MC937A/MO603A – Computação Gráfica - 2020-S2 - Jorge Stolfi
Trabalho de laboratório 02 - 2020-10-14
Distanciamento computacional

Objetivos: treinar uso de comandos da linguagem PO-Ray: repetição, testes condicionais, e procedimentos ("macros").

Enunciado. Um cientista de saúde pública, que prefere se manter no anonimato, afirma que o vírus do COVID-19 só pode propagar dentro de um raio de 1627 milímetros do nariz de uma pessoa infectada. Sua tarefa nesta aula é produzir uma imagem de um auditório com assentos marcados de forma a garantir esta distância mínima.

Parte 1. Antes de começar a programar, **desenhe um esboço do seu auditório e apresente-o ao professor via Meet, quando solicitado, no início da aula.**

Parte 2. Produza uma imagem da sua cena usando POV-ray. O modelo deve incluir: (1) Uma macro do POV-Ray `assento` que gera um assento numa posição fixa, com um parâmetro booleano que indica se ele deve estar "bloqueado" ou "livre". (2) Uma macro `auditorio` que gera um arranjo regular de assentos com m fileiras de n assentos cada, sendo m e n parâmetros da macro. (3) Uma cena que tem pelo menos duas chamadas da macro `auditorio`.

Arquivos. Copie os arquivos da aula passada para uma nova sub-pasta `2020-10-14` da pasta `mc937` no seu computador. Edite o arquivo `main.pov`, conforme solicitado acima. Execute o comando `make` numa shell para gerar a imagem.

Originalidade. O arquivo de descrição `main.pov` deve ser construído manualmente, com um editor de texto comum, **sem** o auxílio de qualquer editor gráfico ou outra ferramenta de modelagem geométrica. Não é permitido copiar ou incluir quaisquer arquivos POV-Ray além dos fornecidos pelo professor ou escritos por você mesmo. Porém, é permitido re-usar arquivos ou trechos de código de exercícios anteriores.

Individualidade. Lembre-se de que todo trabalho prático é **individual**. Não é permitido pedir qualquer tipo de ajuda a colegas ou outras pessoas. Dúvidas devem ser tiradas apenas com o professor.

Exportação. Não se esqueça de **exportar seu arquivo `main.pov` até o final da aula para sua pasta WWW pública**
`http://students.ic.unicamp.br/~ra`*SEU_RA*`/mc937-2020-2/2020-10-14/`

Comandos. Os comandos de POV-Ray relevantes para este laboratório são:

- Definição de macro:
  `#macro` *nome* `(`*parâmetros*`)`
  comandos
  `#end`

- Declaração e atribuição de variáveis locais dentro de uma macro:
  `#local` *nome* `=` *número ou vetor*`;`
  `#local` *nome* `=` *objeto*
  O valor pode ser uma fórmula. Note que o ponto-e-vírgula é obrigatório se o valor for um número ou vetor, e proibido se o valor for um objeto ou textura. Note também que a chave `#local` é obrigatória inclusive para alterar o valor de uma variável local já definida. O comando `#declare` é semelhante mas define ou altera variáveis globais. textbfAtenção: Os seguintes nomes são palavras reservadas da linguagem POV-Ray: `x`, `y`, `z`, `u`, `v`, `t`.

- Retorno de resultado de uma macro: Não há comando `return` explícito. O resultado da macro é qualquer fórmula ou objeto que aparecer sem o `#local` ou `#declare`. Por exemplo:

```
#macro treco(tipo)
  #local R = 10*sqrt(2);
  union{
    sphere{ <0,0,0>, R }
    #if (tipo = 7)
      cylinder{ <0,0,R>, <0,0,2*R>, 5 }
      box{ <-10-10,2*R>, <+10,+10,2*R+10> }
    #else
      cone{ <0,0,R>, R/2, <0,0,3*R+10> 1.5*R }
    #end
  }
#end
```

  O resultado da macro será `union{ .. }`.

- Invocação de uma macro:
  nome`(`*argumentos*`)`
  Por exemplo:

```
#local raio = calcula_raio(...);
object{ treco(tt) translate <10,10,0> }
```

  A segunda forma deve ser usada quando o resultado da macro for um objeto (simples ou composto). Se o resultado for dois ou mais objetos, eles devem ser agrupados com `union{ ... }` dentro da macro.

- Comando condicional:
  `#if (`*condição*`)`
  comandos
  `#else`
  comandos
  `#end`
  A textitcondicão pode usar operadores de comparação `=` (note: não `==`), `!=`, `>`, `<`, `>=`, `<=`, ou operadores lógicos `&` ("e"), `|` ("ou"), `!` ("não"). A parte `#else` *comando* pode ser omitida.

- Comando de repetição:
  `#while (`*condição*`)`
  comandos
  `#end`
  Este comando funciona como o similar em C e outras linguagens. Por exemplo:

```
union{
  #local kk = 0;
  #while (kk < N)
    sphere{ <0, 0, kk*kk >, 5 translate < 0, kk*10, 0> }
    #local kk = kk+1;
  #end
}
```