MAC0115 - IFUSP - 2o semestre 2020

Segundo Exercício-Programa

Críticas ao projeto anterior

Noventa dias após a submissão de seu projeto inicial da FAPPESP, seu projeto foi aprovado com ressalvas. A proposta foi elogiada pois aborda importante problema da área de captura de pokémons, mas há falhas metodológicas que, se não forem abordadas e corrigidas na próxima versão do projeto, comprometerão a qualidade do resultado final.

1. Foi questionado o uso de valores inteiros para representar a posição e velocidade da pokébola, do treinador e do pokémon. Apesar de permitir precisão de até um metro na simulação, o que funcionaria bem para pokémons grandes como Snorlax que tem 2.11 m de altura[1], a captura de menores como o Pidgey (30cm) não seriam simulados corretamente. A sugestão foi o uso de variáveis de ponto flutuante para aumento de precisão.

2. Em particular, o Δt (intervalo de tempo entre os instantes simulados) poderia ser de frações de segundo.

3. Foi sugerido também que deve-se considerar também tamanhos de pokémons diferentes. O parecer sugere modelar um pokémon como um círculo de raio r centrado em (x_p,y_p).

4. Nesse sentido, sugere-se que a colisão entre pokébola e pokémon seja definida como a passagem do centro da pokébola por dentro do círculo de centro (x_p, y_p) e raio r; haverá colisão entre pokébola e chão quando a coordenada y da pokébola for menor que um valor ϵ (EPSILON).

5. Foi criticada a escolha de colocar como entrada os componentes x e y da velocidade da pokébola - um lançador não teria acesso a estas variáveis, e sim à velocidade escalar e ao ângulo de inclinação. O avaliador observou que o programa deve ser modificado para aceitar a velocidade escalar e o ângulo de inclinação do lançamento em graus, mas que o uso dos componentes da velocidade de fato simplificam o algoritmo, e que podem ser convertidos através das fórmulas:

   $v_y = v * seno(\theta)$

   $v_x = v * cosseno(\theta)$

   onde v é a velocidade escalar, θ o ângulo de inclinação em graus, e v_x e v_y os componentes x e y da velocidade da pokébola.

6. Quanto ao formato do programa, a escolha de programar tudo apenas em uma função `main()`, ou diretamente dentro do arquivo (.py) foi duramente criticada pois leva a problemas de manutenção, tais como: dificuldade

[1] http://bulbapedia.bulbagarden.net/wiki/Snorlax

de reaproveitamento de código e possibilidade de utilização indevida de variáveis. Embora como protótipo o formato é aceitável, pois demonstra a funcionalidade desejada, não é provável que um projeto com escopo maior seja facilmente desenvolvido sem modularizar o código em funções.

7. Por último, foi estipulado que apresentar toda a trajetória da bola poderia confundir o treinador para intervalos de tempo pequenos como os desejados, então a ferramenta deveria mostrar apenas se o pokémon foi pego ou não.

8. Apesar da flexibilidade do programa no tocante à aceleração da gravidade (simulando captura de pokémons em outros planetas) ter sido elogiada, sugeriu-se que o valor da aceleração da gravidade fosse definida como constante e igual a 9.81 m/s^2 , simplificando o programa. Vislumbrou-se a possibilidade de captura de pokémons na próxima missão com veículo motorizado (Rover) a Marte ou Europa.

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 angulo 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=DELTA_T):
    '''
    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=DELTA_T):
    '''
    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 dados por
    (x1, y1) e (x2, y2).
    Entrada: As coordenadas de dois pontos no plano, x1, y1, x2, y2,
    em metros.
    Saída: A distância entre (x1, y1) e (x2, y2).
    '''

def houveColisao(xpokebola, ypokebola, xpokemon, ypokemon, r):
    '''
    Esta função calcula se houve ou não colisão entre a pokebola e o
    pokemon considerando-se um raio r.
    Entrada: posição x e y da pokebola, posição x e y do pokemon
             e o raio r, todas medidas em metros.
    Saída: Retorna True caso haja colisão, e False caso contrário.
    '''

def simula_lancamento (xpokebola, ypokebola,
                       vlancamento, angulolancamento,
                       xpokemon, ypokemon, r):
    '''
    Esta função simula o lançamento da bola até que ela atinja o
    pokemon, ou o solo a menos de EPSILON.
    Na simulação, considere as seguintes constantes:
    EPSILON é uma constante de precisão de 1.0e-2 metro.
    DELTA_T é uma constante de precisão de 1.0e-2 segundo.
    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.
             O raio r em metros.
     Saída:  Um booleano (True se o lançamento teve sucesso e acertou o
             pokemon, ou False caso contrário) e as coordenadas finais
             x e y da pokébola.
    '''
```

Neste EP2, você deverá atender às solicitações dos avaliadores da FAPPESP e adaptar seu protótipo PALC-9000.

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

```
GRAVIDADE = 9.81
EPSILON = 0.01
DELTA_T = 0.01
PI = 3.14159265358979323846
```

As seguintes funções e operações podem ser usadas:

`abs`
`**`

Funcionamento do programa

O programa deve ler inicialmente as coordenadas `x_pokemon` e `y_pokemon` do pokémon, bem como seu raio `r`. O treinador tem, como antes, no máximo três chances para acertá-lo. 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 uma das condições ser atingida:

1. A pokébola bateu no chão (posicao `y_pokebola` menor do que EPSILON).
2. A pokébola atingiu o pokémon.

No final de cada tentativa, seu programa deve imprimir uma mensagem indicando se o pokémon foi atingido e caso não tenha sido, a distância do ponto em que a bola caiu até a borda do pokémon.

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)

Pidgey no chão:

```
Digite a coordenada x do pokemon:  10
Digite a coordenada y do pokemon:  0
Digite o raio do pokemon (> 0) em metros:  0.3

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 atingiu o pokemon por 2.0675 metros.

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 atingiu o pokemon por 1.4269 metros.

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 atingiu o pokemon.
```

Pidgey voando:

```
Digite a coordenada x do pokemon:  10
Digite a coordenada y do pokemon:  5
Digite o raio do pokemon (> 0) em metros:  0.3

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 atingiu o pokemon por 6.5487 metros.

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 atingiu o pokemon.
```