Universidade Federal de Santa Maria
Departamento de Eletrônica e Computação
Prof. Cesar Tadeu Pozzer
Disciplina: Computação Gráfica Avançada
pozzer@inf.ufsm.br
19/08/2008

Fundamentos Matemáticos para Jogos

1. Interseções

Cálculos de interseções são úteis em diversas situações em um jogo. Pode-se determinar por exemplo se um disparo atingiu o alvo ou se um veículo colidiu com um muro. Sem testes de interseção, todos os elementos móveis do jogo poderiam atravessar paredes, por exemplo.

Para simplificação dos cálculos, testes de interseção podem ser realizados em 2D, mesmo que o ambiente seja 3D. Isso não se aplica para todos os casos. Jogos modernos fazem cálculos complexos em 3D para tratar colisão entre todos os elementos, especialmente para fazer simulações físicas.

Neste material são apresentados dois testes de interseção em 2D: linha (segmento de reta) com linha e linha com círculo. O primeiro caso se aplica quando se deseja calcular a interseção de um personagem com um muro, por exemplo. O segundo caso pode ser usado para calcular a interseção de um disparo contra um alvo aproximado por um círculo (*bounding circle*).

1.1. Interseção entre Linhas

O teste de interseção entre retas faz uso de duas retas definidas em forma paramétrica $P + sD$. **O módulo do vetor direção *D* deve ser o tamanho da linha**. Vai haver interseção se os parâmetros *s* e *t*, no ponto de interseção das linhas, estiverem no intervalo entre [0, 1]. Para achar o ponto de interseção, deve-se igualar as duas linhas e evidenciar os parâmetros *s* e *t*.

$$P_0 + sD_0 = P_1 + tD_1$$

evidenciando D_0 e D_1

$$sD_0 - tD_1 = P_1 - P_0$$

assumindo

$$D_0 = (x_0, y_0)$$
$$D_1 = (x_1, y_1)$$
$$P_0 - P_1 = (c_0, c_1)$$

tem-se

$$s(x_0, y_0) - t(x_1, y_1) = (c_0, c_1)$$

Reescrevendo como um sistema de equações 2x2 obtém-se

$$\begin{cases} x_0 s - x_1 t = c_0 \\ y_0 s - y_1 t = c_1 \end{cases}$$

$$s = \frac{c_0 + x_1 t}{x_0}$$

$$y_0 \left(\frac{c_0 + x_1 t}{x_0} \right) = c_1 + y_1 t$$

$$y_0(c_0 + x_1 t) = x_0(c_1 + y_1 t)$$
$$y_0 c_0 + y_0 x_1 t = x_0 c_1 + x_0 y_1 t$$
$$t(y_0 x_1 - x_0 y_1) = x_0 c_1 - y_0 c_0$$
$$t = \frac{x_0 c_1 - y_0 c_0}{y_0 x_1 - x_0 y_1}$$

Aplicando o mesmo cálculo para *s* obtém-se

$$s = \frac{x_1 c_1 - y_1 c_0}{y_0 x_1 - x_0 y_1}$$

Determinante dos vetores direção

O termo $y_0 x_1 - x_0 y_1$ é um cálculo de determinante e é comum aos dois casos e pode ser calculado uma única vez. O determinante neste caso diz se o sistema é LI ou LD. Se for LD (gerar valor zero), significa que as retas são paralelas, ou seja, o **ângulo entre elas é zero**, e por isso tem interseção no infinito.

```
bool intersect( Vector& P0,  Vector& D0, Vector& P1, Vector& D1 )
{
  float Det = D1.x*D0.y - D1.y*D0.x;
  Vector Diff( P1 - P0 );


  //intersecao das linhas em um único ponto. Calcula s e t.
  if (Det*Det > 0.001 * D0.norm() * D1.norm() )
  {
     double InvDet = 1.0/Det;
     double InvDet = 1.0/Det;
     g_s = (D1.x*Diff.y - D1.y*Diff.x)*InvDet;
     g_t = (D0.x*Diff.y - D0.y*Diff.x)*InvDet;

     if( g_t > 0 && g_t < 1 && g_s > 0 && g_s < 1 )
        return true;
  }
  //linhas paralelas
  return false;
}
```

1.2. Interseção entre Linha e Círculo (por Henrique Vicentini, Cesar Pozzer)

Um circulo de raio r com o centro na origem é definida pela equação: $x^2 + y^2 = r^2$. Levando em consideração a reta descrita de forma paramétrica definimos que:

$$x = q_x + t v_x$$
$$y = q_y + t v_y$$

substituindo na equação do circulo:

$$(q_x + t v_x)^2 + (q_y + t v_y)^2 = r^2$$

expandindo as potências e reagrupando temos:

$$(v_x^2 + v_y^2)t^2 + 2(q_x v_x + q_y v_y)t + q_y^2 + q_x^2 - r^2 = 0$$

assim chegamos em uma equação de grau 2 na forma: $ax^2 + bx + c = 0$ no qual:

$a = v_x^2 + v_y^2 = V^2$ coeficiente angular da reta ao quadrado

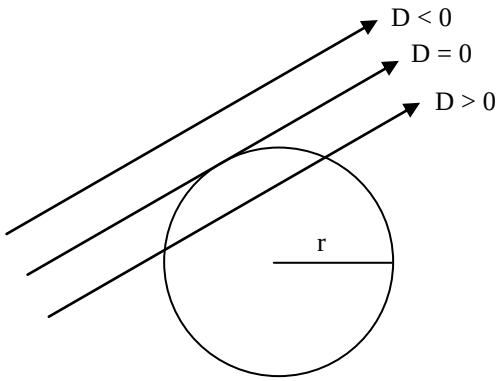
$b = 2(q_x v_x + q_y v_y) = 2(Q.V)$ produto escalar entre o coeficiente linear e o angular da reta

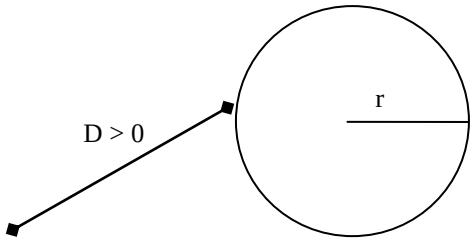
$c = q_y^2 + q_x^2 - r^2 = Q^2 - r^2$ coeficiente linear da reta ao quadrado subtraído do quadrado do raio do circulo

A partir da equação de segundo grau podemos chegar nas raízes da mesma aplicado a fórmula de Báscara aonde o delta D determina se existem raízes para a equação conforme mostrado na figura a seguir:

$$t = \frac{-b \pm \sqrt{D}}{2a}$$

$$D = b^2 - 4ac$$





Porém, somente encontrar o valor de D não necessariamente implica na intersecção da reta com o círculo, pois a reta pode terminar antes de interceptar o circulo.

Assim podemos determinar o número de possíveis pontos de colisão com o círculo, caso:

- **D<0** não teremos pontos de interseção;
- **D=0** teremos um possível ponto de intersecção;
- **D>0** teremos dois possíveis pontos de intersecção.

$$t' = \frac{-b + \sqrt{D}}{2a} \quad t'' = \frac{-b - \sqrt{D}}{2a}$$

Como a reta utilizada é paramétrica, basta determinarmos se o valor de *s* e *t* pertencem ao intervalo [0,1] para determinarmos se a reta intercepta o círculo. Para encontramos os pontos de x e y equivalente basta substituirmos os valores de *t* na equação paramétrica da reta.

```
bool intersects(Circle &c, float &s, float &t)
{
   float aQuad, bQuad, cQuad, r2, delta;
   // é subtraido o centro da circunferencia para centrarmos a circunferencia na origem
   Vector pi1_2(p.x - c.p.x,  p.y - c.p.y);

   // o raio ao quadrado da circunferencia
   r2 = c.r * c.r;

   // a da quadrica
   aQuad = d.x*d.x + d.y*d.y;
   // b da quadrica
   bQuad = 2*(pi1_2.x*d.x + pi1_2.y*d.y);
   // c da quadrica
   cQuad = pi1_2.x*pi1_2.x + pi1_2.y*pi1_2.y - r2;

   // calcula o delta da quadrica
   delta = bQuad*bQuad - 4*aQuad*cQuad;

   // se negativo nao colide em hipotese alguma
   if( delta < 0 )
      return false;
   else
   {
      // calcula os pontos de interseccao
      s = ((-bQuad)+sqrt(delta)) / (2*aQuad);
      t = ((-bQuad)-sqrt(delta)) / (2*aQuad);

      if( (t > 0 && t < 1) || (s > 0 && s < 1) )
         return true;
   }
   return false;
}
```

1.3. Material Adicional

Existem diversos outros testes de interseção presentes na literatura [1, 2, 3, 4, 5] que tratam colisão entre objetos dinâmicos e estáticos. Dente alguns deles pode-se mencionar:

- Interseção linha-triângulo
- Interseção linha-cilindro
- Interseção linha-elipsoide
- Interseção plano-esfera
- Interseção plano-caixa orientada
- Interseção esfera-esfera
- Interseção triângulo-caixa

3. Exercícios

1. Elabore uma hierarquia de classes e um algoritmo para evitar que um personagem atravesse a parede de um prédio. Este prédio pode ser representado por 4 paredes. Imagine que no cenário existam vários prédios. O personagem deve parar no momento exato que colidir com o prédio. Considere o fato da aplicação estar rodando com baixo FPS e o personagem andando com alta velocidade.
2. Elabore uma hierarquia de classes e um algoritmo para determinar se um tiro acertou um alvo aproximado por um círculo. O alvo está em um cenário onde existem vários prédios. Considere os casos do disparo ser instantâneo (pistola) ou com movimento (míssil).

Referências

[1] Lengyel, E. **Mathematics for 3D Game Programming and Computer Graphics**. Charles River Media, 2002.

[2] David H. Eberly **3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics** (The Morgan Kaufmann Series in Interactive 3D Technology). Elsevier, 2nd edition, 560p, November 2006.

[3] David H. Eberly. **Game Physics** (Morgan Kaufmann Series in Interactive Technologies). Morgan Kaufmann, 1st edition, 850 p, 2004.

[4] Gino van den Bergen. **Collision Detection in Interactive 3D Environments** (The Morgan Kaufmann Series in Interactive 3D Technology). Elsevier, 1st edition, October 2003.

[5] Christer Ericson. **Real-Time Collision Detection** (The Morgan Kaufmann Series in Interactive 3-D Technology). Morgan Kaufmann, December 2004.