

**MC937A/MO603A – Computação Gráfica - 2021-S2 - Jorge Stolfi**  
**Trabalho de laboratório 06 - 2021-09-29**  
**Robôs com EI (Elegância Artificial)**

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

**Enunciado.** Infelizmente, os robôs alienígenóides que tentamos infiltrar nas hordes inimigas na semana passada foram imediatamente reconhecidos e destruídos. Consta que o problema foi a deselegância de seus movimentos. Como todos sabem, alienígenas são batráquios; embora tenham péssima visão para objetos parados, são muito sensíveis a movimentos. Então nossa missão de hoje é desenvolver o controle de movimentos suaves para nossos robôs.

**Parte 1.** Antes de começar a programar, **desenhe um esboço do seu robô em quatro poses ao longo de um movimento suave 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 auxiliar `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 `2021-09-29` 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 as 21:00 para [http://students.ic.unicamp.br/~raseu\\_ra/mc937-2021-2/2021-09-29/](http://students.ic.unicamp.br/~raseu_ra/mc937-2021-2/2021-09-29/)**

**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 somente 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 resultado é uma *função afim* do parâmetro  $tt$ , ou seja um polinômio de grau 1:  $A tt + B$ . (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  $(tt_0, vv_0)$  e termina no ponto  $(tt_3, vv_3)$ . As derivadas no início e fim do intervalo são determinadas pelos valores  $vv_1$  e  $vv_2$ .

