Região Visível a Partir de um Ponto (Anotações)

Evandro Baccarin

29 de outubro de 2025

1 Introdução

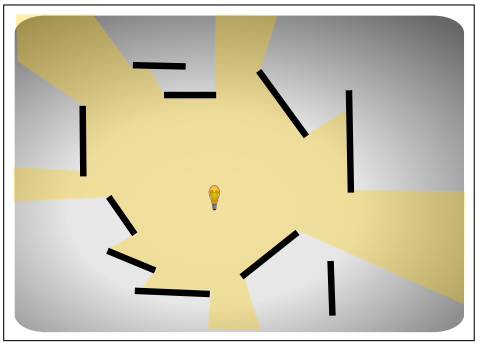

Figura 1: *Região iluminada por um ponto luminoso*

Definição do Problema: Sejam S um conjunto de segmentos de um plano que se interceptam, no máximo em seus extremos e x um ponto neste plano. Queremos determinar $V(x)$, a região de visibilidade do ponto x.

2 O Algoritmo: descrição informal

O algoritmo consiste de uma varredura planar de um raio r em torno de um ponto x (Figura 2).

Sejam $S = s_1, s_2, \ldots, s_n$ um conjunto de segmentos e um ponto x, tal que s_k $(1 \leq k \leq n)$ é determinado pelos vértice v_{kini} e v_{kfim}

Vamos ordenar os vértices v_i $(i \leq i \leq 2k)$ segundo seu ângulo em torno do ponto x (Fig 3), a partir de um vértice especial que apresentaremos mais tarde.

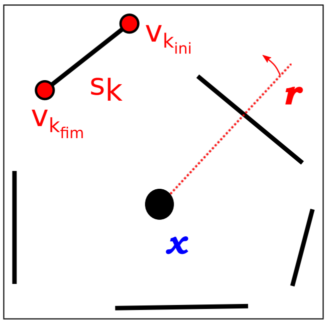

Figura 2: *Varredura planar em torno de x*

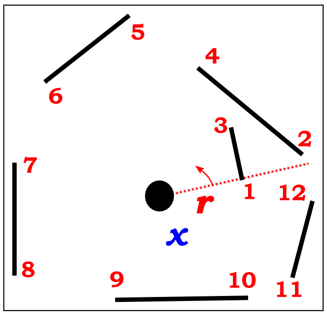

Figura 3: *Vértices ordenados*

Percorreremos estes vértices na ordem estabelecida atualizando as seguintes estruturas:

- **conjunto de segmentos ativos**: armazena o conjunto de segmentos de S cujos vértices iniciais foram visitados, mas não tiveram seus vértices finais encontrados. Estes são os vértices que correntemente podem obstruir a visão de x.
- **biombo**: um sub-segmento de algum segmento ativo que efetivamente está obscurescendo a visão do ponto x num dado instante.
- **região de visibilidade**: conjunto de segmentos que determinam a área visível a partir de x, isto é, a resposta do problema.

Quando encontramos um vértice v de **início de segmento**, temos duas situações possíveis:

1. O vértice v está "atrás" de algum segmento ativo (Fig 4).

Neste caso, apenas inserimos o segmento s_v, associado ao vértice v, no conjunto de segmentos ativos, pois, o(s) segmento(s) que o obstruia terão seus vértice finais encontrados e, possivelmente, s_v tornar-se-á o segmento "mais à frente" entre os segmentos ativos.

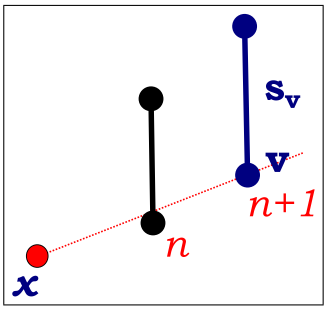

Figura 4: *Vértice de início atrás de segmento ativo*

2. o vértice v está "na frente" dos segmentos ativos (Fig. 5)

   Neste caso, o segmento s_v, que inicia a ser processado, esconde parte do plano e, assim, poderia esconder parte do segmento ativo mais próximo de x[1] (biombo).

   Sejam (i) $\overline{s}$, o segmento do biombo; (ii) $b.ini$ o vértice do início do biombo; (iii) y, a intersecção da reta determinada por x e v com o segmento $\overline{s}$.

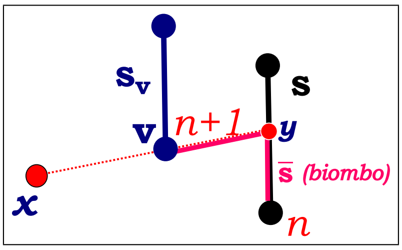

Figura 5: *Vértice de início em frente dos segmentos ativos*

   Vamos inserir os segmentos $[b.ini, y]$ e $[y, v]$ no conjunto dos segmentos de resposta, pois eles dividem o plano em uma região visível a partir de x e outra invisível.

[1] que até este momento era visível a partir de x

Como o segmento s_v esconde parte dos segmentos ativos, ele determinará um novo biombo com início em v.[2]

Por outro lado, ao encontrarmos um vértice v_f de **fim de segmento**, temos também duas possibilidades:

1. o vértice v_f está "atrás" de algum dos segmentos ativos sendo, portanto, invisível a partir do ponto x. O segmento s_v, encerrado por v_f, não influenciará mais na visibilidade do ponto x (Fig. 6). Ele é simplesmente desativado.

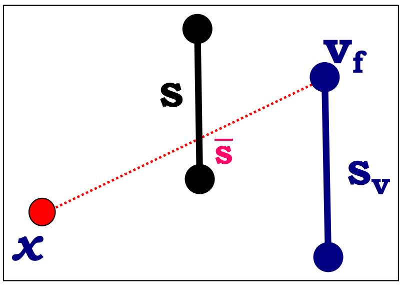

Figura 6: *Vértice de fim atrás dos segmentos ativos*

2. O vértice v_f está na "frente" dos segmentos ativos (Fig. 7).

   Neste caso, v_f é vértice final do biombo corrente. O segmento s_v, cujo extremo final é v_f, é o segmento que correntemente está obstruindo a visão a partir do ponto x.

   Devemos, então, além de desativar o segmento, inserir o segmento $[b.ini, v_f]$ e o segmento $[v_f, y]$ no conjunto da região visível. O ponto y é a intersecção da reta determinada por x e v_f e o segmento s_y imediatamente atrás de s_v (caso exista). O segmento s_y determinará o novo biombo com vértice inicial em y.

   Note que assumimos que o vértice de fim de segmento (quando visível) pertence ao mesmo segmento do biombo corrente. Isto é garantido pelo fato que os segmentos se interceptam, no máximo, em seus extremos.

Os predicados geométricos virada à esquerda e virada à direita são úteis para a resolução deste tipo de problema. A Fig. 8 ilustra o conceito. Considere a sequência ordenada de três pontos x, y, z. O ponto z pode estar à direita (virada à direita) ou à esquerda (virada à esquerda) do segmento determinado pelos pontos x e y, orientados de x para y

[2]Veremos que o início do biombo não é necessariamente algum vértice do segmento. Pode ser qualquer ponto do segmento do biombo.

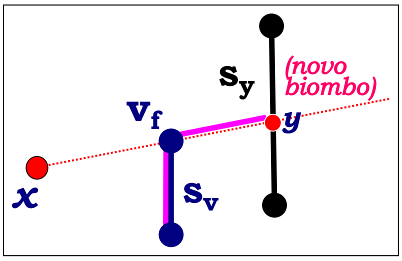

Figura 7: *Vértice de fim em frente dos segmentos ativos*

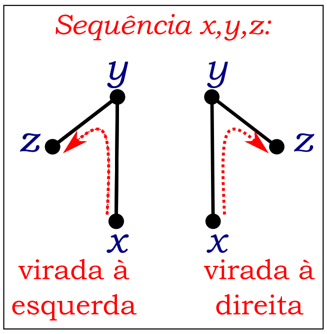

Figura 8: *Viradas à esquerda e à direita*

O cálculo destes predicados é bastante simples. Usa-se o mesmo método para calcular a área do triangulo, cujos vértices são os pontos x, y e z, dado pelo determinante abaixo. Note que se o valor do determinante for positivo, a sequência x, y, z é virada à esquerda; negativo, virada à direita; caso seja 0, os três pontos são colineares.

$$\mathbf{2 * A(T)} = \begin{vmatrix} x.x & x.y & 1 \\ y.x & y.y & 1 \\ z.x & z.y & 1 \end{vmatrix} \qquad A(T) = \begin{cases} > 0 & \text{virada à esquerda,} \\ 0 & \text{colineares,} \\ < 0 & \text{virada à direita.} \end{cases}$$

3 O Algoritmo e as Estruturas de Dados: descrição detalhada

Na seção anterior fizemos uma descrição superficial do algoritmo. Interessa-nos, agora, esmiuçar detalhes do algoritmo e das estruturas de dados a fim de atingir a cota inferior do algoritmo.[3]

Em resumo: vamos ordenar os vértices segundo seu ângulo em torno do ponto x e, conforme os percorremos nesta ordem, queremos de forma eficiente, (i) ativar/desativar o segmento a que pertencem; (ii) determinar o segmento ativo mais próximo do ponto x.

Para não excedermos a cota inferior, devemos garantir que, para cada vértice, estas operações (ativar, desativar, achar mais próximo) não excedam tempo $O(logn)$.

> Vamos utilizar uma árvore balanceada de segmentos ativos (SegsAtvs).[4] A árvore é organizada de forma que dados dois segmentos s_t e s_r, tal que o nó da árvore correspondente a s_t é antecessor ao nó de s_r: s_r está na sub-árvore direita de s_t, se $s_r.ini$ estiver à direita do segmento s_t orientado de $s_t.ini$ para $s_t.fim$. Assim, será possível ativar/desativar um segmento e encontrar segmento ativo mais próximo em tempo $O(logn)$.
>
> Considere, por exemplo, a Fig. 9. Note que foram inseridos três segmentos na sequência mostrada. Como s_t foi o primeiro a ser inserido, ele ficou localizado na raiz da árvore. A seguir, foi inserido o segmento s_r. Note que o vértice inicial de s_r está à esquerda do segmento s_t, portanto, s_r é inserido na sub-árvore esquerda de s_t. Similarmente, s_u é inserido na sub-árvore direita de s_t

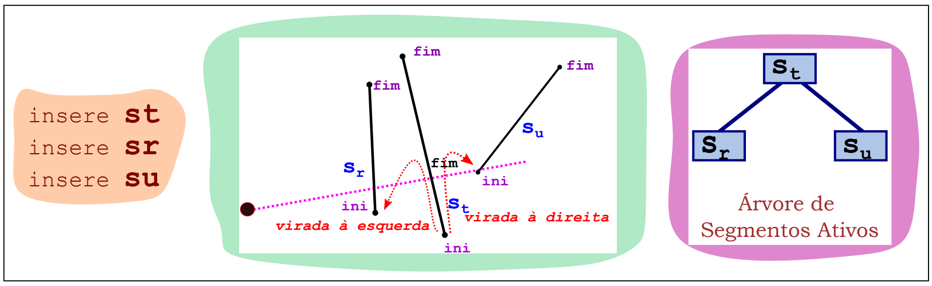

Figura 9: *Construção da Árvore de Segmentos Ativos*

A seguir, são resumidamente apresentadas as principais estruturas de dados utilizadas no algoritmo.

- **vertice(tipo,pSeg,ponto,codigo)**: Um vértice contém as seguintes informações:

[3] Vocês estudarão cotas inferiores em disciplinas futuras. Por hora, basta saber que desejamos que o algoritmo tenha complexidade $O(nlogn)$.

[4] Conteúdo para o próximo semestre.

- tipo: inicial ou final
- pSeg: apontador para o segmento a que pertence
- ponto: coordenadas (x, y) do vértice
- codigo:[5] ORIG (original), ARTF (artificial), RE (retângulo envolvente).

A ordenação dos vértices considera os seguintes critérios (em ordem decrescente de prioridade, para desempate). $v_i < v_j$:

1. se $\alpha(v_i) < \alpha(v_j)$
2. se $distancia(x, v_i) > distancia(x, v_j)$
3. se v_i é vértice de início de segmento e v_j é vértice de fim de segmento
4. **caso contrário**, $v_i = v_j$

- **segmento(pto-ini, pto-fim)**: é um segmento orienta determinado pelo ponto inicial (pto-ini) e pelo ponto final (pto-fim), tal que a sequência $x, pto - ini, pto - fim$ seja "virada à esquerda" (Fig. 8).
- **biombo**: contém o ponto inicial do biombo corrente.
- **SegsAtvs**: árvore de segmentos ativos. Operações permitidas:
  - ativaSegmento(s: segmento)
  - desativaSegmento(s: segmento)
  - segAtivoMaisProx(v: vertice): retorna o segmento ativo imediatamente atrás do vértice v

A fim de simplificar o algoritmo, vamos fazer algumas modificações nas instâncias do problema:

- Ao conjunto S de segmentos de entrada, são acrescentados quatro segmentos extras que determinam um retângulo que envolve os segmentos de S (Fig. 10).

  Isto pode ser feito em tempo linear e garante que o raio r sempre intercepte algum segmento. Vértices que, por ventura, pertençam a estes segmento são denominados vértices do retângulo.

[5] Mais à frente veremos que novos vértices podem ser criados e que inserimos um retângulo envolvendo todo o conjunto de segmentos

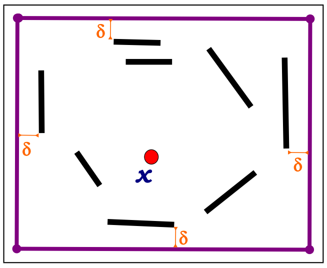

Figura 10: *Acrescentando retângulo envolvente*

- Vamos criar um vértice inicial v_0 adequado, a partir do qual serão ordenados os outros vértices (ou seja, v_0 tem ângulo 0^o).

  A partir de x é traçado um raio r (Fig. 11). Qualquer segmento s que o raio r interceptar em um ponto, digamos, y, é dividido em dois segmentos: s_i e s_f. O segmento s_i é aquele à esquerda de r (orientado de x para ∞). Os novos segmentos são inicializados da seguinte forma:

  - $s_i.ini = y; s_i.fim = s.fim$
  - $s_f.ini = s.ini; s_f.fim = y$

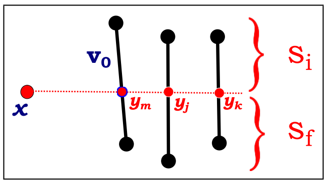

Figura 11: *Criando vértice v_0 adequado*

O vértice v_0 será o vértice inicial mais próximo de x de algum segmento s_i.

Esta adaptação é necessária, pois, como podemos notar na Fig. **??**, qualquer um dos vértices que tomássemos como vértice inicial, ou obscureceria ou seria obscurecido pelo segmento anterior, do qual ainda não teria consciência. Note que quando tal segmento fosse processado, o segmento que contém o vértice inicial já teria sido desativado. Este dois fatos levariam a execuções errôneas do algoritmo.

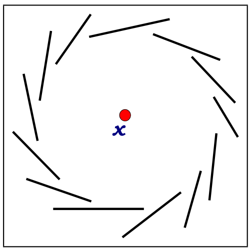

Figura 12: *Nenhum vértice óbvio para começar*

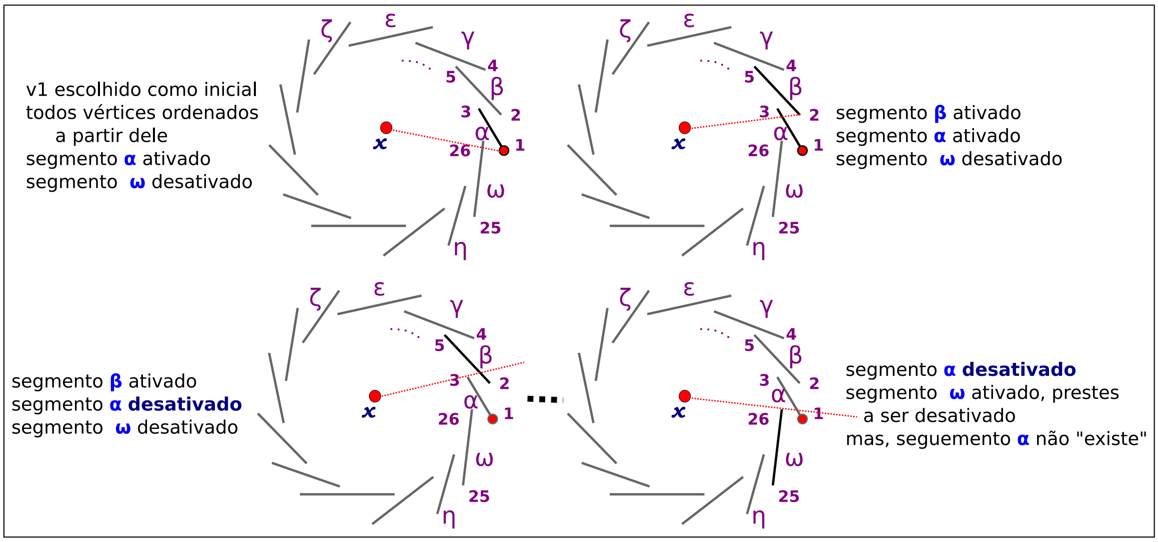

Figura 13: *Nenhum vértice óbvio para começar*

4 O Algoritmo

Algorithm 1: CalculeVisibilidade(S: segmentos; x: ponto)

```
V ← φ;
Insira os 4 segmentos do retângulo envolvente;
Crie os 2 conjuntos S_i e S_f conforme descrito;
Ative os segmentos de S_i ;
s_0 ← segAtivoMaisProx(x);
verticesLst ← ExtrairVertices(S);
Ordene verticesLst conforme o critério estabelecido;
biombo ← s_0.ini ;
foreach v ∈ verticesLst do
    if v.tipo = INICIO then
        if v ∉ S_i then
            ativaSegmento(v.pseg) ;
        if !encoberto(v) then
            r ← raio(x, v) ;
            s ← segAtivoMaisProx(v);
            y ← interseccao(r, s) ;
            insira(V, [biombo, y]);
            insira(V, [y, v]);
            biombo ← v;
    else
        desativaSegmento(v.pseg);
        if !encoberto(v) then
            r ← raio(x, v) ;
            s ← segAtivoMaisProx(v);
            y ← interseccao(r, s) ;
            insira(V, [biombo, v]);
            insira(V, [v, y]);
            biombo ← y;
    return V
end
```

Obs.: $insira(V, [a, b])$: insere o segmento com extremo inicial no ponto a e final em b no conjunto dos segmentos da resposta V.