Geometria Computacional

Vetores: definição

Prof. Edson Alves

Faculdade UnB Gama

Vetores

Definição de vetores

- Vetores são segmentos de retas orientados

Definição de vetores

- Vetores são segmentos de retas orientados
- Os vetores são caracterizados pela sua
    1. direção, isto é, a inclinação da reta que contém o segmento;

Definição de vetores

- Vetores são segmentos de retas orientados
- Os vetores são caracterizados pela sua
  1. direção, isto é, a inclinação da reta que contém o segmento;
  2. orientação, a partir da indicação do ponto de partida e do ponto de chegada; e

Definição de vetores

- Vetores são segmentos de retas orientados
- Os vetores são caracterizados pela sua
    1. direção, isto é, a inclinação da reta que contém o segmento;
    2. orientação, a partir da indicação do ponto de partida e do ponto de chegada; e
    3. tamanho, ou seja, a distância entre os dois pontos

Definição de vetores

- Vetores são segmentos de retas orientados
- Os vetores são caracterizados pela sua
  1. direção, isto é, a inclinação da reta que contém o segmento;
  2. orientação, a partir da indicação do ponto de partida e do ponto de chegada; e
  3. tamanho, ou seja, a distância entre os dois pontos
- Dois vetores são iguais apenas se coincidirem nestas três características

Definição de vetores

- Vetores são segmentos de retas orientados
- Os vetores são caracterizados pela sua
  1. direção, isto é, a inclinação da reta que contém o segmento;
  2. orientação, a partir da indicação do ponto de partida e do ponto de chegada; e
  3. tamanho, ou seja, a distância entre os dois pontos
- Dois vetores são iguais apenas se coincidirem nestas três características
- Dados dois pontos A e B, $\vec{u} = \overrightarrow{AB}$ é o vetor que parte do ponto A em direção ao ponto B

Definição de vetores

- Vetores são segmentos de retas orientados
- Os vetores são caracterizados pela sua
  1. direção, isto é, a inclinação da reta que contém o segmento;
  2. orientação, a partir da indicação do ponto de partida e do ponto de chegada; e
  3. tamanho, ou seja, a distância entre os dois pontos
- Dois vetores são iguais apenas se coincidirem nestas três características
- Dados dois pontos A e B, $\vec{u} = \overrightarrow{AB}$ é o vetor que parte do ponto A em direção ao ponto B
- Observe que $\overrightarrow{AB}$ e $\overrightarrow{BA}$ tem mesma direção e comprimento, mas orientações distintas

Vetor posição

- O vetor posição de um ponto P é o vetor que une a origem O ao ponto P ($\overrightarrow{OP}$)

Vetor posição

- O vetor posição de um ponto P é o vetor que une a origem O ao ponto P ($\overrightarrow{OP}$)
- Na prática, trabalha-se apenas com vetores-posição: o vetor posição que equivale ao vetor $\overrightarrow{AB}$ é o vetor $\vec{v} = (x_b - x_a, y_b - y_a)$

Vetor posição

- O vetor posição de um ponto P é o vetor que une a origem O ao ponto P ($\overrightarrow{OP}$)
- Na prática, trabalha-se apenas com vetores-posição: o vetor posição que equivale ao vetor $\overrightarrow{AB}$ é o vetor $\vec{v} = (x_b - x_a, y_b - y_a)$
- Deste modo, embora seja possível definir um tipo de dado para representar vetores, é possível utilizar pontos para representar vetores

Vetor posição

- O vetor posição de um ponto P é o vetor que une a origem O ao ponto P ($\overrightarrow{OP}$)
- Na prática, trabalha-se apenas com vetores-posição: o vetor posição que equivale ao vetor $\overrightarrow{AB}$ é o vetor $\vec{v} = (x_b - x_a, y_b - y_a)$
- Deste modo, embora seja possível definir um tipo de dado para representar vetores, é possível utilizar pontos para representar vetores
- Esta estratégia pode dificultar a leitura das rotinas, pois embora usem a mesma estrutura, a semântica é diferente

Vetor posição

- O vetor posição de um ponto P é o vetor que une a origem O ao ponto P ($\overrightarrow{OP}$)
- Na prática, trabalha-se apenas com vetores-posição: o vetor posição que equivale ao vetor $\overrightarrow{AB}$ é o vetor $\vec{v} = (x_b - x_a, y_b - y_a)$
- Deste modo, embora seja possível definir um tipo de dado para representar vetores, é possível utilizar pontos para representar vetores
- Esta estratégia pode dificultar a leitura das rotinas, pois embora usem a mesma estrutura, a semântica é diferente
- Há, porém, a vantagem da velocidade de codificação, devido a eliminação de código redundante

Exemplo de implementação de vetores em C++

```cpp
template<typename T>
struct Vector
{
    T x = 0, y = 0;

    Vector(const Point<T>& A, const Point<T>& B)
        : x(B.x - A.x), y(B.y - A.y) {}
};
```

Direção de um vetor

- A direção de um vetor pode ser caracterizada também pelo ângulo que o vetor posição equivalente faz com o eixo-x positivo

Direção de um vetor

- A direção de um vetor pode ser caracterizada também pelo ângulo que o vetor posição equivalente faz com o eixo-x positivo
- Este ângulo pode ser computado pela função `atan2()` da biblioteca `cmath`

Direção de um vetor

- A direção de um vetor pode ser caracterizada também pelo ângulo que o vetor posição equivalente faz com o eixo-x positivo
- Este ângulo pode ser computado pela função `atan2()` da biblioteca `cmath`
- Esta função recebe dois parâmetros: a coordenada y e a coordenada x do vetor posição

Direção de um vetor

- A direção de um vetor pode ser caracterizada também pelo ângulo que o vetor posição equivalente faz com o eixo-x positivo
- Este ângulo pode ser computado pela função `atan2()` da biblioteca `cmath`
- Esta função recebe dois parâmetros: a coordenada y e a coordenada x do vetor posição
- Esta função não lança exceções e nem erros, e tem retorno no intervalo $[-\pi, \pi]$

Direção de um vetor

- A direção de um vetor pode ser caracterizada também pelo ângulo que o vetor posição equivalente faz com o eixo-x positivo
- Este ângulo pode ser computado pela função `atan2()` da biblioteca `cmath`
- Esta função recebe dois parâmetros: a coordenada y e a coordenada x do vetor posição
- Esta função não lança exceções e nem erros, e tem retorno no intervalo $[-\pi, \pi]$
- A função `atan()` difere no número de argumentos (um único) e no intervalo do retorno ($[-\pi/2, \pi/2]$, ou $-\infty$, ou ∞, ou NaN)

Translações

- Um ponto P pode ser transladado no espaço, conhecidos os deslocamentos dx e dy nas direções paralelas aos eixos x e y, respectivamente

Translações

- Um ponto P pode ser transladado no espaço, conhecidos os deslocamentos dx e dy nas direções paralelas aos eixos x e y, respectivamente
- Transladar ambos pontos que delimitam o vetor mantém o vetor inalterado

Translações

- Um ponto P pode ser transladado no espaço, conhecidos os deslocamentos dx e dy nas direções paralelas aos eixos x e y, respectivamente
- Transladar ambos pontos que delimitam o vetor mantém o vetor inalterado
- Contudo, transladar apenas o ponto final P de um vetor posição pode alterar todas as três características de um vetor

Translações

- Um ponto P pode ser transladado no espaço, conhecidos os deslocamentos dx e dy nas direções paralelas aos eixos x e y, respectivamente
- Transladar ambos pontos que delimitam o vetor mantém o vetor inalterado
- Contudo, transladar apenas o ponto final P de um vetor posição pode alterar todas as três características de um vetor

```cpp
template<typename T>
Point<T> translate(const Point<T>& P, T dx, T dy)
{
    return { P.x + dx, P.y + dy };
}
```

Rotações

- Um vetor posição pode ser rotacionado em θ graus no sentido anti-horário através da multiplicação da matriz de rotação R_θ e o vetor $\vec{v}$, onde

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \quad \vec{v} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotações

- Um vetor posição pode ser rotacionado em θ graus no sentido anti-horário através da multiplicação da matriz de rotação R_θ e o vetor $\vec{v}$, onde

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \quad \vec{v} = \begin{bmatrix} x \\ y \end{bmatrix}$$

- Esta matriz pode ser deduzida observando-se que as coordenadas do ponto P do vetor posição $\vec{v}$ podem ser expressas como

$$x = r\cos\omega, \quad y = r\sin\omega,$$

onde r é o tamanho do vetor $\vec{v}$ e ω é o ângulo que $\vec{v}$ faz com o eixo-x positivo

Rotações

- Um vetor posição pode ser rotacionado em θ graus no sentido anti-horário através da multiplicação da matriz de rotação R_θ e o vetor $\vec{v}$, onde

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \quad \vec{v} = \begin{bmatrix} x \\ y \end{bmatrix}$$

- Esta matriz pode ser deduzida observando-se que as coordenadas do ponto P do vetor posição $\vec{v}$ podem ser expressas como

$$x = r\cos\omega, \quad y = r\sin\omega,$$

onde r é o tamanho do vetor $\vec{v}$ e ω é o ângulo que $\vec{v}$ faz com o eixo-x positivo

- Assim, as coordenadas do ponto resultante da rotação são

$$x' = r\cos(\omega + \theta), \quad y' = r\sin(\omega + \theta)$$

Rotações

- Utilizando as fórmulas para soma de ângulos do seno e do cosseno obtemos

$$x' = r\cos\omega\cos\theta - r\sin\omega\sin\theta$$

e

$$y' = r\sin\omega\cos\theta + r\cos\omega\sin\theta,$$

o que corresponde ao resultado do produto matricial já citado

Rotações

- Utilizando as fórmulas para soma de ângulos do seno e do cosseno obtemos

$$x' = r\cos\omega\cos\theta - r\sin\omega\sin\theta$$

e

$$y' = r\sin\omega\cos\theta + r\cos\omega\sin\theta,$$

o que corresponde ao resultado do produto matricial já citado

```cpp
template<typename T>
Point<T> rotate(const Point<T>& P, T angle)
{
    auto x = cos(angle) * P.x - sin(angle) * P.y;
    auto y = sin(angle) * P.x + cos(angle) * P.y;

    return { x, y };
}
```

Rotação em torno de um ponto arbitrário

- Caso se deseje rotacionar o ponto P em torno de outro ponto C que não seja a origem (mais precisamente, outro eixo paralelo ao eixo-z que passe pelo ponto dado), basta seguir os três passos abaixo:

Rotação em torno de um ponto arbitrário

- Caso se deseje rotacionar o ponto P em torno de outro ponto C que não seja a origem (mais precisamente, outro eixo paralelo ao eixo-z que passe pelo ponto dado), basta seguir os três passos abaixo:
    1. transladar o ponto com deslocamentos iguais aos simétricos das coordenadas de C, obtendo-se o ponto P'

Rotação em torno de um ponto arbitrário

- Caso se deseje rotacionar o ponto P em torno de outro ponto C que não seja a origem (mais precisamente, outro eixo paralelo ao eixo-z que passe pelo ponto dado), basta seguir os três passos abaixo:
    1. transladar o ponto com deslocamentos iguais aos simétricos das coordenadas de C, obtendo-se o ponto P'
    2. rotacionar o ponto transladado P'

Rotação em torno de um ponto arbitrário

- Caso se deseje rotacionar o ponto P em torno de outro ponto C que não seja a origem (mais precisamente, outro eixo paralelo ao eixo-z que passe pelo ponto dado), basta seguir os três passos abaixo:
  1. transladar o ponto com deslocamentos iguais aos simétricos das coordenadas de C, obtendo-se o ponto P'
  2. rotacionar o ponto transladado P'
  3. transladar P', com deslocamentos iguais às coordenadas de C

Rotação em torno de um ponto arbitrário

- Caso se deseje rotacionar o ponto P em torno de outro ponto C que não seja a origem (mais precisamente, outro eixo paralelo ao eixo-z que passe pelo ponto dado), basta seguir os três passos abaixo:
    1. transladar o ponto com deslocamentos iguais aos simétricos das coordenadas de C, obtendo-se o ponto P'
    2. rotacionar o ponto transladado P'
    3. transladar P', com deslocamentos iguais às coordenadas de C
- A translação inicial muda o sistema de coordenadas do problema, o levando a um novo sistema onde C é a origem

Rotação em torno de um ponto arbitrário

- Caso se deseje rotacionar o ponto P em torno de outro ponto C que não seja a origem (mais precisamente, outro eixo paralelo ao eixo-z que passe pelo ponto dado), basta seguir os três passos abaixo:
  1. transladar o ponto com deslocamentos iguais aos simétricos das coordenadas de C, obtendo-se o ponto P'
  2. rotacionar o ponto transladado P'
  3. transladar P', com deslocamentos iguais às coordenadas de C
- A translação inicial muda o sistema de coordenadas do problema, o levando a um novo sistema onde C é a origem
- Assim, pode-se utilizar a rotina de rotação em torno da origem e, ao final do processo, retornar ao sistema original, aplicando a translação inversa

Rotação em torno de um ponto arbitrário

- Caso se deseje rotacionar o ponto P em torno de outro ponto C que não seja a origem (mais precisamente, outro eixo paralelo ao eixo-z que passe pelo ponto dado), basta seguir os três passos abaixo:
    1. transladar o ponto com deslocamentos iguais aos simétricos das coordenadas de C, obtendo-se o ponto P'
    2. rotacionar o ponto transladado P'
    3. transladar P', com deslocamentos iguais às coordenadas de C
- A translação inicial muda o sistema de coordenadas do problema, o levando a um novo sistema onde C é a origem
- Assim, pode-se utilizar a rotina de rotação em torno da origem e, ao final do processo, retornar ao sistema original, aplicando a translação inversa
- Importante notar que as três operações devem ser realizadas exatamente na ordem descrita

Implementação da rotação em torno de um ponto arbitrário

```cpp
template<typename T>
Point<T> rotate(const Point<T>& P, T angle, const Point<T>& C)
{
    auto Q = translate(P, -C.x, -C.y);
    Q = rotate(Q, angle);
    Q = translate(Q, C.x, C.y);

    return Q;
}
```

Rotações tridimensionais

- A mesma ideia da rotação pode ser aplicada em pontos tridimensionais

Rotações tridimensionais

- A mesma ideia da rotação pode ser aplicada em pontos tridimensionais
- As matrizes R_x, R_y e R_z abaixo rotacionam o ponto tridimensional $P = (x_p, y_p, z_p)$ em θ graus no sentido anti-horário

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \quad R_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Observações sobre translações e rotações

- Dado um conjunto de pontos $\mathcal{A}$, se aplicadas a todos pontos $P \in \mathcal{A}$, as operações de translação e rotação não alteram as distâncias entre os pares de pontos

Observações sobre translações e rotações

- Dado um conjunto de pontos $\mathcal{A}$, se aplicadas a todos pontos $P \in \mathcal{A}$, as operações de translação e rotação não alteram as distâncias entre os pares de pontos
- Desta forma, se uma figura é descrita por um conjunto de pontos, todas as suas características que são baseadas em distâncias (ângulos internos, perímetro, área, volume, etc) são invariantes a estas duas transformações

Observações sobre translações e rotações

- Dado um conjunto de pontos $\mathcal{A}$, se aplicadas a todos pontos $P \in \mathcal{A}$, as operações de translação e rotação não alteram as distâncias entre os pares de pontos
- Desta forma, se uma figura é descrita por um conjunto de pontos, todas as suas características que são baseadas em distâncias (ângulos internos, perímetro, área, volume, etc) são invariantes a estas duas transformações
- Este importante fato pode ser utilizado para simplificar problemas, como exemplificado no caso da rotação em torno de um ponto arbitrário

Escala

- Outra transformação possível de um vetor posição é a escala

Escala

- Outra transformação possível de um vetor posição é a escala
- A escala consiste na multiplicação de cada componente v_i de um vetor por um determinado escalar s_i

Escala

- Outra transformação possível de um vetor posição é a escala
- A escala consiste na multiplicação de cada componente v_i de um vetor por um determinado escalar s_i
- Se o mesmo escalar é utilizado em todos as componentes a escala é dita uniforme

Escala

- Outra transformação possível de um vetor posição é a escala
- A escala consiste na multiplicação de cada componente v_i de um vetor por um determinado escalar s_i
- Se o mesmo escalar é utilizado em todos as componentes a escala é dita uniforme
- Ao contrário das transformações anteriores, a escala não preserva distâncias

Escala

- Outra transformação possível de um vetor posição é a escala
- A escala consiste na multiplicação de cada componente v_i de um vetor por um determinado escalar s_i
- Se o mesmo escalar é utilizado em todos as componentes a escala é dita uniforme
- Ao contrário das transformações anteriores, a escala não preserva distâncias

```cpp
template<typename T>
Vector<T> scale(const Vector<T>& v, T sx, T sy)
{
    return { sx * v.x, sy * v.y };
}
```

Normalização de vetores

- Uma aplicação comum da escala é a normalização de vetor

Normalização de vetores

- Uma aplicação comum da escala é a normalização de vetor
- Um vetor é dito unitário se o seu comprimento é igual a 1

Normalização de vetores

- Uma aplicação comum da escala é a normalização de vetor
- Um vetor é dito unitário se o seu comprimento é igual a 1
- Dado um vetor $\vec{v}$ qualquer, é possível determinar um vetor unitário $\vec{u}$, na mesma direção e sentido de $\vec{v}$, dividindo-se as coordenadas de $\vec{v}$ pelo tamanho $|\vec{v}|$ de $\vec{v}$

Normalização de vetores

- Uma aplicação comum da escala é a normalização de vetor
- Um vetor é dito unitário se o seu comprimento é igual a 1
- Dado um vetor $\vec{v}$ qualquer, é possível determinar um vetor unitário $\vec{u}$, na mesma direção e sentido de $\vec{v}$, dividindo-se as coordenadas de $\vec{v}$ pelo tamanho $|\vec{v}|$ de $\vec{v}$
- Observe que a escala com constantes positivas preserva a direção e o sentido do vetor

Normalização de vetores

- Uma aplicação comum da escala é a normalização de vetor
- Um vetor é dito unitário se o seu comprimento é igual a 1
- Dado um vetor $\vec{v}$ qualquer, é possível determinar um vetor unitário $\vec{u}$, na mesma direção e sentido de $\vec{v}$, dividindo-se as coordenadas de $\vec{v}$ pelo tamanho $|\vec{v}|$ de $\vec{v}$
- Observe que a escala com constantes positivas preserva a direção e o sentido do vetor

```cpp
template<typename T>
Vector<T> normalize(const Vector<T>& v)
{
    auto len = v.length();
    return { v.x / len, v.y / len };
}
```

Referências

1. **HALIM**, Felix; **HALIM**, Steve. *Competitive Programming 3*, 2010.
2. **LAAKSONEN**, Antti. *Competitive Programmer's Handbook*, 2018.
3. **De BERG**, Mark; **CHEONG**, Otfried. *Computational Geometry: Algorithms and Applications*, 2008.