ATC – Aula 03 – Parte 1: Introdução à modelagem computacional

Objetivos: identificar as vantagens do uso de modelagem/simulação numérica nesta disciplina; reconhecer sua complementaridade com o cálculo diferencial e integral e sua importância em quase todas as áreas de pesquisa teóricas e experimentais; adquirir noções básicas do funcionamento de um ambiente de desenvolvimento de programas (Geany ou VIDLE) e da programação em VPython (a linguagem Python com um módulo de visualização 3D).

Metas: nesta atividade em particular vocês vão:

- aprender a usar o Geany para editar programas em VPython;
- descobrir que programas são feitos de um conjunto de linhas com instruções, que são executadas em sequência, e precisam der digitadas de forma precisa;
- criar objetos gráficos simples como esferas e setas;
- mudar as características desses objetos (posição, tamanho, cor, orientação);
- criar setas para representar vetores com as características desejadas (escala correta, orientação);
- criar e usar uma nomenclatura simbólica para objetos e quantidades.

Introdução

Um programa de computador consiste de uma sequência de instruções. O computador executa essas instruções uma a uma, na ordem em que aparecem, e para quando chega ao final delas. Cada instrução deve ser digitada de forma exata (como se fossem instruções ou operações em uma calculadora). Se o computador encontrar um erro em uma instrução (como um erro de digitação), ele vai parar e escrever uma mensagem de erro.

Um programa simples típico tem quatro seções:

- comandos de definições gerais;
- definição de constantes;
- criação de objetos;
- cálculos para localizar, desenhar ou mover objetos; esses cálculos podem se repetir.

Nesta atividade, vamos nos concentrar nas três primeiras partes. Os membros da equipe devem se revezar nas funções, de acordo com as instruções.

- **Gerente:** lê as instruções em voz alta; controla o tempo e evita dispersão da equipe.
- **Relator:** digita os comandos no computador.
- **Crítico:** anota as respostas das perguntas do texto, garantindo que todos os membros da equipe as compreendam.

Parte A: aluno 1 é o gerente, aluno 2 é o relator e aluno 3 é o crítico.

- abram no moodle a "Atividade em classe - aula 03 (parte 0)" e digitem os nomes dos membros da equipe presentes na aula, seguidos do seu número (fulano é aluno 1 ou está ausente, por ex.). Certifiquem-se que o sistema operacional do computador é o Linux.

1 – Usando o Geany

Procurem pelo ícone do editor e abram-no. Digitem as três linhas de comando abaixo na tela do editor:

```
# -*- coding: cp1252 -*-
from __future__ import division
from visual import *
```

Todo programa em VPython deve começar com essas três linhas. A primeira delas diz ao computador para aceitar sinais de acentuação nos comentários. A segunda (from <espaço> <underscore> <underscore> future <underscore> <underscore> <espaço> import division) diz à linguagem VPython para tratar ½ como 0,5. Sem isso a linguagem assume que as divisões são entre números inteiros com resultados inteiros (truncados) e ½ será igual a zero! A terceira linha diz ao programa para usar o módulo 3D (chamado "visual"). Notem ainda que, como no caso da maioria das linguagens de programação, o VPython se baseia na língua inglesa. Decimais em um número devem ser indicados por um ponto e as componentes de um vetor devem ser separadas por uma vírgula. Apenas neste texto, por simplicidade, vamos manter essa nomenclatura para vetores: <3.1, 4, -2> em vez de <3,1; 4; -2>.

Antes de continuar, vamos salvar o programa:

- Vão até “arquivo”, “guardar como”, selecionem um local e dêem ao programa um nome (vetor-XXXXX.py), em que XXXXX é o número da sua turma e equipe, por exemplo T308B. **Todo programa em VPython deve terminar com a extensão “.py”.** CADA VEZ QUE VOCÊ EXECUTAR UM PROGRAMA, O GEANY AUTOMATICAMENTE O SALVARÁ ANTES. Dessa forma, se for introduzir modificações, mas não quiser perder a versão original, lembre-se de salvá-lo com outro nome antes de executá-lo.

2. Criando um objeto
Nesta seção vocês vão aprender a criar um objeto 3D. Instruam o VPython a criar uma esfera. Na linha seguinte do seu programa no Geany, no início da linha, digitem:

```
sphere()
```

Esse comando diz ao computador para criar uma esfera. Executem (rodem) o programa pressionando a tecla F5 (ou através do menu “construir” → “executar”). Duas janelas devem aparecer, além daquela na qual vocês estão editando. Uma delas é uma janela gráfica, que agora contém uma esfera.

3. A janela gráfica 3D
Por padrão (default) a esfera tem sua origem no centro da janela e a “câmera” (isto é, o seu ponto de vista) está voltado diretamente para o centro.

- Apertem e segurem ambos os botões e movam o mouse para cima e para baixo de forma a aproximar e afastar a câmera do centro da cena (pode haver pequenas diferenças de sistema operacional para sistema operacional).
- Pressionem e segurem o botão direito do mouse enquanto o movimentam, para fazer a câmera “girar”, mudando a direção a partir da qual se está olhando para o centro (pode haver pequenas diferenças de sistema operacional para sistema operacional).

Quando vocês rodam o programa pela primeira vez, o sistema de coordenadas cartesiano tem o eixo *x* apontando para a direita, o eixo *y* apontando para cima e o eixo *z* saindo da tela, em sua direção. Vocês podem rodar a câmera, fazendo com que os eixos apontem em outras direções.

4. A janela de texto do Python (Python shell window)
A segunda janela que se abre é a janela de texto do Python. Ela é importante, pois nela aparecem as mensagens de erro. Também é nessa janela que resultados de cálculos ou outras informações serão impressas se linhas de código com essas instruções forem colocadas no programa. **Importante**: arrumem as janelas na tela de modo que a janela de texto fique sempre visível. NÃO FECHEM ESSA JANELA DURANTE A EXECUÇÃO DO SEU PROGRAMA.

- Usem o mouse para diminuir essa janela, movendo-a para fora do centro da tela (para baixo, por exemplo) mas deixando-a aberta quando estiverem escrevendo e executando programas, de modo a ver as mensagens de erro.
- Não expandam a janela de edição do Geany de modo que tome toda a tela. Vocês vão perder informações importantes se o fizerem.
- Para encerrar um programa, fechem a janela de texto do Python.

5. Mensagens de erro
Para ver um exemplo de mensagem de erro, façam um erro simples de digitação.

- Mudem a terceira linha inicial para: `from bisual import *`
- Rodem o programa. Vejam que aparece uma mensagem de erro na tela de texto. Essa mensagem informa o nome do arquivo, a linha na qual o erro ocorreu e uma descrição do erro (no caso “ImportError: No module named bisual” = Erro de importação: nenhum módulo com nome bisual).

Leiam as mensagens de erro de baixo para cima, buscando a informação sobre a linha (‘line’) em que se localiza o erro.

- Corrijam o erro.

Sempre procurem as mensagens na janela de texto se um programa não rodar. Mesmo que vocês não entendam a

mensagem de erro, é importante que a vejam, de modo que possam localizar onde, no código, o erro ocorreu.

Parte B: aluno 2 é o gerente, aluno 3 é o relator e aluno 1 é o crítico.

6. Mudando "atributos" (posição, cor, forma, tamanho, etc.) de um objeto.
Vamos agora colocar a esfera em outro lugar, com outro tamanho.

- Alterem a última linha digitada, da seguinte forma:

  ```
  sphere(pos=vector(-5,2,-3), radius=0.40, color=color.red)
  ```

- Executem o programa. Experimentem outras posições, cores e raios, executando-o após cada modificação. Discutam e anotem as respostas para as perguntas abaixo (não é necessário entregar respostas).

PERGUNTAS:

(a) O que ocorre se mudamos o atributo "pos" da esfera?
(b) O que ocorre quando alteramos o atributo "radius"?
(c) Que cores vocês pode usar? Como alterá-las? Listem algumas que funcionaram.

7. Autoescala e unidades
O VPython automaticamente "dá um zoom" de modo que todos os objetos criados apareçam na tela. Por conta desse ajuste automático de escala, podemos usar, nos números informados nos atributos "pos" e "radius", qualquer sistema consistente de unidades, como metros, centímetros, etc. Por exemplo, o raio 0.20 pode representar 0,2 m e o centro da esfera estará em <2,4,0> m. Ou representar 0.2 polegadas (e o centro estará na posição <2,4,0> polegadas).

8. Criando um segundo objeto

- Em uma nova linha, após o comando que criou a esfera vermelha, criem uma esfera verde com centro em <-3, -1, 0> e raio 0.15.
- Executem o programa. Vocês devem ver ambas as esferas.

9. Criando uma seta
Com muita frequência, vamos utilizar setas para representar grandezas vetoriais. Vamos aprender como criá-las e quais são seus principais atributos.

- Digitem em uma nova linha do programa:
  ```
  arrow(pos=vector(2,-3,0), axis=vector(3,4,0), color=color.cyan)
  ```
- Executem o programa.
- Experimentem mudar os atributos "pos", "axis" e "color", executando o programa após as alterações. Mudem uma coisa de cada vez, de forma a perceber claramente o efeito de cada alteração.
- Tentem colocar uma segunda seta na mesma "pos", mas com um diferente "axis" (e cor).
- Encontrem as respostas para as perguntas abaixo e anotem as respostas (não é necessário entregá-las).

PERGUNTAS:

(d) O atributo "pos" de uma seta indica a posição do início, da ponta ou do meio da seta?
(e) O atributo "axis" indica a posição do início da seta, da ponta da seta ou alguma outra coisa, como a posição da ponta em relação ao início?
(f) Para criar uma esfera cujo centro esteja na ponta da seta desenhada anteriormente, qual deveria ser o atributo "pos" da esfera?

10. Multiplicando por um escalar
Como o atributo "axis" é um vetor, podemos multiplicá-lo por uma grandeza escalar.

- Comecem com uma seta definida por
  ```
  arrow(pos=vector(2,-3,0), axis=vector(3,4,0), color=color.cyan)
  ```
- Modifiquem o atributo "axis", mudando o comando anterior para o abaixo, e observem o que ocorre:
  ```
  arrow(pos=vector(2,-3,0), axis= –0.5*vector(3,4,0), color=color.cyan)
  ```

PERGUNTAS:

(g) Para criar uma seta 3 vezes mais longa, sem mudar a sua direção ou seu sentido, por que fator devemos multiplicar o atributo “axis”?
(h) Para inverter o sentido de uma seta, sem mudar a sua direção e sem mudar o seu comprimento, devemos multiplicar o atributo “axis” por qual número?

11. Linhas de comentários
As linhas de comentários são ignoradas na execução de um programa. Uma linha de comentário começa com o símbolo #. Podemos escrever nelas qualquer coisa. Em geral um comentário é uma observação para você mesmo, ou para outro usuário do programa. Eles permitem acompanhar o que o programa está fazendo. Por exemplo:
objetos criados nas linhas seguintes

Um comentário também pode ser usado para suprimir temporariamente uma linha de código, sem apagá-la.

- Ponham o símbolo # no início da linha que cria a seta, rodem o programa e vejam o que ocorre.
  #arrow(pos=vector(2,-3,0), axis=vector(3,4,0), color=color.cyan)

Fiquem atentos: só é possível colocar acentos usuais na língua portuguesa nos comentários se a primeira linha de código corresponder a -- coding: cp1252 -*-. Caso contrário, eles vão gerar mensagens de erro.*

Parte C: aluno 3 é o gerente, aluno 1 é o relator e aluno 2 é o crítico.

12. Representando um vetor posição por meio de setas.

- Limpem seu programa de modo que contenha apenas os seguintes objetos:
  - uma esfera vermelha (representando uma bola de basquete) com centro na posição < - 5, 2, - 3> e raio 0.4;
  - uma esfera verde (representando uma bola de tênis) com centro em < -3, -1, 0> e raio 0.15;
  - uma seta em ciano, com início em <2, -3, 0> e ponta em <2, -3, 0> + <3, 4, 0>.

Podemos usar setas para indicar vetores (como o vetor posição). Lembrem-se que um vetor que se inicia na posição $\vec{A}$ e termina na posição $\vec{B}$ pode ser encontrado por meio da operação subtração (vetorial) “final menos inicial”, ou seja, $\vec{B} - \vec{A}$. Queremos criar uma seta representando um vetor que vá da bola de basquete até a bola de tênis. Isso significa que o início da seta deve estar na posição da bola de basquete (esfera vermelha) e a sua ponta na posição da bola de tênis (esfera verde).

PERGUNTAS:
(i) Qual deve ser o atributo “pos” dessa seta?
(j) Qual deve ser o atributo “axis” dessa seta?

- Usando esses valores para os atributos “pos” e “axis”, alterem a última linha do programa de modo que a seta ciano aponte da esfera vermelha para a verde.
- Executem o programa.

Ponto de Verificação: examinem a janela 3D com bastante atenção. Se alguma coisa estiver errada, corrijam o seu programa. **Se necessário, chamem um instrutor.**

13. Nomeando objetos; usando os nomes e atributos de um objeto

- Mudem a posição da esfera verde para < -3, -1, 3.5>, e não alterem mais nada.
- Executem o programa.

Notem que a seta ainda aponta na direção anterior. Como fazer para que essa seta **sempre aponte para a bola de tênis, esteja ela onde estiver**? Para conseguir isso teremos que nos referir à posição da bola de tênis de forma simbólica. Mas, antes disso, precisamos dar nomes a cada uma das esferas, para distingui-las.

- Nomeiem as esferas, alterando as linhas de comando que as definem para:
  basquete = sphere(pos=vector(-5, 2, -3), radius=0.40, color=color.red)
  tenis = sphere(pos=vector(-3,-1,3.5), radius=0.15, color=color.green)

Dessa forma atribuímos nomes às esferas. Podemos nos referir a elas, em outro ponto do programa, por meio desses nomes. Mais ainda, podemos nos referir a determinado atributo dessas esferas por meio desse nome seguido de um

ponto e do atributo. Por exemplo, escrevendo tenis.pos para o atributo "pos" da esfera denominada tenis ou basquete.color para o atributo "color" da esfera denominada basquete. Para ver como isso funciona, façam o exercício abaixo.

- Em uma nova linha do programa escrevam
  print tenis.pos
- Executem o programa.
- Vejam o que apareceu na janela de texto: o vetor impresso deve ser o vetor que indica a posição do centro da esfera verde.

PERGUNTA:

(k) o que vocês esperam que aconteça se digitarem a seguinte linha de comando?
print tenis.pos – basquete.pos

- Digitem a linha, executem o programa e vejam se estavam certos.
- Vamos agora dar também um nome para a seta: editem a linha de comando que define a seta ciano para
  bt = arrow(pos= ....) #basquete para tenis

Como podemos nos referir aos atributos de um objeto de forma simbólica, vamos escrever expressões simbólicas para os atributos "axis" e "pos" da seta "bt". As expressões devem usar os nomes genéricos dos atributos na forma simbólica, como "tenis.pos" ou "beisebal.pos", não vetores numéricos. Dessa forma, cada vez que um desses atributos mudar, os atributos da seta bt também vão mudar. Mas a seta estará sempre apontando da bola de basquete para a bola de tênis!

PERGUNTA:

(l) Simbolicamente, qual deveria ser o atributo "pos" da seta bt para que sua origem esteja sempre na bola de basquete? **Sua expressão não deve conter números!**

(m) Simbolicamente, qual deveria ser o atributo "axis" da seta bt para que sua ponta esteja sempre na bola de tênis (lembre-se de como definimos a posição relativa de dois objetos)? **Sua expressão não deve conter números!**

- Mudem a última linha do programa de forma a definir os atributos "pos" e "axis" da seta bt de forma simbólica.
- Executem o programa. Girem a câmera, aproximem e afastem o centro, de forma a ter certeza de que o código está correto. Se não, corrijam o programa, mas sem usar números.
- Agora, mudem a posição da bola de basquete para (-4, -2, 5) e da bola de tênis para (3, 1, -2). Executem o programa e vejam o que acontece.

Ponto de verificação: o seu programa se comporta como o esperado, isso é, não importa qual a posição do centro das esferas, a seta sempre aponta da esfera vermelha para a verde? Corrijam se necessário, peçam ajuda ou comparem seu código com o de seus colegas.

Ao terminarem a atividade, **abram no moodle a "Atividade em classe – aula 03 (parte 1)" e façam o upload do programa da sua equipe.** Se não houverem acabado a atividade quando o tempo dado por seu professor se esgotar, façam o "upload" do programa com as alterações feitas até o momento. Vocês terão a nota mesmo que não tenham terminado a atividade.

14. Guardem uma cópia de seu programa, para referências futuras. Vocês podem enviá-lo por e-mail, por exemplo.

15. Vocês podem instalar o VPython e o Geany ou o VIDLE no seu computador pessoal, para treinar ou fazer outros exercícios, ou ainda executar os programas diretamente na Internet. Vejam como no texto de leitura disponível no moodle.

16. Vocês também podem rodar programas em VPython em um computador que não tem o VPython ou o Geany instalados, desde que este esteja ligado na Internet. Basta ir ao site www.glowscript.org. Infelizmente o site é em inglês, mas não é necessário muito conhecimento dessa língua para entender o que fazer. Se tiverem dificuldades com o idioma, procurem um instrutor ou um colega.