MAC02166 - 1o semestre 2022

Segundo Exercício-Programa

Introdução

Neste EP2, vocês terão uma introdução à programação do núcleo de um Jogo de Computador (ou seja, sem a interface), no nível mais básico de entrada no jogo. Para isso, vocês deverão implementar um sistema para a captura de pokémons, simulando a trajetória de pokébolas num movimento bidimensional (X e Y) sem atrito, e supondo que o pokémon fica paradinho durante todo esse processo.

No início da simulação, o treinador possui 3 pokébolas, que poderão ser usadas na captura. Para tratar tamanhos de pokémons diferentes, o pokémon deve ser modelado como um círculo de raio r centrado em (x_p,y_p) e imóvel. Dadas as posições iniciais do treinador (x_T, y_T), a velocidade escalar e o ângulo de inclinação do lançamento da pokébola, o programa deverá calcular, em intervalos de tempo Δt, a trajetória da pokébola ($x_b(t_i)$, $y_b(t_i)$), e determinar se a mesma captura o pokémon ou não. Caso a pokébola não capture o pokémon, o usuário poderá digitar novos valores de lançamento e repetir a simulação novamente, até acabarem as pokébolas. A *captura do pokémon* pela pokébola ocorre quando o círculo da pokébola de centro (x_b, y_b) e raio r_b intersecta o círculo do pokémon de centro (x_p, y_p) e raio r_p; ou seja, quando a distância entre os dois centros é menor ou igual à soma dos raios. Diz-se que a *pokébola atinge o chão* quando a coordenada y da pokébola é menor ou igual ao raio r_b da pokébola, devido a uma falha na captura do pokémon.

A simulação deve ser feita calculando-se a posição da pokébola, $(x_b(t_i), y_b(t_i))$, a cada instante de tempo t, com granularidade de Δt segundos. O cálculo da posição da pokébola, $(x_b(t_i), y_b(t_i))$ deve ser feito de forma iterativa, isto é, usando-se a posição e velocidades no instante anterior $t - \Delta t$, segundo as fórmulas abaixo:

$$t_{i+1} = t_i + \Delta t$$

$$x_b(t_{i+1}) = x_b(t_i) + v_{x_b}(t_i) * \Delta t$$

$$y_b(t_{i+1}) = y_b(t_i) + v_{y_b}(t_i) * \Delta t - \frac{g * \Delta t^2}{2}$$

$$v_{x_b}(t_{i+1}) = v_{x_b}(t_i)$$

$$v_{y_b}(t_{i+1}) = v_{y_b}(t_i) - g * \Delta t$$

$$(x_b(0), y_b(0)) = (x_T, y_T)$$

sendo g o valor da atração gravitacional e $v_b(t) = (v_{x_b}(t), v_{y_b}(t))$ a velocidade da pokébola.

A figura 1 ilustra o problema com as equações. Note que os valores da componente y de velocidade também são atualizados. No eixo x assumiremos que o movimento é uniforme, e no eixo y, uniformemente variado. Assumiremos que a gravidade não varia nem com a altura, nem com a latitude e nem com o tempo.

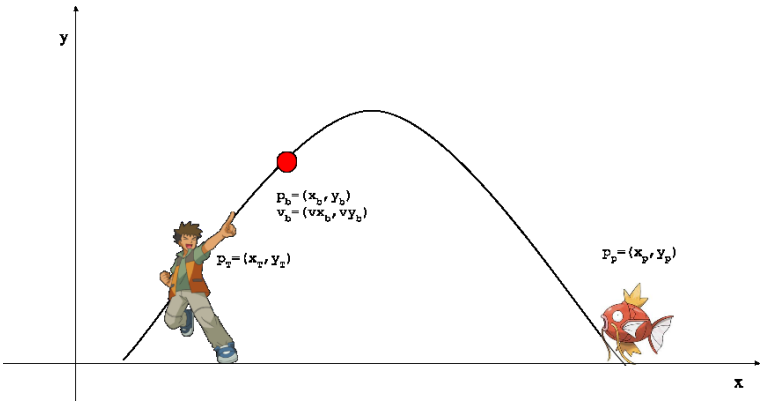

Figura 1: Ilustração do problema geral de captura de pokémons.

Os dados do lançamento da pokébola são dados pela velocidade escalar e o ângulo de inclinação do lançamento em graus. As componentes horizontal e vertical $v_{x_b}(0)$ e $v_{y_b}(0)$ da velocidade inicial da pokébola podem ser calculadas através das fórmulas:

$v_{y_b}(0) = v * seno(\theta)$

$v_{x_b}(0) = v * cosseno(\theta)$

onde v é a velocidade escalar, θ o ângulo de inclinação em graus, e $v_{x_b}(0)$ e $v_{y_b}(0)$ os componentes x e y da velocidade inicial da pokébola.

Funcionamento do programa

O programa deve ler inicialmente as coordenadas `x_pokemon` e `y_pokemon` do pokémon, o seu raio `rp`, o raio da pokébola e a granularidade `delta_t`. O treinador tem no máximo três chances para capturar o pokémon. Para cada tentativa, o programa deve ler as coordenadas iniciais `x_pokebola` e `y_pokebola`, a velocidade `v_lancamento` e o ângulo `angulo_lancamento` de lançamento.

O programa deverá calcular a posição da pokébola a cada intervalo `delta_t`.

A simulação deverá ser executada até que uma das condições seja verificada:

1. A pokébola atinge o chão, i.e., a posição `y_pokebola` é menor ou igual ao raio da pokébola;
2. A pokébola captura o pokémon.

No final de cada tentativa, seu programa deve imprimir uma mensagem indicando se o pokémon foi capturado ou não. A *distância entre a pokébola e o pokémon* é a própria distância entre o círculo da pokébola de centro (x_b, y_b) e raio r_b e o círculo do pokémon de centro (x_p, y_p) e raio r_p: zero, caso se intersectem; a distância entre os dois centros menos a soma dos dois raios, caso contrário. No caso de não ocorrer captura ao longo da trajetória, o programa também deve imprimir a menor distância do pokémon à pokébola ao longo de sua trajetória, bem como as coordenadas da posição em que a pokébola fica mais próxima.

Funções exigidas

Para a organização do programa, as seguintes funções foram propostas e devem ser implementadas com as respectivas assinaturas:

```
def seno(theta):
    '''
    Esta função aproxima o valor da função seno para o ângulo theta
    usando a série de Taylor até que o módulo do próximo termo da
    série calculada seja menor 1e-10.
    Entrada: O ângulo theta que deve ser informado em graus.
    Saída: A aproximação do seno do ângulo theta.
    '''

def cosseno(theta):
    '''
    Esta função aproxima o valor da função cosseno para o ângulo theta
    usando a série de Taylor até que o módulo do próximo termo da
    série calculada seja menor 1e-10.
    Entrada: O angulo theta que deve ser informado em graus.
    Saída: A aproximação do cosseno do ângulo theta.
    '''

def raizQuadrada(x):
    '''
    Esta função aproxima o valor da raiz quadrada de x, através da
    fórmula de recorrência r_0 = x e r_{n+1} = 1/2 (r_n+ x/r_n)
    enquanto o módulo da diferença entre os dois últimos valores
    calculados for maior que 1e-10.
    Entrada: O valor de x
    Saída: A aproximação da raiz quadrada de x.
    '''

def atualizaPosicao(x, y, vx, vy, dt):
    '''
    Esta função calcula as atualizações das posições de x e y usando
    as velocidades escalares respectivamente dadas por vx e vy.
    Entrada: As posições x e y dadas em metros, as velocidades vx e
    vy em metros por segundo e o intervalo de tempo em segundos.
    Saída: Dois valores: o valor atualizado de x e o valor atualizado de y.
    '''

def atualizaVelocidade(vx, vy, dt):
    '''
    Esta função calcula e atualiza as velocidades vx e vy para o
    próximo intervalo de tempo.
    Entrada: As velocidades vx e vy em metros por segundo e o
    intervalo de tempo em segundos.
    Saída: Dois valores: o valor atualizado de vx e o valor atualizado de vy.
    '''

def distanciaPontos(x1, y1, x2, y2):
    '''
    Esta função calcula a distância entre dois pontos (x1, y1) e (x2, y2).
    Entrada: As coordenadas dos pontos do plano (x1, y1) e (x2, y2).
    Saída: A distância entre (x1, y1) e (x2, y2).
    '''
```

```
def simulaLancamento (xpokebola, ypokebola, rb,
                      vlancamento, angulolancamento,
                      xpokemon, ypokemon, rp,
                      delta_t):
    '''
    Esta função simula o lançamento da bola até que ela capture o
    pokemon, ou atinja o chão.
    Entrada: Posição inicial da pokebola (xpokebola e ypokebola), em metros;
    Posição do pokemon (xpokemon e ypokemon), em metros;
    Velocidade escalar, em metros por segundo;
    e ângulo de lançamento, em graus;
    Os raios rb e rp, em metros;
    a granularidade de tempo delta_t usada na simulação, em segundos.
    Saída: Um booleano (True se o lançamento teve sucesso e acertou o
    pokémon, ou False caso contrário), a menor distância do pokémon à
    pokébola e as coordenadas x e y da pokébola nesta posição mais próxima.
        '''
```

As seguintes constantes devem ser definidas e usadas em seu programa:

```
GRAVIDADE = 9.81
PI = 3.14159265358979323846
```

As seguintes funções e operações podem ser usadas:

```
abs
**
```

ATENÇÃO:

1. O ângulo de lançamento pode ser maior do que 90^o, ou seja, a bola pode ser lançada para trás. Podemos supor que o usuário irá digitar um ângulo entre 0 e 180 graus e que 0 significa para a direita (sentido em que a coordenada x aumenta) e 180 significa para a esquerda (sentido em que a coordenada x diminui).

2. A velocidade de lançamento será sempre positiva (> 0).

3. O ângulo é fornecido em graus, mas as séries do seno e do cosseno trabalham com o ângulo em radianos. Para a conversão, utilize a fórmula $\theta_{rad} = \theta_{graus} * \Pi/180$. O valor de Π a ser utilizado é o da constante `PI`.

4. As funções seno e cosseno devem ser implementadas como a soma dos termos das seguintes séries, onde x é o ângulo dado **em radianos**:

   $seno(x) = x - x^3/3! + x^5/5! - x^7/7! + \ldots$

   $cosseno(x) = 1 - x^2/2! + x^4/4! - x^6/6! + \ldots$

5. Para o cálculo das distâncias, você pode considerar a aproximação da raiz quadrada de x dada pela seguinte recorrência, que só pode ser usada quando $x > 0$:

   $r_0 = x$

   $r_{n+1} = 1/2(r_n + x/r_n)$

Observações:

- Você deve utilizar *TODAS* as funções sugeridas.
- Você pode criar outras funções e constantes caso sinta necessidade.

Exemplos de execução:

(os números pintados de vermelho foram digitados pelo usuário)

```
Digite a coordenada x do pokemon:  10
Digite a coordenada y do pokemon:  0
Digite o raio do pokemon (> 0) em metros:  0.12
Digite o raio da pokebola (> 0) em metros:  0.03
Digite a granularidade de tempo da simulacao em segundos:  0.01

Tentativa 1
    Digite a coordenada x do treinador:  0
    Digite a coordenada y do treinador:  0
    Digite a velocidade de lancamento em m/s:  11
    Digite o angulo de lancamento em graus:  45

A pokebola nao captura o pokemon por 1.4034 metros, ao passar em 10.9672,1.2156.

Tentativa 2
    Digite a coordenada x do treinador:  0
    Digite a coordenada y do treinador:  0
    Digite a velocidade de lancamento em m/s:  9
    Digite o angulo de lancamento em graus:  45

A pokebola nao captura o pokemon por 1.5769 metros, ao passar em 8.2731,-0.0163.

Tentativa 3
    Digite a coordenada x do treinador:  0
    Digite a coordenada y do treinador:  0
    Digite a velocidade de lancamento em m/s:  10
    Digite o angulo de lancamento em graus:  45

A pokebola captura o pokemon.
```

```
Digite a coordenada x do pokemon:  10
Digite a coordenada y do pokemon:  5
Digite o raio do pokemon (> 0) em metros:  0.12
Digite o raio da pokebola (> 0) em metros:  0.03
Digite a granularidade de tempo da simulacao em segundos:  0.01

Tentativa 1
    Digite a coordenada x do treinador:  0
    Digite a coordenada y do treinador:  0
    Digite a velocidade de lancamento em m/s:  12
    Digite o angulo de lancamento em graus:  45

A pokebola nao captura o pokemon por 1.5724 metros, ao passar em 9.5035,3.3507.

Tentativa 2
    Digite a coordenada x do treinador:  0
    Digite a coordenada y do treinador:  0
    Digite a velocidade de lancamento em m/s:  14
    Digite o angulo de lancamento em graus:  45

A pokebola captura o pokemon.
```