

# Exercício V – Parte I (2012/I)

Disciplina: Computação Gráfica

Professora: Soraia R. Musse

## Aula prática em POV-Ray – Renderização de Imagens e Iluminação

O **objetivo desta aula** é trabalhar com o programa POV-Ray e suas funcionalidades básicas de renderização de imagens.

Inicialmente, abra o programa POV-Ray instalado em seu computador. Se não estiver instalado, o executável se encontra na página da disciplina. A instalação é rápida e fácil.

Ao abrir o programa, com o arquivo biscuit.pov ou woodbox.pov abertos, clique no botão Run e verifique a imagem renderizada pelo programa. Antes de tentar entender o que está acontecendo, vamos criar uma cena simples do zero para aprendermos as funcionalidades básicas do programa.

Antes de qualquer coisa, vamos entender o que este programa é exatamente. É um software que aplica a técnica de renderização Ray-Tracing, vista em aula, que simula o comportamento dos raios de luz em um ambiente. O programa fornece objetos e texturas para serem utilizados e comandos para controlar a iluminação, posição e tamanho dos objetos dentre muitas outras funcionalidades para facilitar a criação de uma cena 3D. Trabalha com o sistema de coordenadas x, y, z que já estamos familiarizados.

Para criar uma nova imagem, primeiramente aperte o botão “New” na barra de ferramentas para iniciar um novo arquivo. Vamos começar incluindo as bibliotecas que serão utilizadas pela imagem:

```
#include "colors.inc"  
#include "stones.inc"  
#include "textures.inc"  
#include "glass.inc"
```

Por enquanto estas duas bibliotecas são o suficiente. Elas incluem elementos de cena pré-determinados, como cores e texturas de pedras, vidros e outras em geral. Clicando com o botão direito em cima de uma dessas linhas de comando no POV-Ray, existe a opção de abrir o arquivo da linha selecionada, o que permite verificar quais texturas e cores existem em cada um. Futuramente, se quiseres tentar alguma outra textura, existem mais opções de include, como:

```
#include "metals.inc"  
#include "woods.inc"
```

É importante acrescentar que só se deve incluir bibliotecas que serão de fato utilizadas, para economizar trabalho para o processador no momento de executar a renderização da imagem.

Continuando, vamos definir a cor do plano de fundo e os parâmetros da câmera, adicionando o código:

```
background { color Gray }
```

```
camera {  
    location <0, 2, -3>  
    look_at <0, 1, 2>  
}
```

Como podemos ver, para definir o plano de fundo estamos utilizando uma cor pré-definida, porém se a cor que se deseja não existe pronta, pode-se utilizar o formato (exemplo de um tom de rosa):

```
color red 1.0 green 0.8 blue 0.8
```

ou

```
color rgb <1.0, 0.8, 0.8>
```

Quanto aos parâmetros da câmera, o comando **location** define a posição da câmera em relação ao centro do universo, situado em <0,0,0>. Já o comando **look at** define a posição do centro da atenção da câmera (para onde ela vai estar direcionada). Ou seja, aplica uma rotação na mesma para esta apontar para o ponto posicionado nas coordenadas informadas. Experimente alterar os valores para ver o que é modificado.

Agora vamos adicionar uma esfera:

```
sphere {  
    <0, 1.5, -1>, 0.2  
    texture { pigment { color Yellow } }  
}
```

O primeiro vetor especifica a posição da esfera, seguido por um 2 que define o raio da mesma. A textura utilizada por enquanto é apenas um pigmento da cor amarela. Execute o que foi feito até agora, e verifique o que acontece. Não podemos ver as propriedades 3D da esfera, pois ela não está iluminada.

Adicione a linha a seguir para definir uma fonte de luz:

```
light_source { <1, 3.5, -3> color White }
```

O vetor define a localização da luz em relação à origem. A fonte da luz é um ponto invisível que emite luz e não tem nenhum formato ou textura. Execute novamente para ver os resultados.

Agora vamos adicionar outros objetos para incrementar a cena:

Caixa:

```
box {
  <-1, 0.7, -1>, // Near lower left corner
  <1, 1.2, 3> // Far upper right corner
  texture {
    T_Stone20 // Pre-defined from stones.inc
    scale 2 // Scale by the same amount in all
            // directions
  }
  rotate y*30 // Equivalent to "rotate <0,20,0>"
  rotate x*3 // Equivalent to "rotate <0,20,0>"
}
```

Como se pode perceber a caixa é definida pela especificação das coordenadas 3D de seus dois cantos opostos. O primeiro vetor define a posição do vértice do canto inferior esquerdo, perto da câmera, e outro define a posição do canto superior direito, longe da câmera. A textura que é utilizada é a T\_Stone20 e a escala modifica o quão espaçosos são os ruídos existentes na textura. Brinque um pouco com esses parâmetros mude para T\_Stone1, T\_Stone2, T\_Stone15, e mude a escala para ver as alterações que ocorrem.

O rotate aplica uma rotação no objeto, equivalente às rotações que fazemos no OpenGL.

Agora tente comentar a linha **texture { pigment { color Yellow } }** da esfera, e adicione no lugar a linha **texture { T\_Glass4 } interior {I\_Glass }**. Verás que a esfera ficará com uma aparência de vidro.

Agora vamos adicionar mais um objeto e o tão famoso chão em xadrez:

Cilindro:

```
cylinder {
  <-1, 1.5, 0>, // Center of one end
  <0, 1.5, 1>, // Center of other end
  0.2 // Radius
  open // Remove end caps
  texture { T_Stone25 scale 4 }
  translate x*1.5 // Equivalent to "translate <1.5,0,0>"
}
```

O cilindro funciona basicamente definindo o centro de cada ponta e o raio. O comando **open** define que o cilindro será aberto e oco. Experimente retirar essa linha e ver o resultado.

Por último vamos adicionar um chão xadrez na cena:

```
plane { <0, 1, 0>, -1
  pigment {
    checker color Black, color White
  }
}
```

O vetor define a normal da superfície do plano, no caso uma reta positiva em y. O -1 define a distância que o plano é deslocado ao longo da normal a partir da origem.

Vimos algumas formas e funcionalidades básicas do POV-Ray. Sinta-se à vontade para testar outras texturas e posições para os objetos. A documentação do POV-Ray contém um tutorial ensinando a fazer tarefas mais complicadas, demonstra mais texturas, objetos e outras funcionalidades que o programa disponibiliza. Ela está disponível para download no site: <http://www.povray.org/download/>