# 7-Construção de Gráficos em Tempo Real

April 20, 2021

## 1 Informes iniciais

1. A interface do VPython, juntamente com as suas figuras, aparece na mesma célula de importação do vpython pela primeira vez após o kernel ser reiniciado. Então, após executar a célula para se gerar as figuras, **o usuário irá vizualizá-las na célula de importação da biblioteca**;
2. Caso o usuário queira "limpar" as figuras na interface, é preciso reiniciar o kernel e as saídas, indo no menu em `Kernel -> Restart & Clear Output`, se estiver no Jupyter Notebook;
3. É preciso lembrar de **sempre** executar a célula para importar a biblioteca VPython após reiniciar o kernel, uma vez que ela é a responsável por proporcionar a criação das figuras e afins.

## 2 Gráficos em Tempo Real

Gráficos são capazes de auxiliar os estudantes na compreensão de aspectos específicos de determinados fenômenos. No caso da construção de gráficos em tempo real, a persepção da modelagem do fenôomeno se torna ainda mais dinâmica, pois além de permitir uma maior interatividade a partir dos dados inseridos pelos usuários, é possível observar a sua construção de forma atenta.

## 3 Construção de Gráficos em Tempo Real

Para a construção do gráfico, utilizamos dois tipos de variáveis: `graph` para representar a interface do gráfico e `gcurve` para gerar a curva do gráfico. Na construção da interface do gráfico, é necessário apenas o parâmetro que define a legenda do eixo x (`xtitle = 'legenda x'`) e do eixo y (`ytitle='legenda y'`). No que se refere aos parâmetros iniciais da curva, ela é deifinida por sua cor(`color`) e legenda(`label`).

*Essas variáveis padronizarão a interface do gráfico, ele será plotado apenas a partir dos dados posteriormente*

[ ]:
```
from vpython import * #Lembrar SEMPRE de importar a biblioteca do VPython

graf_pos = graph(xtitle='tempo (s)', ytitle='Posição (m)')
curva_pos = gcurve(color=color.red, label = 'componente X')
```

Como exemplo, utilizaremos a construção da posição *versus* o tempo no MRU (eq. 1). Assim, solicitaremos os dados para o usuário e, a parir deles, desenvolveremos o gráfico em tempo real.

$x = x_0 + v\Delta t$ (1)

[ ]:
```
x_inicial = float(input('Insira a Posição Inicial(m): '))
v = float(input('Insira a Velocidade da Partícula(m/s): '))
t_inicial = float(input('Insira o Tempo Inicial(s): '))
t_final = float(input('Insira o Tempo Final(s): '))
```

Para a plotagem do gráfico, utilizamos o parâmetro `.plot` na variável da curva, definindo o padão de mudança de posição em função do tempo `pos=(eixo x (tempo), eixo y (posição))`.

**(Lembre-se que a interface é gerada na célula de importação do VPython - ver Informes Iniciais)**

[ ]:
```
delt_t = t_final - t_inicial #tempo de execução da simulação(variação do tempo)
t = 0 #passagem do tempo
while (t < delt_t):
    rate (100)
    t+=0.01
    curva_pos.plot(pos=(t_inicial + t, x_inicial + v*t)) #plotagem do gráfico
```

Para mais informações sobre a plotagem de gráficos, ver: https://www.glowscript.org/docs/VPythonDocs/graph.html

*Reinicie o kernel e limpe as saídas (ver Informes Iniciais) e comece a execução dos códigos a partir da célula abaixo para uma melhor facilidade na vizualização*

## 4 Construção de Gráficos em Tempo Real em Conjunto com a Simulação

Para a construção de gráficos em tempo real em conjunto com a simulação, basta associarmos o código de desenvolvimento de simulações com o código de construção de gráficos. Nesse caso, utilizaremos o código final do tutorial anterior (6-Desenvolvimento de Simulações com o VPython) e desse tutorial.

[ ]:
```
from vpython import * #sempre importar as bibliotecas ao início

graf_pos = graph(xtitle='tempo (s)', ytitle='Posição (m)')
curva_pos = gcurve(color=color.red, label = 'componente X')

x_inicial = float(input('Insira a Posição Inicial(m): '))
v = float(input('Insira a Velocidade da Partícula(m/s): '))
t_inicial = float(input('Insira o Tempo Inicial(s): '))
t_final = float(input('Insira o Tempo Final(s): '))

if (v<0):
    p = 10
else:
    p=-10

#Perceba que vamos inserir os parâmetros da trajetória prórpia esfera
```

```
esfera = sphere(pos = vec(p, 0, 0), radius = 0.2, color = color.red, opacity=1,␣
 ↪make_trail=True, interval=10, trail_type="points", trail_color =color.yellow)
texto_x_inicial = label(pos=vec(p, 0, 0), text= str(x_inicial) + " m",␣
 ↪xoffset=0, yoffset=-100, space=30, height=16, border=4)
delt_t = t_final - t_inicial #tempo de execução da simulação (variação do tempo)
t = 0

e = (abs(v*delt_t))/20 #fator de controle de escala

while (t < delt_t):
    rate (100)
    t+=0.01
    esfera.pos = vec(p + (v*t)/e, 0, 0)#Perceba a inserção do fator de escala␣
 ↪apenas na variação da posição
    curva_pos.plot(pos=(t_inicial + t, x_inicial + v*t)) #plotagem do gráfico
```

[ ]: