Aula 01 - Parte 01 - Vetores

Álgebra Linear e Teoria da Informação

Prof. Tiago Tavares

Vetores e Movimento: Da Matemática ao Código

Nesta aula, vamos explorar como a matemática dos vetores nos ajuda a criar movimento em jogos e simulações, usando PyGame e NumPy.

Lembre-se do loop de jogo:

1. Inicializações
2. Calcular alterações no ambiente
3. Renderizar
4. Voltar ao passo 2

Movimento Ponto a Ponto

No início, podemos controlar um personagem atualizando suas coordenadas X e Y separadamente a cada "quadro" ou iteração do nosso loop.

Em cada passo, a nova posição é a posição antiga mais uma pequena variação.

```
# Posição inicial
x = 200
y = 200

# Variação a cada passo
dx = -1
dy = -1

# Dentro do loop do jogo:
# Atualizamos as posições
x = x + dx # Movimento uniforme!
y = y + dy # Movimento uniforme!
```

Notação Matemática

Essa atualização constante pode ser descrita com uma fórmula matemática. Se "n" é a iteração atual, a posição `x` na iteração `n` depende da posição na iteração anterior, `n-1`.

O código `x = x + dx` é a implementação direta da equação:

$$x_n = x_{n-1} + \Delta x$$

Onde Δx (lido como "delta x") representa a nossa variação `dx`.

Simplificando com Vetores

Gerenciar `x`, `y`, `dx`, `dy` separadamente funciona para 2D, mas e se tivéssemos 3D ou mais dimensões? O código se tornaria repetitivo.

Podemos agrupar as coordenadas em uma única estrutura: um **vetor**. Em Python, usamos os `arrays` do NumPy para isso.

```python
import numpy as np

# Posição inicial como um vetor [x, y]
s0 = np.array([200, 200])

# Velocidade como um vetor [dx, dy]
v = np.array([-1, -1])

# A posição atual também é um vetor
s = s0
```

Notação matemática

Matematicamente (fora de Python), dizemos que os vetores são compostos de elementos:

$$x = [x_1, x_2, x_3, \cdots, x_n]$$

ou então, usando o índice i:

$$x = [x_i]$$

Movimento Vetorial

Com vetores, nosso código de movimento fica bem mais simples - e é bem mais fácil pensar sobre ele, também!

A complexidade de atualizar cada eixo separadamente...

```
x = x + dx
y = y + dy
```

... se resume a uma única linha:

```
s = s + v
```

Isso representa: "a nova posição (`s`) é a posição antiga mais o vetor de velocidade (`v`)".

Código Python e notação matemática

O código:

```
s = s + v
```

É a representação computacional direta da equação vetorial:

$$\boldsymbol{s_n} = \boldsymbol{s_{n-1}} + \boldsymbol{v}$$

Onde $\boldsymbol{s}$ é o vetor de posição e $\boldsymbol{v}$ é o vetor de velocidade.

Como Vetores são Somados?

Quando somamos dois vetores, como em `c = a + b`, o resultado é um novo vetor onde cada elemento é a soma dos elementos correspondentes dos vetores originais.

É uma soma "elemento a elemento".

```
import numpy as np

a = np.array([1, 1, 7])
b = np.array([2, 3, 2])
c = a + b
# c será [1+2, 1+3, 7+2], ou seja, [3, 4, 9]
print(c)
```

Como vetores são somados: notação matemática

Matematicamente, dizemos que: $c_i = a_i + b_i$

Ou, se $c = a + b$, então:

$$c = [a_1 + b_1, a_2 + b_2, \cdots, a_n + b_n]$$

Multiplicação por um Número (Escalar)

Multiplicar um vetor por um número (chamado de "escalar") também funciona elemento a elemento. Esta operação altera a magnitude (tamanho/intensidade) do vetor, mas não sua direção principal.

```python
import numpy as np

a = 0.5
b = np.array([1, 2, 3, 4, 5, 6])
c = a * b
# c será [0.5, 1.0, 1.5, 2.0, 2.5, 3.0]
print(c)
```

Multiplicação por escalar: notação matemática

A fórmula é: $c_i = a \times b_i$. Ou, se $c = ab$, então:

$$c = [ab_1 + ab_2 + \cdots + ab_n]$$

Hoje aprendemos que...

- Vetores nos permitem representar e manipular grandezas multidimensionais (como posição e velocidade) de forma simples e intuitiva.
- A implementação em `NumPy` (`s = s + v`) reflete diretamente a notação matemática ($\boldsymbol{s_n} = \boldsymbol{s_{n-1}} + \boldsymbol{v}$).
- Com operações simples como soma de vetoriais e multiplicação por escalar, podemos criar comportamentos diversificados, como um ponto que segue um alvo.