos, forças, dentre outros. No caso de personagens, deve-se utilizar pelo menos dois vetores para sua representação: um para a posição e outro para a direção.

A magnitude de um vetor  $V$ , módulo, norma ou comprimento, definido por  $|V|$  ou  $\|V\|$ , é dada pela seguinte equação

$$|V| = \sqrt{\sum_{i=1}^n V_i^2}$$

Este cálculo representa a distância da origem até a extremidade do vetor. Matematicamente, utiliza-se a notação  $|V|$  para indicar o módulo do vetor  $V$ . Para calcular a distância entre 2 vetores quaisquer, utiliza-se a mesma fórmula, incluindo-se a subtração de cada componente. Observe que esta equação representa o teorema de Pitágoras.

Quando  $V$  representa um ponto ou direção tridimensional, esta equação pode ser reescrita da seguinte forma:

$$|V| = \sqrt{V_x^2 + V_y^2 + V_z^2}$$

Um vetor cuja magnitude seja exatamente igual tem um comprimento unitário ou pode ser simplesmente chamado de vetor unitário. Para muitas transformações de vetores, é essencial que este possua módulo (magnitude) unitário. Para transformar um vetor em unitário, basta dividir cada componente pela sua norma. A **normalização** mantém a direção e sentido do vetor.

$$V = \frac{V}{|V|} \quad V = \left\langle \frac{V_x}{|V|}, \frac{V_y}{|V|}, \frac{V_z}{|V|} \right\rangle$$

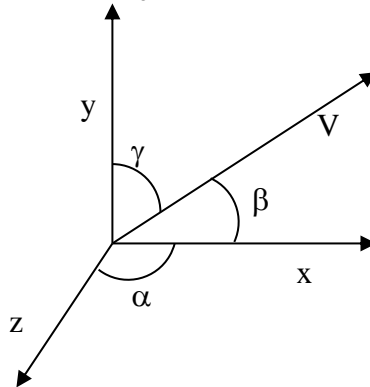
O seguinte código é uma versão otimizada do cálculo do inverso da raiz quadrada de um número real, que é uma operação típica na normalização de vetores, muito comum em jogos. Para mais detalhes do seu funcionamento, bem como investigação na descoberta de seu criador, consulte <http://www.beyond3d.com/content/articles/8/>. A análise do funcionamento deste algoritmo foi desenvolvida por Chris Lomont em <http://www.lomont.org/Math/Papers/2003/InvSqrt.pdf>. Outra dedução da formulação utilizada por ser vista em “*The Mathematics Behind the Fast Inverse Square Root Function Code*”, por Charles McEniry (2007). Este código foi divulgado globalmente com a abertura do código fonte do jogo Quake3.

```
float InvSqrt (float x)
{
    float xhalf = 0.5f*x;
    int i = *(int*)&x;
    i = 0x5f3759df - (i>>1);
    x = *(float*)&i;
    x = x*(1.5f - xhalf*x*x);
    return x;
}
```

A direção de um vetor é dada por 3 ângulos  $\alpha$ ,  $\beta$  e  $\gamma$ , que o vetor forma com cada eixo de coordenada (Figura 3). Estes ângulos são calculados da seguinte forma:

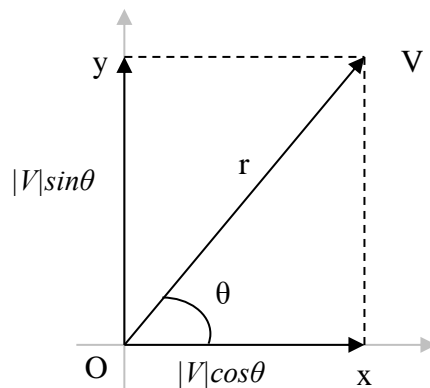
$$\cos \alpha = \frac{\text{adjacente}}{\text{hipotenusa}} = \frac{V_x}{|V|}, \quad \cos \beta = \frac{V_y}{|V|}, \quad \cos \gamma = \frac{V_z}{|V|}$$

Os valores  $\cos \alpha$ ,  $\cos \beta$  e  $\cos \gamma$  são os cossenos diretores do vetor. Uma vez que  $\cos^2 \alpha + \cos^2 \beta + \cos^2 \gamma = 1$ , basta especificar somente dois cossenos para se ter a direção do vetor  $V$ .



### Coordenadas Polares – Um dos conteúdos mais usados na cadeira

Um sistema de coordenadas não cartesianas de uso freqüente é o sistema de coordenadas Polares. Esse sistema usa uma distância  $r$  e um ângulo  $\theta$  para representar um **ponto** no espaço, sendo  $r$  a distância do ponto a origem e  $\theta$  o ângulo formado entre o vetor definido pelo ponto e origem  $O$  com o eixo  $x$ , como mostrado na seguinte figura. Neste sistema, um vetor é representado da mesma maneira. Aumentos positivos no ângulo fazem o vetor girar em sentido anti-horário.



Considerando-se que se tenha somente o módulo e o ângulo de um vetor, pode-se obter as componentes  $x$  e  $y$  deste vetor pelas seguintes equações:

$$x = r \cos \theta$$

$$y = r \sin \theta$$

Estas duas equações podem ser utilizadas na geração de círculos de raio  $r$ . A transformação inversa do sistema Cartesiano para o sistema Polar é dada por

$$r = \sqrt{x^2 + y^2} \quad \theta = \tan^{-1}\left(\frac{y}{x}\right)$$

Usos:

- Criação de entidades móveis – vetor direção.
- Criação de relógio analógico, engrenagem, espiral, cilindro, esfera, sweep rotacional, etc.
- Matriz de rotação
- Etc.

## Operações sobre vetores

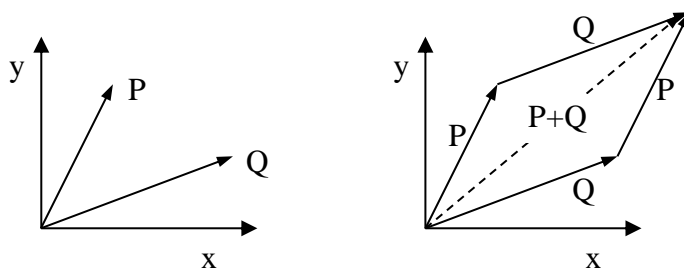
### Soma de Vetores

A soma de vetores  $V_1 + V_2 + \dots + V_n$  é feita somando-se as componentes de cada vetor, ou seja,

$$V_r = P + Q = (P_x + Q_x, P_y + Q_y, P_z + Q_z).$$

Sendo  $P = (2, 3, -4)$  e  $Q = (1, -5, 3)$ ,  $V_r = Q + P = (2+1, 3-5, -4+3) = (3, -2, -1)$ .

Geometricamente, a soma de dois vetores é ilustrada na seguinte figura. Eles podem ser adicionados geometricamente posicionando-se um vetor ao final do outro e desenhando o vetor resultante a partir do início do primeiro vetor até o final do segundo vetor.



### Multiplicação por Escalar

Para multiplicar um vetor por um escalar  $a$ , ou seja,  $V_r = aV$ , deve-se multiplicar cada componente do vetor pelo escalar, ou seja,

$$V_r = aV = (aV_x, aV_y, aV_z).$$

Sendo  $V_l = (2, 4, 0)$ ,  $V_r = 2V_l = (2*2, 2*4, 2*0) = (4, 8, 0)$ . Multiplicações por escalar somente podem mudar a magnitude e o sentido de um vetor. Nunca muda a direção. Para que ocorra mudança de sentido, basta multiplicar por um escalar negativo. Neste tipo de operação, as componentes mantêm as mesmas proporções relativas.

### Propriedades de Vetores

Dados quaisquer dois escalares  $a$  e  $b$  e quaisquer três vetores  $P$ ,  $Q$  e  $R$ , verifica-se as seguintes propriedades:

- $P + Q = Q + P$
- $(P + Q) + R = P + (Q + R)$
- $(ab)P = a(bP)$
- $a(P + Q) = aP + aQ$
- $(a + b)P = aP + bP$
- $|P| \geq 0$
- $|P| = 0$  se e somente se  $P = \langle 0 \ 0 \ \dots \ 0 \rangle$



- h)  $|aP| = |a| |P|$   
 i)  $|P + Q| \leq |P| + |Q|$  (Demonstrável pela Figura 5)

## Produto Escalar

O produto Escalar, também chamado produto interno ou *Dot Product*, é usado para medir ângulo entre vetores (ou a diferença entre as direções dos dois vetores). O produto escalar de dois vetores  $P$  e  $Q$ , é um valor escalar dado pela fórmula

$$P \cdot Q = \sum_{i=1}^n P_i Q_i \quad \text{ou} \quad \langle P, Q \rangle = \sum_{i=1}^n P_i Q_i$$

Ou seja, o produto escalar de dois vetores é dado pela soma do produto de cada componente. Assim,  $|V|^2 = V \cdot V$ . Em três dimensões temos

$$P \cdot Q = P_x Q_x + P_y Q_y + P_z Q_z$$

O produto escalar também pode ser expresso como o produto de duas matrizes

$$P^T Q = \begin{bmatrix} P_1 & P_2 & \dots & P_n \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_n \end{bmatrix}$$

Um propriedade importante do produto escalar de dois vetores  $P$  e  $Q$   $n$ -dimensionais é

$$P \cdot Q = |P||Q| \cos \alpha \quad (0 \leq \alpha \leq \pi) \quad \text{Eq. 1}$$

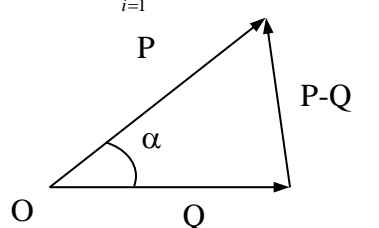
onde  $\alpha$  é o ângulo planar entre as linhas conectando a origem dos pontos representados por  $P$  e  $Q$ . Para demonstrar esta equação, usa-se as leis dos cossenos sobre a Figura 6. Sendo  $\alpha$  o ângulo entre os segmentos de linha  $\overline{OP}$  e  $\overline{OQ}$ ,

$$\begin{aligned} |P - Q|^2 &= |P|^2 + |Q|^2 - 2|P||Q|\cos \alpha \\ \sum_{i=1}^n (P_i - Q_i)^2 &= \sum_{i=1}^n P_i^2 + \sum_{i=1}^n Q_i^2 - 2|P||Q|\cos \alpha \\ \sum_{i=1}^n P_i^2 - \sum_{i=1}^n 2P_i Q_i + \sum_{i=1}^n Q_i^2 &= \sum_{i=1}^n P_i^2 + \sum_{i=1}^n Q_i^2 - 2|P||Q|\cos \alpha \end{aligned}$$

Todos os termos  $P_i^2$  e  $Q_i^2$  cancelam e temos

$$\sum_{i=1}^n -2P_i Q_i = -2|P||Q|\cos \alpha$$

Dividindo por -2 temos o resultado desejado

$$\sum_{i=1}^n P_i Q_i = |P||Q|\cos \alpha$$


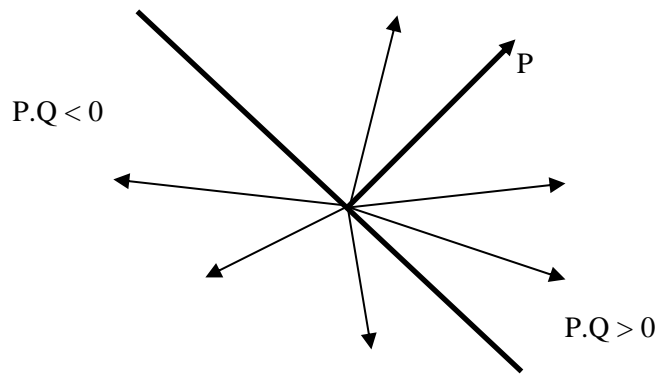
Logo, o ângulo entre dois vetores é dado por

$$\cos \alpha = \frac{P \cdot Q}{|P||Q|}$$

Caso os vetores  $P$  e  $Q$  forem unitários, temos

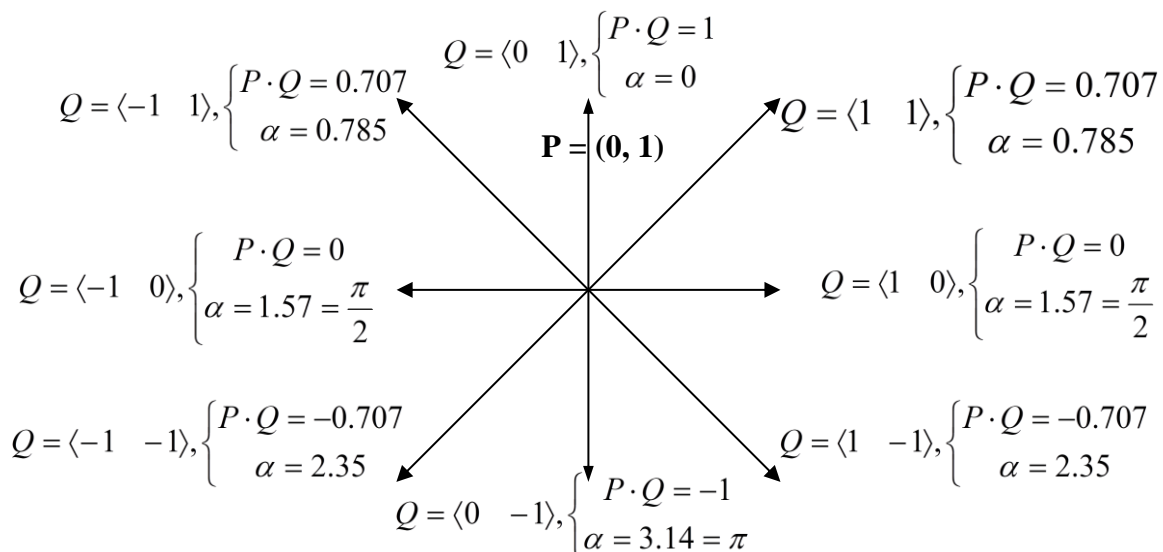
$$\begin{aligned}\cos \alpha &= \frac{P \cdot Q}{1 \cdot 1} = P \cdot Q \\ \alpha &= \cos^{-1}(P \cdot Q)\end{aligned}$$

Uma série de fatos importantes seguem imediatamente a partir da Equação 1. A primeira é que dois vetores  $P$  e  $Q$  são perpendiculares se e somente se  $P \cdot Q = 0$ . Isso é resultado do fato que a função cosseno é zero em um ângulo de 90 graus. Vetores cujo produto interno seja zero são ditos vetores ortogonais. Por definição, o vetor zero  $0 = \langle 0 \ 0 \ \dots \ 0 \rangle$  é ortogonal a qualquer vetor, visto que  $0 \cdot P$  é sempre zero. O segundo fato é que o sinal do produto interno nos diz quanto próximos dois vetores estão apontando na mesma direção. Referente a próxima figura, toma-se o plano passando pela origem e perpendicular ao vetor  $P$ . Qualquer vetor que estiver no mesmo lado do plano como  $P$  produz um produto escalar positivo com  $P$ , e qualquer vetor apontando para o lado oposto do lado de  $P$  produz um valor negativo do produto escalar com  $P$ .



O sinal do produto escalar diz se dois vetores estão em um mesmo lado de um plano

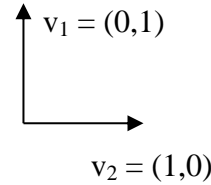
A seguinte figura ilustra alguns valores de  $P \cdot Q$  e  $\alpha = \cos^{-1}(P \cdot Q)$  para diversas situações de  $P$  e  $Q$  **unitários** e bidimensionais (Observe que  $Q$  nem sempre é unitário):



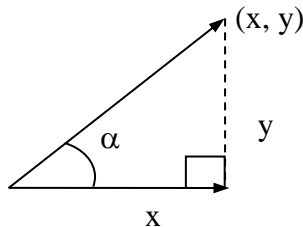
Deve-se observar que o ângulo  $\alpha$  encontrado varia de 0 a  $\pi$  (ou seja, é sempre positivo) e que o produto escalar varia entre 1 e -1. Além disso, não se pode afirmar se um vetor  $Q$  está a esquerda ou direita de  $P$ , como ocorre por exemplo quando  $Q = \langle -1 \ 1 \rangle$  e  $Q = \langle 1 \ 1 \rangle$ .

Uma forma de determinar a posição relativa de dois vetores é realizando a seguinte comparação entre dois vetores  $v_1$  e  $v_2$  unitários:

Se  $(v_1.y * v_2.x > v_1.x * v_2.y)$   
 Ângulo anti-horário  $\rightarrow$  *counterclockwise*  
 Senão  
 Ângulo horário  $\rightarrow$  *clockwise*



Pode-se também usar o arco tangente para calcular o ângulo entre dois vetores, porém em situações bem específicas (ângulo reto). A sintaxe da função é:  $\alpha = \text{atan2}(y, x)$ , ou  $\alpha = \text{atan}(y/x)$ . Essa função calcula o ângulo entre o vetor  $v_2 = (1, 0)$ , definido implicitamente ao longo do eixo  $x$ , e o vetor  $v_1 = (x, y)$  informado pelo usuário. O valor retornado é igual ao do algoritmo acima apresentado.



Existem várias outras propriedades do produto interno:

- a)  $P \cdot Q = Q \cdot P$  (simetria)
- b)  $(aP) \cdot Q = a(P \cdot Q)$
- c)  $P \cdot (Q + R) = P \cdot Q + P \cdot R$
- d)  $|P \cdot Q| \leq |P||Q|$

## Produto Vetorial

O produto vetorial (*Cross Product*) entre dois vetores  $P$  e  $Q$ , denotado por  $P \times Q$ , produz um novo vetor que é perpendicular aos dois vetores  $P$  e  $Q$ . Este recurso é muito útil no desenvolvimento de jogos e pode ser usado para encontrar vetores normais a superfícies bem como para definir trajetória de personagens.

O produto vetorial de dois vetores  $P$  e  $Q$  é dado por

$$P \times Q = \langle P_y Q_z - P_z Q_y, P_z Q_x - P_x Q_z, P_x Q_y - P_y Q_x \rangle$$

Em formato matricial, o produto vetorial pode ser dado pelo cálculo do pseudo-determinante da matriz (tem esse nome porque a linha superior da matriz é composta por vetores, e o restante por escalares):

$$P \times Q = \begin{vmatrix} i & j & k \\ P_x & P_y & P_z \\ Q_x & Q_y & Q_z \end{vmatrix}$$

onde  $i, j$  e  $k$  são vetores unitários paralelos aos eixos  $x, y$  e  $z$ .

$$i = \langle 1 \ 0 \ 0 \rangle$$

$$j = \langle 0 \ 1 \ 0 \rangle$$

$$k = \langle 0 \ 0 \ 1 \rangle$$

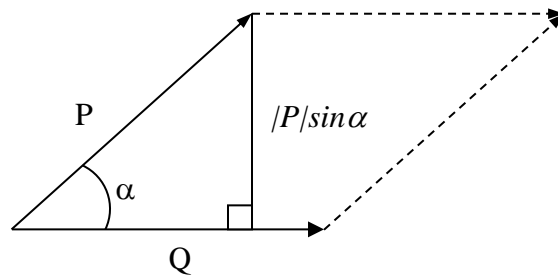
$$\begin{vmatrix} i & j & k \\ P_x & P_y & P_z \\ Q_x & Q_y & Q_z \end{vmatrix} = i(P_y Q_z - P_z Q_y) + j(P_z Q_x - P_x Q_z) + k(P_x Q_y - P_y Q_x)$$

Pela definição de produto escalar e vetorial,  $(P \times Q) \cdot P = 0$  e  $(P \times Q) \cdot Q = 0$

O produto vetorial  $P \times Q$  tem a propriedade de ser perpendicular ao plano definido por  $P$  e  $Q$  e o comprimento é dado por

$$|P \times Q| = |P||Q|\sin \alpha$$

Em termos geométricos, o comprimento (magnitude) de  $P \times Q$  é dado pela área do paralelogramo cujos lados são formados pelos vetores  $P$  e  $Q$  (Figura 9).

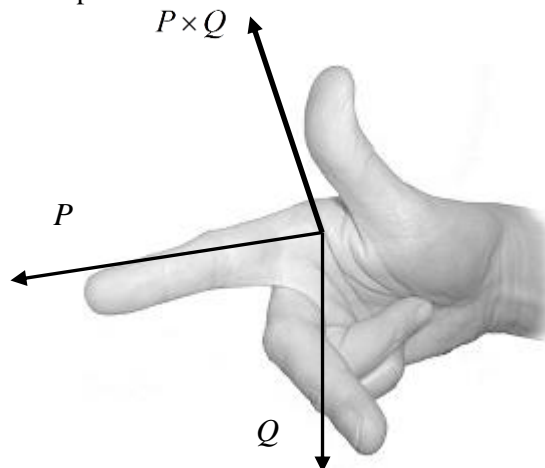


**Figura 9:** A magnitude do produto vetorial de dois vetores é igual a área do paralelogramo cujos lados são formados pelos vetores

O vetor gerado pelo produto vetorial é sempre perpendicular ao plano formado pelos vetores. Mas existem duas possíveis direções para este novo vetor. O produto vetorial segue a regra da mão direita, que estabelece que se os dedos apontam na direção do vetor  $P$ , a palma da mão na direção do vetor  $Q$ , então o vetor perpendicular aponta na direção do polegar.

Existem várias propriedades que devem ser observadas no produto vetorial:

- a)  $P \times Q = -Q \times P$
- b)  $(aP) \times Q = a(P \times Q)$
- c)  $P \times (Q + R) = P \times Q + P \times R$
- d)  $P \times P = 0 = \langle 0 \ 0 \ 0 \rangle$
- e)  $(P \times Q) \cdot R = (R \times P) \cdot Q = (Q \times R) \cdot P$



## Aplicações de Vetores em CG e Jogos

### Deslocamento sob o vetor direção.

Uma forma de fazer o deslocamento de uma unidade do jogo é incrementar ao seu vetor posição um valor proporcional à velocidade de deslocamento na direção de seu deslocamento.

```
float speed = 10;
actorPos.x += actorDir.x * speed;
actorPos.z += actorDir.z * speed;
```

O valor de *speed* vai depender de vários fatores. Um é a velocidade real do personagem no jogo e outro é o frame rate de renderização das cenas. Se a taxa de atualização é elevada, o valor de *speed* deve ser baixo. Isso é usado para garantir que a velocidade do personagem (ou veículo) independa da velocidade do computador. Deve-se observar que o vetor *actorDir* deve ser **unitário**. Sem levar isso em consideração, quanto mais rápido fosse o computador, maior seria a taxa de atualização e conseqüente maior seria a velocidade dos personagens.

$$speed = fator \left( \frac{1}{framerate} \right)$$

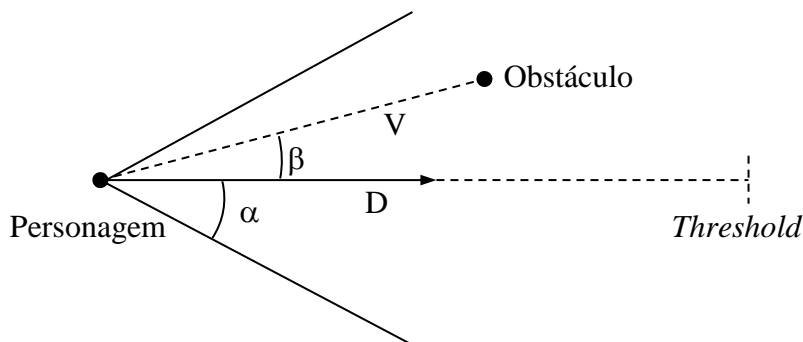
O demo fps, no site da disciplina, ilustra o uso do fps no controle da velocidade da animação. Observe que o controle de fps não é executar a aplicação sempre com o mesmo fps, com o uso de funções do tipo Sleep(), mas sim garantir que a velocidade da animação seja constante, independente se a aplicação está rodando a 30 ou 300 fps.

**O que é controle de FPS?** É garantir que animação execute com a mesma velocidade em qualquer computador, lento ou rápido. Um Sleep() pode ser usado para não animar muito rápido, porém não é suficiente para solucionar animações muito lentas. Tendo-se o tempo de renderização de um quadro, pode-se ajustar a velocidade que a animação irá ocorrer (o passo da animação).

Outro fator que pode modificar o parâmetro *speed* é a consideração das leis físicas, que consideram que um corpo não pode acelerar de forma instantânea. Em um jogo até pode, porém dá um aspecto pouco realista. Deste modo, o valor de *speed* deve aumentar gradativamente assim que o personagem acelera. Obviamente, o crescimento de *speed* não é linear, e obedece a leis físicas, como a lei da aceleração (energia cinética). Por exemplo, um carro em alta velocidade tem aceleração menor do que quando está em baixa velocidade (teste isso no seu carro real, ou na sua bicicleta).

### Campo de visão de um personagem

Para tornar a aquisição de informação do ambiente mais realista (ver seção de agentes), deve-se determinar o campo de visão do personagem. Para isso deve-se definir um ângulo de visão e um *threshold* que determina quão distante o personagem consegue ver. Qualquer objeto ou outro personagem dentro deste campo de visão pode ser percebido, para que o personagem tome decisão da ação que deve realizar.



Neste seguinte algoritmo, não se está considerando a atenuação da percepção com a distância e nem com o ângulo de abertura do obstáculo.

```

if( dist < Threshold_Visao )
{
    Vector3 V = pos_personagem - pos_obstaculo;
    v.normalize();
    float  $\beta$  = D.angulo(V);
    if(  $\beta$  <=  $\alpha$  )
    {
        → O obstáculo esta visível
        → Fazer teste de interseção em caso de escolha de caminho
    }
}

```

### Determinação da linha de tiro

Suponha que um NPC encontre um adversário e precise disparar contra ele. Para isso, o NPC deve se alinhar na direção do alvo e disparar. Para isso, ele deve saber o valor do ângulo  $\beta$  formado entre o seu vetor direção  $D$  e o vetor  $V$ , como mostrado na seguinte figura.  $\alpha = \cos^{-1}(P \cdot Q)$  fornece apenas um ângulo relativo entre dois vetores. Por isso, não se sabe se o NPC deve girar (rotacionar no eixo vertical)  $\beta$  no sentido horário (*clockwise*) ou anti-horário (*counterclockwise*). Para solucionar este problema, verifica-se a posição relativa dos vetores.

```

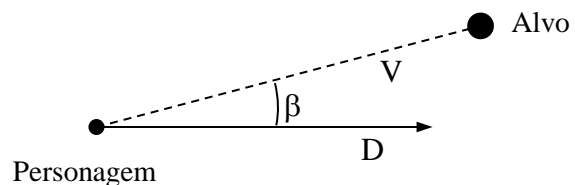
float Vector3::angleSign(const Vector3& v) const
{
    float dp, angPI ;

    dp = dot( v ); //dot product
    if(dp >= 1.0) dp = 1.0f;
    if(dp <=-1.0) dp = -1.0f;

    angPI = acos( dp );

    //teste de lado
    if (z*v.x > x*v.z)
        return angPi;
    else
        return -angPi;
}

```



### Dicas Para definição de entidades móveis

Toda entidade móvel deve ser definida por somente um vetor posição e um vetor direção (ou ângulo). Pode-se também utilizar escalares como velocidade, peso, força, etc. O vetor direção deve sempre ser unitário. Todos estes atributos devem estar dentro da classe que define o veículo. O seguinte fragmento de código ilustra uma forma muito otimizada para fazer o tratamento de movimentação.

```

void render()
{
    //aplica uma rotação no veículo segundo interação do usuário. Ang é global.
    ang = ang + 0.1;

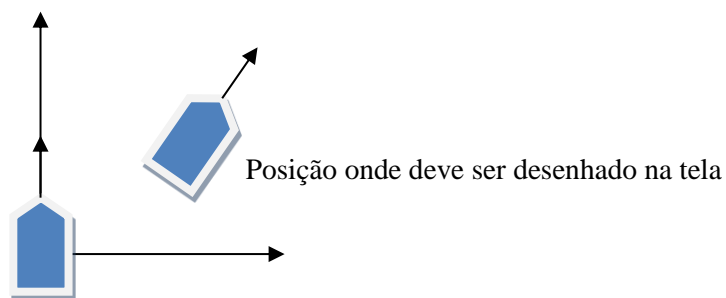
    //define o vetor direção a cada render, em função do ângulo
    float x = cos(ang);
    float y = sin(ang);
    dir.set(x,y); //esse vetor já esta normalizado

    //faz o deslocamento
    pos.x += dir.x * speed;
    pos.y += dir.y * speed;

    //desenha o objeto
    glTranslated(pos.x, pos.y, 0);
    glRotatef(ang * 57.29577, 0, 0, 1); //transforma em graus
    glBegin(GL_....);
    //.....
    glEnd();
}

```

A geometria do veículo deve ser definida em relação à origem e nunca deve ser alterada durante a execução do programa. Desta forma, toda vez que o veículo for desenhado, deve-se fazer uma cópia da geometria, e aplicar sobre ela a translação e rotação necessárias. Se for utilizado a API OpenGL, deve-se somente aplicar transformações sobre a geometria original, sem a necessidade de se criar uma cópia dos vértices que definem sua geometria. Para movimentação de veículos 3D, andando em um plano, pode-se utilizar a mesma formulação.



### 3 Conceitos de Álgebra Linear

**Espaço vetorial:** Conjunto  $V$  de vetores onde somente a adição e multiplicação por escalar estão definidos e deve ter as seguintes propriedades:

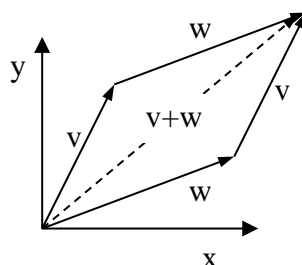
Soma Associativa:  $[(\mathbf{v} + \mathbf{w}) + \mathbf{z} = \mathbf{v} + (\mathbf{w} + \mathbf{z})]$

Soma Comutativa:  $[\mathbf{v} + \mathbf{w} = \mathbf{w} + \mathbf{v}]$

Multiplicação associativa:  $(\alpha\beta)\mathbf{v} = \alpha(\beta\mathbf{v})$

Multiplicação distributiva:  $[(\alpha + \beta)\mathbf{v} = \alpha\mathbf{v} + \beta\mathbf{v} \text{ e } \alpha(\mathbf{v} + \mathbf{w}) = \alpha\mathbf{v} + \alpha\mathbf{w}]$

Ex:  $R^2, R^3$



**Combinação Linear:** para preencher um espaço vetorial pode-se fazer uso de combinações lineares de vetores  $S = \{v_1, v_2, \dots, v_n\}$ , da seguinte forma

$$\alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_n v_n.$$

- A coleção de todas as combinações lineares de um conjunto de vetores é chamado de **varredura** (*span*) do conjunto.
- A varredura (*span*) de um vetor não-nulo é uma linha passando pela origem.
- A varredura de dois vetores não paralelos é um plano.

Um conjunto de vetores é **linearmente dependente** se pelo menos um deles é uma combinação linear dos outros (em outras palavras, se um deles está no *span* dos outros).

Por exemplo, dois vetores  $v_1$  e  $v_2$  colineares (*i.e.* em uma mesma linha) são linearmente dependentes, porque  $v_2 = \alpha v_1$ .

Os vetores  $v_1, v_2, \dots, v_n$  são ditos **linearmente independentes** (LI) se eles não são dependentes.

**Base de um Espaço Vetorial:** conjunto mínimo de varredura deste espaço.

Para o  $\mathbb{R}^3$ , a base pode ser  $\{e_1, e_2, e_3\}$  (base canônica)

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad e_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad e_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Estes vetores são chamados de **base padrão**.

- Qualquer vetor  $v$  de um espaço é escrito de uma maneira única como uma combinação linear de uma base e os coeficientes desta combinação linear são chamados de **coordenadas do vetor  $v$** .
- No caso do  $\mathbb{R}^3$ ,  $v = \alpha_1 e_1 + \alpha_2 e_2 + \alpha_3 e_3$  e as coordenadas de  $v$  são  $[\alpha_1 \ \alpha_2 \ \alpha_3]^T$ .

**Teorema:** uma base de um espaço é sempre linearmente independente.

Qualquer vetor de um espaço vetorial é escrito de maneira única como **combinação Linear** de uma base e os **coeficientes** desta combinação linear são as **coordenadas** do vetor  $v$  nesta base.

**Demonstração que um conjunto de vetores é uma base:**

Se o conjunto  $S = \{e_1, e_2, e_3\}$  é uma base do  $\mathbb{R}^3$ , devemos mostrar que um vetor

$$b = (b_1, b_2, b_3)$$

qualquer pode ser expresso como uma combinação linear

$$b = c_1 v_1 + c_2 v_2 + c_3 v_3$$

dos vetores em  $S$ . Expressando esta equação em termos de componentes, temos

$$\begin{aligned} (b_1, b_2, b_3) &= c_1(1, 0, 0) + c_2(0, 1, 0) + c_3(0, 0, 1) \\ (b_1, b_2, b_3) &= (c_1+0+0, c_2+0+0, c_3+0+0) \end{aligned}$$

Igualando os componentes

$$c_1 + 0 + 0 = b_1$$



$$\begin{aligned}0 + c_2 + 0 &= b_2 \\ 0 + 0 + c_3 &= b_3\end{aligned}$$

Assim, para mostrar que  $S$  gera o  $R^3$  devemos mostrar que o sistema tem solução para qualquer escolha de  $b = (b_1, b_2, b_3)$ . Para mostrar que  $S$  é LI deve-se mostrar que a única solução de

$$c_1v_1 + c_2v_2 + c_3v_3 = 0$$

é  $c_1 = c_2 = c_3 = 0$  (solução trivial).

Para mostrar que o sistema é LI deve-se mostrar que a matriz de coeficientes do sistema tem determinante não-zero.

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \det(A) = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} = 1$$

**Definição:** Se  $V$  é um espaço vetorial qualquer e  $S = \{v_1, v_2, \dots, v_n\}$  é um conjunto de vetores em  $V$ , dizemos que  $S$  é uma base de  $V$  se valerm as seguintes condições:

- $S$  é LI
- $S$  gera  $V$ .

**Unicidade da Representação em Base:** Se  $S = \{v_1, v_2, \dots, v_n\}$  é uma base de um espaço vetorial  $V$ , então cada vetor de  $V$  pode ser expresso da forma  $v = c_1v_1 + c_2v_2 + \dots + c_nv_n$  de uma única maneira. Isso ocorre porque os vetores do conjunto  $S$  são LI.

#### Teoremas

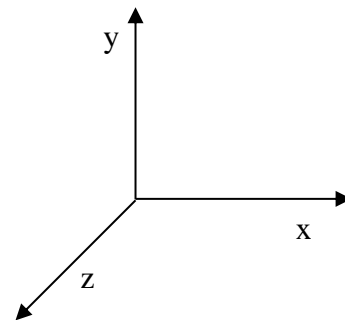
- Seja  $S = \{v_1, v_2, \dots, v_r\}$  um conjunto de vetores em  $R^n$ . Se  $r > n$ , então  $S$  é linearmente dependente. Se  $r < n$ , então  $S$  não gera  $R^n$ .
- Todas as bases de um espaço vetorial de dimensão finita tem o mesmo número de vetores.
- A dimensão de um espaço vetorial de dimensão finita  $V$  é definida como o número de vetores de uma base de  $V$  e denotada por  $\dim(V)$ . O espaço vetorial nulo tem dimensão zero.

#### Espaço Euclidiano 3D

É um espaço vetorial, pois as operações de soma e multiplicação são válidas.

Base:  $\{e_1, e_2, e_3\}$

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad e_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad e_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$



O conjunto  $\{e_1, e_2, e_3\}$  forma uma base **ortonormal**, pois os vetores são ortogonais e com comprimento 1.

**Prova:** Um cálculo direto mostra que  $\langle e_1, e_2 \rangle = \langle e_1, e_3 \rangle = \langle e_2, e_3 \rangle = 0$  e que  $\|e_1\| = \|e_2\| = \|e_3\| = 1$ . Assim  $\{e_1, e_2, e_3\}$  é um conjunto ortonormal. Para todo  $(x, y, z)$  no  $R^3$  temos que  $(x, y, z) = xe_1 + ye_2 + ze_3$ , de forma que  $\{e_1, e_2, e_3\}$  gera (*span*) o  $R^3$  e por isso deve ser uma base.

Deve-se observar também que se esta base for rotacionada na origem em qualquer eixo ou refletida em um plano que passa pela origem forma uma base ortonormal do  $R^3$ .

## Exercícios:

1. Implemente uma classe Vector em C++ para dar suporte a todas as operações sobre vetores, como soma, multiplicação por escalar, produto escalar, etc.
2. Implemente uma classe Matrix em C++ para dar suporte a operações sobre matrizes, como multiplicação, soma, etc.
3. Implemente um relógio analógico com ponteiros para hora/minuto/segundo. Cada ponteiro deve ter uma flecha na extremidade. Deve-se ler a hora do sistema.
4. Implemente um cano de canhão que aponta continuamente na direção do cursor do mouse. Deve-se imprimir na tela o ângulo formado com o eixo x.
5. Implemente o jogo Balls Bounce disponível para Android.



## Referências:

- [1] Lengyel, E.. **Mathematics for 3D Game Programming and Computer Graphics**. Charles River Media, 2002.
- [2] Hearn, D., Baker, M. P. **Computer Graphics, C Version** (2<sup>nd</sup> Edition), Prentice Hall, New Jersey, 1997
- [3] Foley, J.D., van Dam, A., Feiner, S.K. and Hughes, J.F. **Computer Graphics: Principles and Practice in C** (2<sup>nd</sup> Edition), Addison-Wesley Pub. Co., Reading, MA, 1995.
- [4] Buckland, M. **Programming GameIA by Example**. Wordware Publishing inc., Texas, 2005
- [5] Gomes, J., Velho, L. *Computação Gráfica, Volume 1*. IMPA, 1998.
- [6] Howard, A., Horres, C. **Álgebra Linear com Aplicações**. Bookman, 2001.
- [7] Buckland, M. **Programming GameIA by Example**. Wordware Publishing inc., Texas, 2005
- [8] <http://www.ajudamatematica.com/viewtopic.php?f=117&t=271>