MC937A/MO603A – Computação Gráfica - 2020-S2 - Jorge Stolfi
Trabalho de laboratório 05 - 2020-11-04
Robogarçom II

Objetivos: treinar o conceito de matrizes e de interpolação linear e de Bézier.

Enunciado. Um robô garçom deve ter movimentos suaves e precisos, por exemplo para servir um copo de vinho, recolher um prato, ou limpar a sopa que derramou na cabeça do freguês. Sua tarefa hoje é programar o robogarçom da aula passada para produzir tal movimento.

Parte 1. Antes de começar a programar, **desenhe um esboço do seu robogarçom em quatro poses ao longo do movimento planejado,** e apresente-o ao professor via Meet, quando solicitado, no início da aula.

Parte 2. Produza uma macro `robo_mov(tt)` que chama sua macro `robo` da aula passada com parâmetros tais que produz a posição em um instante arbitrário `tt` do movimento; sendo `tt=0` a posição inicial e `tt=1` a posição final. Para isso, defina uma macro auxilar `robo_vet` que recebe um vetor de parâmetros, e chama a sua macro `robo` com os elementos desse vetor como parâmetros separados.

Arquivos. Copie os arquivos da aula passada para uma nova sub-pasta `2020-11-04` da pasta `mc937` no seu computador. Edite o arquivo `main.pov`, conforme solicitado acima. Execute o comando `make` numa shell para gerar a imagem.

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*`/mc937-2020-2/2020-11-04/`

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

Declaração de matriz. o comando `#local` *nome* `= array[`n`]` define um vetor chamado *nome* com n elementos. Como em C ou Python, os elementos são *nome*`[0]`, *nome*`[1]`, ..., *nome*$[n-1]$.

Para atribuir valores, use `#local` *nome*`[`*índice*`] =` *valor*`;`, onde o *índice* pode ser qualquer fórmula com valor inteiro. Cada elemento de um vetor pode guardar qualquer valor do POV-Ray que possa ser atribuído a uma variável — número, ponto, objeto, textura, etc.. Porém, todos os elementos devem ter o mesmo tipo. O ponto-e-vírgula no final do comando deve ser usado se e somete se o *valor* é um número ou vetor.

Interpolação afim. Dados dois pontos `(xx0,yy0)` e `(xx1,yy1)` no gráfico de uma função, o valor para um argumento `xx` genérico pode ser calculado por `#local yy = interpola1(xx, xx0,xx1, yy0,yy1)`, onde `interpola1` é a macro

```
#macro interpola1(tt, tt0,tt1, vv0,vv1)
  #local rr = (tt - tt0)/(tt1 - tt0);
  #local vv = (1-rr)*vv0 + rr*vv1;
  vv
#end
```

Note a maneira de retornar um valor de uma macro que vai ser usada como uma função. Este algoritmo é chamado de *interpolação afim* porque o resulado é uma *função afim* do parâmetro `tt`, ou seja um polinômio de grau 1: $A\,\texttt{tt} + B$. (Estes conceitos são incorretamente chamados *interpolação linear* e *função linear*.)

Interpolação de Bézier. Uma interpolação mais complicada é dada pela macro

```
#macro interpola3(tt, tt0,tt3, vv0,vv1,vv2,vv3)
  #local vv01 = interpola1(tt, tt0,tt3, vv0,vv1);
  #local vv12 = interpola1(tt, tt0,tt3, vv1,vv2);
  #local vv23 = interpola1(tt, tt0,tt3, vv2,vv3);
  #local vv012 = interpola1(tt, tt0,tt3, vv01,vv12);
  #local vv123 = interpola1(tt, tt0,tt3, vv12,vv23);
  #local vv0123 = interpola1(tt, tt0,tt3, vv012,vv123);
  vv0123
#end
```

Este algoritmo calcula um polinômio de grau 3 que começa no ponto `(tt0,vv0)` e termina no ponto `(tt3,vv3)`. As derivadas no início e fim do intervalo são determinadas pelos valores vv1 e vv2.