MC937A/MO603A – Computação Gráfica - 2021-S2 - Jorge Stolfi
Trabalho de laboratório 05 - 2021-09-22
Robô alieniginóide

Objetivos: treinar uso de *modelos articulados hierárquicos*, e transformações `rotate` e `scale`.

Enunciado. A arma mais temida de alienígenas invasores, como todos sabem, são os robôs humanóides que se infiltram entre nós para nos atacar "por dentro". Para equlibrar a guerra, nós precisamos desenvolver robôs alieniginóides. Que é nossa missão de hoje. (Felizmente, alienígenas tem péssima visão, então mesmo robôs toscos terão boas chances de infiltração.)

Parte 1. Antes de começar a programar, **desenhe um esboço de seu robô alieniginóide e apresente-o ao professor via Meet, quando solicitado, no início da aula.** Seu robô deve ter pelo menos quatro apêndices (pernas ou braços, sabe-se lá), formados de partes rígidas articuladas, com pelo menos 16 parâmetros de pose independentes.

Parte 2. Escreva uma macro `robo(...)` que gera um robô usando a técnica de *modelo articulado hierárquico*, a ser explicada na aula. Os argumentos devem ser os parâmetros de pose dos apêndices. A descrição deve usar um conjuto de macros auxiliares `dedo(...)`, `mao(...)`, `braco1(...)`, `braco2(...)`, `perna(...)` que geram os sub-componentes (sub-árvores) do robô.

Cada macro gera sua componente numa posição e orientação fixas, com o ponto de ancoragem na origem. Cada grau de liberdade *interno* da componente deve ser um parâmetro separado da macro. Por exemplo, se um dedo tem uma articulação interna, a macro `dedo` deve ter 1 parâmetro, que é o ângulo nessa articulação, Se a mão tem 7 dedos, cada um com 2 graus de liberdade na sua base, a macro `mao` deve ter $7 \times (1 + 2) = 21$ argumentos; e assim por diante.

No final, mostre uma cena com três instâncias do robô, em poses diferentes.

Arquivos. Copie os arquivos da aula passada para uma nova sub-pasta `2021-09-22` 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*`/mc937-2021-2/2021-09-22/`

Comandos. Os seguintes comandos de POV-Ray são relevantes para esta tarefa:

Vetores canônicos. As variáveis `x`, `y`, e `z` são reservadas na linguagem POV-Ray, e tem valores `<1,0,0>`, `<0,1,0>`, e `<0,0,1>`, respectivamente. Assiim, por exemplo, `#local pp = 30*z - 5*x;` equivale a `#local pp = <-5,0,30>;`.

Rotaçao. Ao definir ou instanciar um objeto, o modificador `rotate` *vetor* pode ser usado antes do '}' final (como `translate`) para rodar o mesmo. Se o *vetor* é `<20,10,60>`, o objeto será rodado por 20 graus em torno do eixo X, 10 graus em torno do eixo Y, e 60 graus em torno do eixo Z, nessa ordem. O sentido da rotação é sempre o contrário do que você espera. Por exemplo, `cylinder{ <0,0,10>, <0,0,20>, 5 rotate 20*x }` é um cilindro inicialmente vertical, rodado 20 graus em torno do eixo X (ou seja, inclinado 20 graus na direção do eixo Y).

Escala. Outro modificador que pode ser usado na mesma posição é `scale` *vetor*. Se o *vetor* é `<2.0, 0.5, 1.0>`, o objeto será esticado por um fator de 2 na direção do eixo X e encolhido pela metade na direção do eixo Y, sem alteração na direção Z. Por exemplo, `box{ <4,4,4>, <10,10,10> scale <2.0, 0.5, 1.0> }` equivale a `box{ <8,2,4>, <20,5,10> }` Fatores de escala negativos invertem o objeto na direçao correspondente; por exemplo, se `object{ luva }` gera uma luva para mão esquerda perto de `<10,20,30>`, então `object{ luva scale <-1.0, +1.0, +1.0> }` gera uma luva para a mão direita, perto de `<-10,20,30>`.

Combinando modificadores. É possível aplicar dois ou mais modificadores `translate`, `rotate`, e `scale` no memo objeto; porém, a ordem geralmente é importante. Dado `#local poste = cylinder{ <0,0,0>, <0,0,10>, 3}`, as chamadas

```
object{ poste translate 10*z rotate 90*x scale <2,3,4> }
object{ poste rotate 90*x translate 10*z scale <2,3,4> }
object{ poste rotate 90*x scale <2,3,4> translate 10*z }
```

etc. produzem resultados bem diferentes, assim como

```
object{ poste rotate 90*x rotate 90*z }
object{ poste rotate 90*z rotate 90*x }
```

Centro da rotação e escala. Note que os modificadores `rotate` e `scale` são sempre relativos à origem, e não ao centro do objeto. Para rodar um objeto `coisa` em relação a um ponto arbitrário `ctr`, por exemplo seu centro, o objeto deve ser deslocado de modo a colocar esse ponto na origem, antes de aplicar a rotação, e depois deslocado de volta: `object{ coisa translate -ctr rotate 30*x translate +ctr}`.

Lembrete. As palavras `x`, `y`, `z`, `u`, `v`, `t` são reservadas pela linguagem POV-Ray e não podem ser usadas como nomes de variáveis. Use letras maiúsculas, ou nomes com 2 ou mais letras.