# EP1 - CCMI - 2007 - v. 1.0
# Paredes

Paulo J. S. Silva

18 de setembro de 2007

## 1 Descrição

Você deve implementar um programa de simulação da trajetória de uma partícula em uma sala quadrada de $1 \times 1$ metro. Veja a figura 2.2. Esta sala possui uma minúscula porta localizada no canto inferior direito de 20cm.

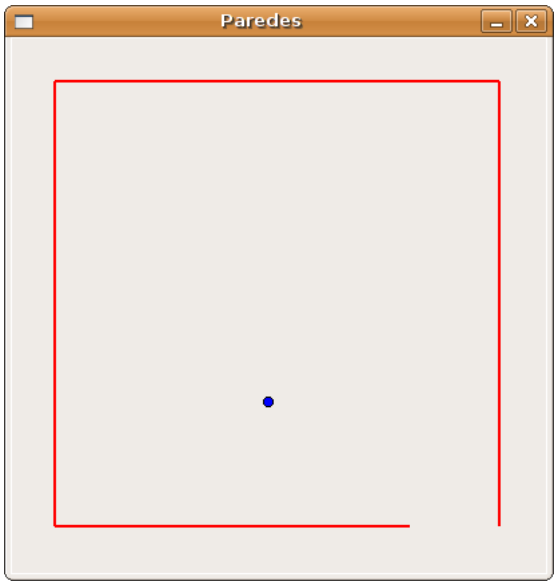


Figura 1: Uma sala quadra de $1 \times 1$ metro com a porta no canto inferior direito.

O programa deve ler um arquivo de entrada chamado entrada.txt que possui em cada uma das linhas as seguintes informações (em ordem):

```
posição x inicial da partícula
```

```
posição y inicial da partícula
Componente x da velocidade da partícula, em metros por segundos
Componente y da velocidade da partícula,  em metros por segundos
Intervalo máximo entre duas cenas, em segundos
```

Considera-se que o eixo de coordenadas tem origem no canto inferior esquerdo.

Seu programa deve então simular a trajetória da partícula por até 200 segundos, verificando se ela escapa ou não da sala. A simulação deve ser representada por uma animação que apresente pelo menos um quadro a cada intervalo fornecido na última linha do arquivo de entrada.

Sua simulação deve considerar a partícula como um ponto material dentro da sala (pense em um fóton). Ao encontrar uma das paredes a partícula deve se refletir de modo que o ângulo de incidência seja igual ao de reflexão (de novo, imagine um fóton em uma sala com espelhos nas paredes). O módulo da velocidade não deve se alterar. Note que se o intervalo entre os quadros for muito grande (ou se a velocidade for muito alta) é possível que a partícula quique em mais de uma parede dentro do intervalo. O seu programa deve levar isso em consideração.

A simulação deve acabar em dois casos:

1. Terminou o período de 200 segundos e a partícula não saiu da sala. Nesse caso o programa deve imprimir o instante e a posição final da partícula. Você pode assumir que o intervalo entre os quadros é tal que 200 segundos é um múltiplo inteiro deste.

2. A partícula saiu da sala antes de terminar o prazo. O programa deve então parar e informar o instante em que a partícula deixa a sala e sua posição.

Após apresentar a mensagem final, o programa deve esperar que o usuário digite `enter` para fechar a janela da animação.

# 2 Exemplos de execução

## 2.1 Partícula sai da sala

Arquivo de entrada:

```
0.5
0.5
0.057
0.023
0.1
```

Imagens da tela ao final:

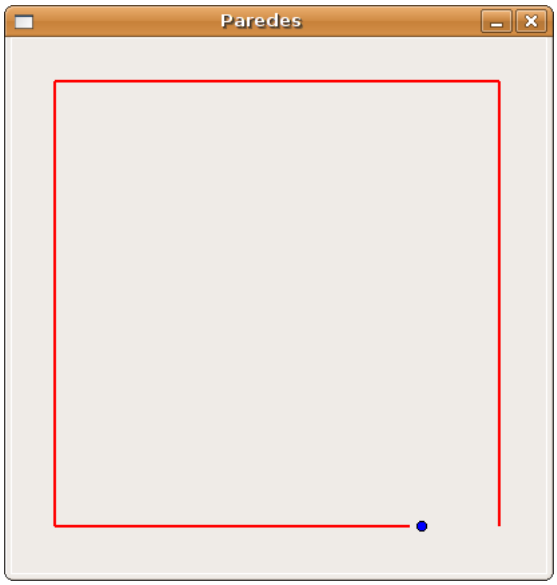


Figura 2: Imagem ao final de uma imagem em que a partícula escapa sala.

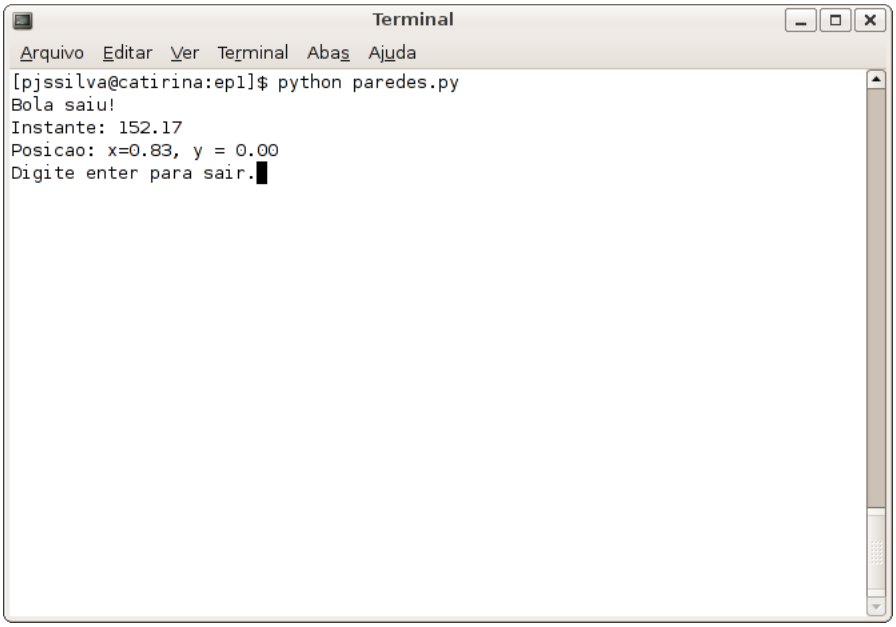


Figura 3: Mensagens no terminal quando a partícula deixa a sala.

## 2.2 Partícula fica da sala

Arquivo de entrada:

```
0.5
0.5
0.17
0.017
1.0
```

Imagens da tela ao final:

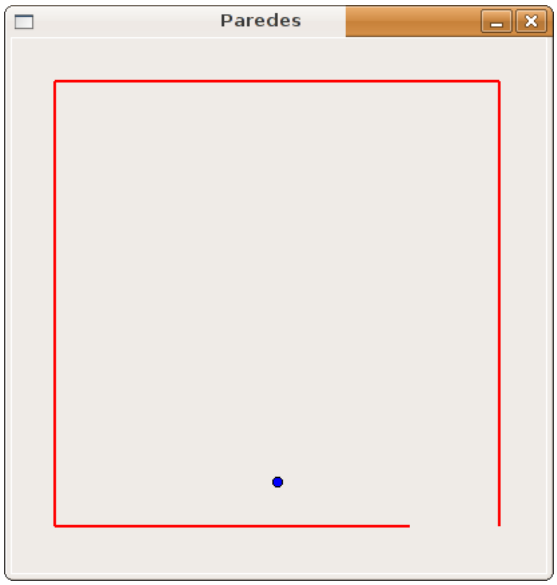


Figura 4: Imagem ao final de uma imagem em que a partícula fica na sala.

# 3 Detalhes de número em ponto flutuante

Vocês deve tomar cuidado com um detalhe importante de implementações que usam números reais (de ponto flutuante). Como já comentei em sala, as contas e a representação dos números nesse sistema não é exata, incorrendo sempre em pequenos erros. Nesse em caso, não faz muito sentido comparar dois números que são resultados de contas para verem se são iguais.

Assim, código que envolvem números em ponto flutuante geralmente comparam igualdades usando faixas, ou seja, define-se um $epsilon > 0$ e para verificar se $x = y$ usa-se um código do tipo

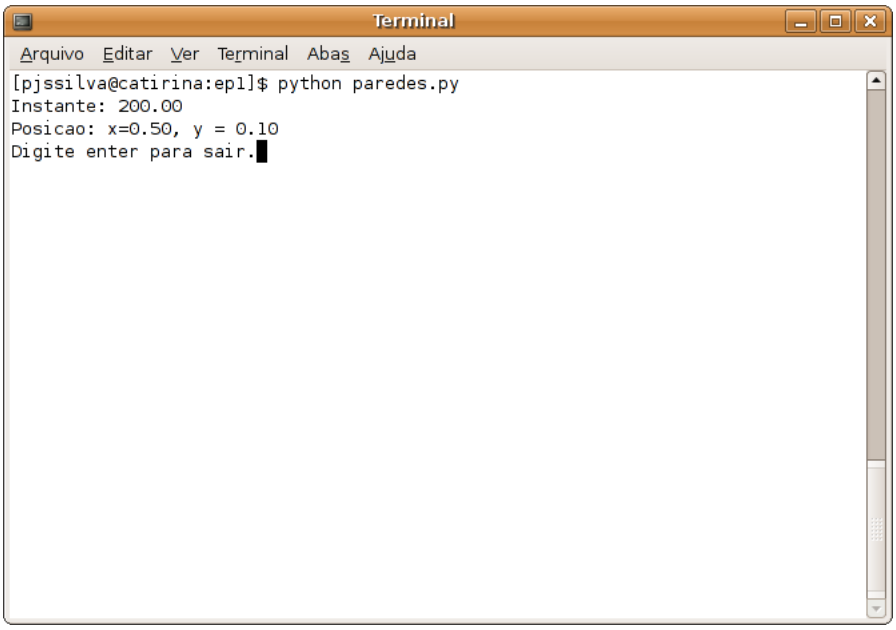


Figura 5: Mensagens no terminal quando a partícula fica na sala.

```
if math.fabs(x - y) < epsilon:
    print 'iguais'
else:
    print 'diferentes'
```

Nesse EP, esse tipo de salvaguarda, deve ser observada em pelo menos dois pontos: ao se verificar se a partícula bateu em uma parede e ao se testar se a partícula passou pela porta. Por exemplo, para verificar se a partícula bateu na parede da esquerda você não deve fazer uma comparação do tipo:

```
if x == 0.0:
    print 'bateu'
else:
    print 'nao bateu'
```

Mas sim, fazer algo do tipo:

```
if x < epsilon:
    print 'bateu'
else:
    print 'nao bateu'
```

# 4 Considerações finais

Você deve organizar bem o seu programa, inclusive usando (várias) funções. A melhor idéia é seguir o ciclo de desenvolvimento de um programa, escrevendo primeiro uma especificação do que ele deve fazer e fazendo refinamentos até se chegar a uma solução completa. Isso deve ajudá-lo a encontrar uma organização razoável do código.

Além disso, se você sentir necessidade, comente o seu código. Para novatos, como vocês, é melhor pecar pelo excesso de comentários do que pela falta destes. No *mínimo* comente cada uma das funções da forma comentada em classe.

Ainda, o seu arquivo deve conter um cabeçalho contendo o nome do arquivo, o nome do autor e seu número USP e uma breve descrição dizendo o que o programa faz.

Por fim, lembrem de usar o fórum de discussão na página da disciplina. Isso é fundamental. No fórum posso disponibilizar correções no enunciado ou esclarecer dúvidas e definir detalhes que passaram despercebidos.

Não custa lembrar que o EP é um trabalho *individual*. Você pode discutir com os colegas o programa mas nunca trocar trechos de código. Sugiro nem mesmo olhar código do colega, apenas conversar. Caso encontremos trabalhos com trechos iguais os dois alunos receberão 0 no EP e a média final de EPs ainda receberá um desconto extra de 10%.