Modelação de uma rede social com base num grafo com arcos de largura, orientação e inclinação arbitrárias. Detecção de colisões em voo livre e em voo rasante.

João Paulo Jorge Pereira

Dezembro de 2012

Índice

Índice de figuras

1 Convenções e terminologia

1.1 Identificadores

- **Constantes e macros**: expressas em maiúsculas (por exemplo: *K_ESFERA*);
- **Variáveis**: expressas em minúsculas (por exemplo: *dir*);

1.2 Cena

- **Direcção e sentido do vector "*para cima*"**: os correspondentes ao semieixo positivo dos ZZ.

1.3 Nó n_i

- **Localização**: ponto de coordenadas (x_i, y_i, z_i);
- **Largura**: w_i (a largura de um nó será igual à maior das larguras dos arcos que convergem/divergem nesse/desse nó).

1.4 Arco a_{ij}

- **Ligação**: do nó n_i ao nó n_j;
- **Desnível**: $h_{ij} = z_j - z_i$;
- **Comprimento**: s_{ij};
- **Largura**: w_{ij};
- **Orientação**: $\alpha_{ij} = \arctan^{*}((y_j - y_i) / (x_j - x_i))$ (em radianos);
- **Inclinação**: β_{ij} (em radianos).

1.5 Grafo

- **Cota mínima**: $z_{min} = \min(z_i)$;
- **Cota máxima**: $z_{max} = \max(z_i)$.

1.6 Câmara

- **Geometria** (apenas para o voo livre): cubo imaginário de lado *DIMENSÃO_CÂMARA*;
- **Distância ao solo** (apenas para o voo rasante): *DISTÂNCIA_SOLO*;
- **Localização** (centro geométrico): ponto de coordenadas (x_P, y_P, z_P);
- **Orientação**: *dir* (em radianos);
- **Velocidade horizontal**: vel_h;
- **Velocidade vertical** (apenas para o voo livre): vel_v;
- **Velocidade total** (apenas para o voo livre): $vel = \sqrt{(vel_h^2 + vel_v^2)}$.

[*] Deverá usar-se a função atan2() em vez de atan().

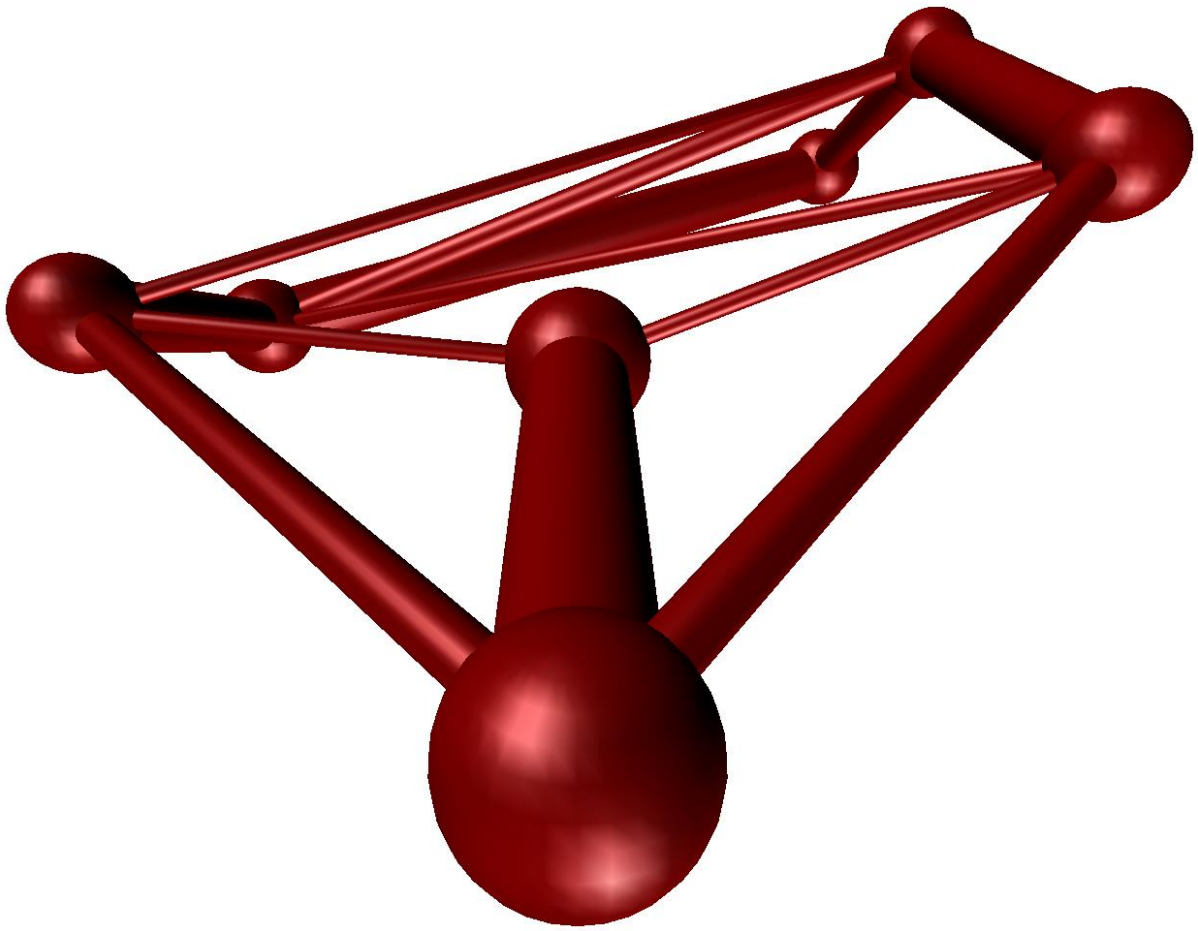

Figura 1 – Modelo da rede social

2 Modelação

A rede social poderá ser modelada da maneira que a seguir se descreve (Figura 1).

2.1 Modelação de um nó

A geometria associada a um nó n_i poderá ser a de uma esfera com as propriedades que a seguir se discriminam:

- **centro**: (x_i, y_i, z_i);
- **raio**: r_i = *K_ESFERA* * w_i / 2.0;
  em que *K_ESFERA* designa uma constante superior a 1.0 (por exemplo, *K_ESFERA* = 2.1).

Poderá ser desenhado da seguinte maneira:

- translação segundo o vector (x_i, y_i, z_i);
- desenhado com centro na origem e raio igual a r_i[†].

2.2 Modelação de um arco

A geometria associada a um arco a_{ij} poderá ser a de um cilindro com as propriedades que a seguir se discriminam:

[†] Poderá usar-se a função gluSphere().

- **comprimento da projecção no plano OXY**: $p_{ij} = \sqrt{((x_j - x_i)^2 + (y_j - y_i)^2)}$;
- **desnível**: $h_{ij} = z_j - z_i$;
- **comprimento**: $s_{ij} = \sqrt{(p_{ij}^2 + h_{ij}^2)}$;
- **raio**: $w_{ij} / 2.0$;
- **orientação**: α_{ij};
- **inclinação**: $\beta_{ij} = \arctan(h_{ij} / p_{ij})$.

Poderá ser desenhado da seguinte maneira:

- translação segundo o vector (x_i, y_i, z_i);
- rotação de GRAUS(α_{ij}) em torno do eixo dos ZZ;
- rotação de GRAUS($\pi / 2.0 - \beta_{ij}$) em torno do eixo dos YY;
- desenhado ao longo do semieixo positivo dos ZZ, assente no plano OXY, comprimento igual a s_{ij} e raio igual a $w_{ij} / 2.0$[‡].

3 Colocação da câmara na cena

Para posicionar e orientar a câmara, a matriz de modelação e visualização poderá ser definida da seguinte maneira:

- matriz identidade;
- rotação de GRAUS($-\pi / 2.0$) em torno do eixo dos XX;
- rotação de GRAUS($\pi / 2.0 - dir$) em torno do eixo dos ZZ;
- translação segundo o vector $(-x_P, -y_P, -z_P)$.

4 Movimento interactivo (controlado pelo utilizador) – voo livre

Para o voo livre, poderão implementar-se os seguintes controlos[§]:

- tecla '←': rodar para a esquerda;
- tecla '→': rodar para a direita;
- tecla '↑': avançar;
- tecla '↓': recuar;
- tecla 'Q': subir;
- tecla 'A': descer.

A localização da câmara no próximo fotograma será dada pelas seguintes equações:

- $x'_P = x_P + k * vel_h * \cos(dir)$;
- $y'_P = y_P + k * vel_h * \sin(dir)$;
- $z'_P = z_P + k * vel_v$;
  em que k designa a fracção da distância a percorrer pela câmara; o seu valor será calculado mais adiante.

[‡] Poderá usar-se a função gluCylinder().
[§] Também poderá usar-se o rato.

4.1 Detecção de colisões

A detecção de colisões assume que a câmara tem a geometria de um cubo de lado *DIMENSÃO_CÂMARA* e baseia-se na utilização do modo de selecção conjugado com uma vista na primeira pessoa em projecção ortográfica. Mais concretamente:

- a matriz de projecção deverá ser definida da seguinte maneira:
  - matriz identidade;
  - glOrtho(-*DIMENSÃO_CÂMARA* / 2.0, *DIMENSÃO_CÂMARA* / 2.0,
    -*DIMENSÃO_CÂMARA* / 2.0, *DIMENSÃO_CÂMARA* / 2.0,
    0.0, *DIMENSÃO_CÂMARA* / 2.0 + *vel*);
    em que *vel* designa a velocidade total que resulta da combinação das velocidades horizontal e vertical da câmara;
- a matriz de modelação e visualização deverá ser a correspondente a uma vista na primeira pessoa, orientada na direcção e sentido que resultam da combinação dos movimentos horizontal e vertical. Poderá ser definida da seguinte maneira:
  - matriz identidade;
  - rotação de GRAUS(-π / 2.0 - arctan[**](*vel*$_v$, *vel*$_h$)) em torno do eixo dos XX;
  - rotação de GRAUS(π / 2.0 - *dir*) em torno do eixo dos ZZ;
  - translação segundo o vector (-x_P, -y_P, -z_P);
- note-se que a definição da matriz de modelação e visualização recorre às coordenadas da localização corrente da câmara (x_P, y_P, z_P) e não às da nova localização (x'_P, y'_P, z'_P);
- desenha-se os elementos constituintes do grafo, bem como quaisquer outros objectos com os quais se pretenda testar a colisão. Não será necessário atribuir-lhes nomes (no *stack* de nomes), a menos que se pretenda, por algum motivo, identificá-los inequivocamente mais tarde;
- haverá colisão se e só se a execução do desenho no modo de selecção assim definido tiver como resultado um conjunto não vazio.

Verificando-se esta condição, é necessário ajustar a localização da câmara de modo a fazê-la coincidir com o ponto de colisão. A realização deste ajustamento é particularmente importante nas situações em que a velocidade *vel* é elevada:

- inspecciona-se o *buffer* de resultados de maneira a identificar-se o menor dos valores mínimos de profundidade;
- sabendo as coordenadas do centro da janela e o menor dos valores mínimos de profundidade, recorre-se à função gluUnProject() para determinar o ponto (x''_P, y''_P, z''_P) da cena onde se verificou a colisão;
- a distância da câmara ao ponto de colisão será dada pela seguinte equação:
  - $d = \sqrt{(x''_P - x_P)^2 + (y''_P - y_P)^2 + (z''_P - z_P)^2}$;
- a fracção da distância a percorrer pela câmara será dada pela seguinte equação:
  - $k = (d$ - *DIMENSÃO_CÂMARA* / 2.0 - *INFINITÉSIMO*$) /$ *vel*.

Não se verificando qualquer colisão, a fracção da distância a percorrer pela câmara será igual à unidade:

[**] Deverá usar-se a função atan2() em vez de atan().

- $k = 1.0$.

Independentemente de se ter ou não verificado uma colisão, as coordenadas actualizadas da localização da câmara serão dadas pelas equações:

- $x_P = x'_P$;
- $y_P = y'_P$;
- $z_P = z'_P$.

5 Movimento interactivo (controlado pelo utilizador) – voo rasante

Para o voo rasante, poderão implementar-se os seguintes controlos[††]:

- tecla '←': rodar para a esquerda;
- tecla '→': rodar para a direita;
- tecla '↑': avançar;
- tecla '↓': recuar.

Deverá manter-se um registo actualizado da localização da câmara no grafo, ou seja, se esta se encontra num dado nó (o nó n_i) ou num dado arco (o arco a_{ij}).

Caso não houvesse colisão, a localização da câmara no próximo fotograma seria dada pelas seguintes equações (apenas a abcissa e a ordenada; a cota será calculada mais adiante):

- $x'_P = x_P + vel_h * \cos(dir)$;
- $y'_P = y_P + vel_h * \sin(dir)$.

5.1 Detecção de colisões num nó

Caso a câmara se encontre no nó n_i do grafo, não haverá colisão se o ponto correspondente à nova localização pertencer:

- a esse nó;
- a um dos arcos que convergem/divergem nesse/desse nó.

5.2 Detecção de colisões num arco

Caso a câmara se encontre no arco a_{ij} do grafo, não haverá colisão se o ponto correspondente à nova localização pertencer:

- a esse arco;
- ao nó n_i;
- ao nó n_j.

5.3 Determinação de pertença – primeira abordagem

A primeira abordagem é puramente matemática e restringe significativamente a complexidade da geometria dos nós e dos arcos constituintes do grafo, sob pena de os cálculos se tornarem demasiado complicados. Baseia-se no facto de a projecção ortogonal de uma esfera e de um cilindro no plano OXY corresponderem a um círculo e um rectângulo, respectivamente.

[††] Também poderá usar-se o rato.

5.3.1 Pertença de um ponto a um nó

O ponto correspondente à nova localização da câmara pertencerá ao nó n_i se e só se a sua projecção no plano OXY pertencer ao círculo (esfera projectada) que o representa. Por outras palavras, se e só se a distância do ponto projectado ao centro do círculo não for superior ao raio:

- $(x'_P - x_i)^2 + (y'_P - y_i)^2 \leq r_i^2$.

Verificando-se esta condição, considera-se que a câmara passa a estar (caso não estivesse já) localizada no nó n_i do grafo. As coordenadas da nova localização serão dadas pelas equações:

- $x_P = x'_P$;
- $y_P = y'_P$;
- $z_P = z_i + DISTÂNCIA\_SOLO$.

5.3.2 Pertença de um ponto a um arco

Para determinar se o ponto correspondente à nova localização da câmara pertence ao arco a_{ij}, poderá proceder-se da maneira que a seguir se descreve:

Efectua-se uma mudança de sistema de coordenadas que verifique as seguintes condições:

- faça coincidir a nova origem com o ponto (x_i, y_i);
- alinhe o novo eixo dos XX com o eixo longitudinal da projecção do arco no plano OXY.

Neste novo sistema, as coordenadas correspondentes à nova localização da câmara serão dadas pelas seguintes equações:

- $x''_P = (x'_P - x_i) * \cos(\alpha_{ij}) + (y'_P - y_i) * \sin(\alpha_{ij})$;
- $y''_P = (y'_P - y_i) * \cos(\alpha_{ij}) - (x'_P - x_i) * \sin(\alpha_{ij})$.

O ponto correspondente à nova localização da câmara pertencerá ao arco se e só se a sua projecção no plano OXY não ultrapassar os limites do rectângulo (cilindro projectado) que o representa:

- $0.0 < x''_P < p_{ij}$;
- $-w_{ij} / 2.0 \leq y''_P \leq w_{ij} / 2.0$.

Verificando-se estas condições, considera-se que a câmara passa a estar (caso não estivesse já) localizada no arco a_{ij} do grafo. As coordenadas da nova localização serão dadas por equações que se assemelham às da pertença a um nó. A única diferença reside na inclusão de uma regra de três simples no cálculo da cota:

- $x_P = x'_P$;
- $y_P = y'_P$;
- $z_P = z_i + x''_P / p_{ij} * h_{ij} + DISTÂNCIA\_SOLO$.

5.4 Determinação de pertença – segunda abordagem

A segunda abordagem é mais flexível do que a anterior, na medida em que não impõe grandes restrições à geometria dos nós e dos arcos constituintes do grafo. Também é mais simples, pois dispensa a realização de qualquer mudança de sistema de coordenadas. Baseia-se na utilização do modo *picking* conjugado com uma vista superior em projecção ortográfica centrada no ponto do

plano OXY correspondente à nova localização da câmara, ou seja, o ponto (x'_P, y'_P, 0.0). Mais concretamente:

- o centro da região de *picking* deverá coincidir com o centro da janela;
- a matriz de projecção poderá ser definida da seguinte maneira:
  - matriz identidade;
  - glOrtho(x'_P - vel_h, x'_P + vel_h, y'_P - vel_h, y'_P + vel_h, *near*, *far*);
    em que *near* < $-z_{max}$ e *far* > $-z_{min}$ [‡‡];
- a matriz de modelação e visualização poderá ser a matriz identidade;
- desenha-se apenas os elementos do grafo que se pretende testar, tendo o cuidado de lhes atribuir nomes (no *stack* de nomes) que permitam identificá-los inequivocamente mais tarde;
- o ponto correspondente à nova localização da câmara pertencerá a um destes elementos se e só se a execução do desenho no modo *picking* assim definido tiver como resultado um conjunto não vazio;
- verificando-se esta condição, inspecciona-se o *buffer* de resultados de maneira a identificar-se o elemento do conjunto com a cota mais elevada, ou seja, aquele a que corresponde o menor dos valores mínimos de profundidade;
- sabendo as coordenadas do centro da janela e o menor dos valores mínimos de profundidade, recorre-se à função gluUnProject() para determinar o ponto da cena correspondente (x''_P, y''_P, z''_P). Na realidade apenas se pretende calcular a cota z''_P, pois os valores de x''_P e de y''_P já são conhecidos: x''_P será forçosamente igual a x'_P; e y''_P igual a y'_P.

5.4.1 Pertença de um ponto a um nó

Se o elemento identificado pela execução do desenho no modo *picking* corresponder ao nó n_i, considera-se que a câmara passa a estar (caso não estivesse já) localizada neste nó do grafo. As coordenadas da nova localização serão dadas pelas equações:

- $x_P = x''_P$;
- $y_P = y''_P$;
- $z_P = z''_P$ + *DISTÂNCIA_SOLO*.

5.4.2 Pertença de um ponto a um arco

Se o elemento identificado pela execução do desenho no modo *picking* corresponder ao arco a_{ij}, considera-se que a câmara passa a estar (caso não estivesse já) localizada neste arco do grafo. As coordenadas da nova localização serão dadas por equações idênticas às da pertença a um nó:

- $x_P = x''_P$;
- $y_P = y''_P$;
- $z_P = z''_P$ + *DISTÂNCIA_SOLO*.

[‡‡] Se estiver a ser usada uma transformação de escala no desenho do rede social (por exemplo, glScalef(s_x, s_y, s_z)), o factor segundo o eixo dos ZZ (s_z) deverá ser tido em consideração no cálculo dos planos *near* e *far*. Mais concretamente: *near* < $-z_{max} * s_z$ e *far* > $-z_{min} * s_z$.