MC937A/MO603A – Computação Gráfica - 2020-S2 - Jorge Stolfi
Trabalho de laboratório 08 - 2020-11-25
Vírus tentacular animado

Objetivos: Treinar animação de movmentos suaves usando interpolação cúbica de Hermite.

Enunciado. Parece que a população não ficou suficientemente impressionada com o vírus COVID-19 mutante tentaculado da aula anterio. Quem sabe uma animação ameaçadora do mesmo ajuda.

Parte 1. Não haverá parte 1 nesta aula.

Parte 2. Produza uma macro `virus_anim(tt)` que devolve o virus, como explicado na página seguinte.

Arquivos. Copie os arquivos da aula `2020-11-11` (animação, não imagem estática) para uma nova sub-pasta `2020-11-25` da pasta `mc937` no seu computador. Copie o `main.pov` da aula passsada `2020-11-18` eedite conforme explicado abaixo. Execute o comando `make fast`, `make strip`, `maxe movie` etc. 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*`/mc937-2020-2/2020-11-25/`

Notas de implementação:

OBS: A versão distribuída na aula de 2020-11-25 tinha erros na macro `interpola3H` (as fórmulas de `dh0` e `dh1`) e na macro `encontra_intervalo` (`NQ-1` em vez de `NQ`). Além disso, o valor de `mV` em `tentaculo_aleatorio_anim` era muito pequeno, resultando em paradinhas nos quadros-chave. Esses erros foram corrigidos nesta versão.

- **Interpolação de Hermite.** Outra maneira de obter uma função cúbica que intepola certos dados é dada pela macro

```
#macro interpola3H(tt, t0,t1, p0,d0,d1,p1)
  #local rr = (tt - t0)/(t1 - t0);
  #local ss = 1 - rr;
  #local ph0 = rr*rr*(2*rr - 3) + 1;
  #local dh0 = + rr*ss*ss * (t1 - t0);
  #local dh1 = - ss*rr*rr * (t1 - t0);
  #local ph1 = ss*ss*(2*ss - 3) + 1;
  #local pp = ph0*p0 + dh0*d0 + dh1*d1 + ph1*p1;
  pp
#end
```

  Esta macro calcula uma polinômio de grau 3 na variável `tt`. O valor desse polinômio é `p0` quando `tt` = `t0` e `p1` quando `tt` = `t1`.

  A derivada em relação a `tt` é `d0` quando `tt` = `t0` and `d1` quando `tt` = `t1`. Os valores `p0` e `p1` podem ser pontos; nesse caso, `d0` e `d1` são velocidades.

- **Tentáculo com interpolação de hermite.** Você vai precisar das macros
  `segmento(N, P0,P1,P2,P3, R0, R3)`
  `tentaculo(N, A0,A1,A2,B1,B2,C1,C2,C3, RA0,RC3)`
  da aula anterior. Além disso você vai precisar de uma nova macro
  `tentaculo_hermite(N, tt,t0,t1, A0, A1H,A2H,B1H,B2H,C1H,C2H,C3H, RA0,RC3)`
  que devolve o tentáculo animado num instante `tt` entre `t0` e `t1`.

  Esta macro é parecida com macro `tentaculo` da aula passada, exceto que apenas texttttA0 é um ponto. Cada um dos parâmetros `A1H,A2H,B1H,B2H,C1H,C2H,C3H` será um array de 4 elementos, onde os elementos `[0]` e `[3]` são a posição inicial e final do ponto correspondente de `tentaculo`, e os elementos `[1]` `[2]` são as velocidades inicial e final desse ponto. esta macro calcula pontos `A1,A2,B1,B2,C1,C2,C3` para o tempo dado `tt`, com interpolaçãode Hermite, e chama `tentaculo`.

- **Testando o tentaculo.** Para testar, use a seguinte macro:

```
  #macro tentaculo_hermite_teste(tt)
    #local A0 = < 0, 0, 0 >;
    #local A1H = parado(< 1, 0, 0 >);
    #local A2H = parado(< 2, 0, 0 >);
    #local B1H = parado(< 4, 0, 0 >);
    #local B2H = parado(< 5, 0, 0 >);
    #local C1H = parado(< 7, 0, 0 >);
    #local C2H = parado(< 8, 0, 0 >);

    #local C3H = array[4];
    #local C3H[0] = < 9, 0, -1 >;
    #local C3H[1] = < 0, +1, 0 >;
    #local C3H[2] = < 0, -1, 0 >;
    #local C3H[3] = < 9, 0, +1 >;
    #local RA0 = 0.2;
    #local RC3 = 0.02;
    object{ tentaculo_hermite(N, tt,0,1, A0, A1H,A2H,B1H,B2H,C1H,C2H,C3H, RA0,RC3)
}
  #end

  #macro parado(P)
    #local PH = array[4];
    #local PH[0] = P;
    #local PH[1] = < 0, 0, 0 >;
    #local PH[2] = < 0, 0, 0 >;
    #local PH[3] = P;
    PH
  #end
```

- **Tentáculo animado.** Em seguida você precisa de uma macro que gera um tentáculo animado para um tempo `tt` de animação entre 0 e 1, quebrando a animação entre pelo menos três intervalos de tempo, dados as posições e velocidades dos pontos de controle nos quadros chave:

```
#macro tentaculo_anim(N,tt, A0,
    NQ,TQ, PA1,DA1, PA2,DA2, PB1,DB1, PB2,DB2, PC1,DC1, PC2,DC2, PC3,DC3,
    RA0,RC3)

  // Determina os quadros chave a interpolar k0,k1 e os tempos t0,t1:
  #local k0 = encontra_intervalo(tt, NQ, TQ); #local k1 = mod(k0 + 1, NQ);
  #local t0 = TQ[k0]; #local t1 = TQ[k0 + 1];

  // Pega posicoes e velocidades nesses dois quadros chave:
  #local A1H = array[4];
  #local A1H[0] = PA1[k0];
  #local A1H[1] = DA1[k0];
  #local A1H[2] = DA1[k1];
  #local A1H[3] = PA1[k1];

  #local A2H = array[4];
  #local A2H[0] = PA2[k0];
  #local A2H[1] = DA2[k0];
  #local A2H[2] = DA2[k1];
  #local A2H[3] = PA2[k1];
  ...
  #local C3H = array[4];
  #local C3H[0] = PC3[k0];
  #local C3H[1] = DC3[k0];
  #local C3H[2] = DC3[k1];
  #local C3H[3] = PC3[k1];

  // Gera o tentaculo entre esses dois quadros chave:
  object{ tentaculo_hermite(N, tt,t0,t1, A0, A1H,A2H,B1H,B2H,C1H,C2H,C3H, RA0,RC3)
}
#end
```

Nesta macro, o parâmetro `NQ` é o número de quadros-chave **excluindo**o quadro final que é igual ao inicial. O parâmetro `TQ` deve ser um array de `NQ+1` posições com os tempos (clocks) dos quadros chaves **incluindo** o quadro final, começando com `TQ[0] = 0.0000` e terminando com `TQ[NQ] = 1.0000`. Os parâmetros `PA1,DA1` são arrays de NQ elementos; `PA1` contém as posições do ponto `A1` do tentáculo em cada quadro chave, e `DA1` contém as velocidades correspondentes. Os parãmetros `PA2,DA2,...,PC3,DC3` são análogos. Os demais parâmetros tem o mesmo sentido da aula passada: `N` é o número de bolas, `A0` é a posição da base do tentáculo (suposta fixa), e `RA0,RC3` são os raios do tentáculo na base e na ponta.

- **Tentáculo aleatório animado.** Finalmente, você precisa de uma macro que gera os dados para `tentaculo_anim`. A macro a seguir gera esses dados aleatoriamente. Modifique-a se quiser um movmimento mais significativo:

```
#macro tentaculo_aleatorio_anim(N,it,tt,A0,Z3,RA0,RC3)
  #local roleta = seed(4615 + 417*it); // Roleta para este tentaculo.
  #local NQ = 3; // Quadros-chave excluindo final (= inicial).
  #local TQ = array[NQ+1]; // Tempos dos quadros-chave
  #local TQ[0] = 0.0000; // Inicio da animacao.
  #local TQ[1] = 0.3333; // Fim do primeiro movimento.
  #local TQ[2] = 0.6667; // Fim do segundo movimento.
  #local TQ[3] = 1.0000; // Fim da animacao.

  // Gera posicoes e velocidades aleatorias do tentaculo para quadros-chave
  #local mP = 0.5*vlength(Z3 - A0); // Max perturbacao do ponto.
  #local mV = 4.0*vlength(Z3 - A0); // Max velocidade.
  #local PA1 = array[NQ]; #local DA1 = array[NQ];
  #local PA2 = array[NQ]; #local DA2 = array[NQ];
  #local PB1 = array[NQ]; #local DB1 = array[NQ];
  #local PB2 = array[NQ]; #local DB2 = array[NQ];
  #local PC1 = array[NQ]; #local DC1 = array[NQ];
  #local PC2 = array[NQ]; #local DC2 = array[NQ];
  #local PC3 = array[NQ]; #local DC3 = array[NQ];
  #local kq = 0;
  #while (kq < NQ)
    #local PA1[kq] = interpola1(1/9, 0,1, A0,Z3) + (1/9)*mP*ranvec(roleta);
    #local DA1[kq] = (1/9)*mV*ranvec(roleta);

    #local PA2[kq] = interpola1(2/9, 0,1, A0,Z3) + (2/9)*mP*ranvec(roleta);
    #local DA2[kq] = (2/9)*mV*ranvec(roleta);
    ...
    #local PC2[kq] = interpola1(8/9, 0,1, A0,Z3) + (8/9)*mP*ranvec(roleta);
    #local DC2[kq] = (8/9)*mV*ranvec(roleta);

    #local PC3[kq] = Z3 + mP*ranvec(roleta);
    #local DC3[kq] = mV*ranvec(roleta);

    #local kq = kq + 1;
  #end;
  object{ tentaculo_anim(N,tt, A0, NQ,TQ, PA1,DA1, ..., PC3,DC3, RA0,RC3) }
#end;
```

Nesta macro, o parâmetro `it` é o número do tentáculo, e `Z3` é a posição média da ponta do tentáculo. Os demais parâmetros são como nas outras macros. Use `it` se quiser fazer com que certos tentáculos façam algum movimento especial.

- **Macro auxiliar para encontrar quadros chaves.**

```
#macro encontra_intervalo(tt, NQ,TQ)
  #local k0 = 0; // Por via das duvidas.
  #local kq = 0;
  #while (kq < NQ)
    #if ((TQ[kq] <= tt) & (tt <= TQ[kq+1]))
      #local k0 = kq;
    #end
    #local kq = kq + 1;
  #end
  k0
#end
```

- **Macro auxiliar para perturbação.** A macro a seguir gera um vetor do $\mathbb{R}^3$ com coordenadas aleatórias entre -1 e $+1$:

```
#macro ranvec(roleta)
  #local vv = < 2*rand(roleta)-1, 2*rand(roleta)-1, 2*rand(roleta)-1 >;
  vv
#end
```

- **Virus animado.** Sua nova macro `virus_anim` deve chamar `tentaculo_aleatorio_anim` tantas vezes quanto desejado, com a variável externa `clock` no parâmetro `tt`.