# Anti-aliasing rápido para algoritmo de SCAN LINE

MARCELO DA CUNHA RAMOS
JOSÉ ANTONIO DOS SANTOS BORGES

Grupo de Computação Gráfica
Núcleo de Computação Eletrônica da UFRJ
CP. 2324 - CEP 20001-970 - Rio de Janeiro - RJ
marcelo@nce.ufrj.br

**Abstract.** This paper introduces a technique for fast antialiasing in scanline algorithms. Some particular cases are not solved quite well but, in general, the algorithm produces a better result than that obtained with usual implementations of non-adaptative supersampling.

## 1. Descrição do problema

A apresentação de imagens em telas é essencialmente um processo de amostragem realizado em cada pixel. O tamanho dos pixels determina o limite superior de frequencias que podem ser mostradas. Alguns efeitos ocorrem devido ao uso de frequências maiores do que o limite de Nyquist, ou seja, um ciclo a cada dois pixels [Cook, 1989]:

a) arestas serrilhadas em cenas estáticas
b) arestas em "efeito escada rolante" em animações
c) padrões de interferência (Moiré) em texturas coerentes
d) efeitos de coerência em áreas de gradiente
e) figuras pequenas que aparecem e desaparecem em animações

O efeito mais crítico aparece em animações, em que os artefatos provenientes de aliasing chamam muito a atenção do observador.

As soluções de filtragem, como definidas em teoria da amostragem, produzem resultado relativo quando aplicadas a computação gráfica, pois não existe limite nas altas frequências que podem ser geradas. Além disso, existe uma dificuldade de realizar filtragem sem alterar muito a estrutura dos algoritmos tradicionais, e a implementação tende a ser complicada.

Desta forma, o processo de anti-aliasing em computação gráfica é usualmente uma aproximação. Os resultados muitas vezes não precisam ser exatos e, especialmente na geração de imagens sintéticas, o que se busca é apenas esconder efeitos grosseiros de aliasing.

## 2. Técnicas tradicionais de anti-aliasing

Existem essencialmente três categorias de algoritmos de anti-aliasing:

a) algoritmos de super-amostragem

Nesse método, a imagem é gerada em resolução maior que a imagem real, e é realizada uma filtragem sobre os pontos gerados [Lobb, 87]. Uma variante deste método é a super-amostragem adaptativa [Glassner, 89], que tenta só super-amostrar regiões em que ocorre o aliasing e não todos os pontos indiscriminadamente.

A implementação desse método é muito fácil, entretanto existe um aumento excessivo de tempo de processamento e (dependendo da implementação) de gasto de memória devido à quantidade muito maior de pontos a amostrar.

b) algoritmos de geometria de subpixel

Estes algoritmos tentam descobrir as áreas dos elementos geométricos (quase sempre polígonos) que interferem na pintura do pixel [Rogers, 85]. A cor do pixel é obtida como a média ponderada das cores dos polígonos intervenientes e suas áreas visíveis no pixel.

Os algoritmos deste tipo não podem ser implementados sem alterar bastante a estrutura dos algoritmos convencionais de rendering.

c) Amostragem estocástica

A idéia aqui é distribuir "randomicamente" pontos de amostragem [Cook, 89]. Como consequência deste processo, parte dos artefatos de aliasing são substituídos por ruído, que é bem assimilado pelo mecanismo de visão.

A vantagem deste processo é que são necessárias menos amostras por pixel. Entretanto, esse método é em geral somente indicado para ray-tracing.

## 3. Uma idéia geral do método

O método proposto foi inspirado no método apresentado em [Wyvill, 89], criado para processamento em ray-tracing, e aqui modificado para processamento de varredura (scan-line). A idéia geral do algoritmo é gerar uma super-amostragem adaptativa, de acordo com a variação de cor na imagem. O algoritmo assume, portanto, que regiões sem mudanças bruscas de cor não precisarão de anti-aliasing.

O algoritmo original propõe o lançamento de um raio em cada um dos cantos de cada pixel (e não no centro do pixel, como é usual). Caso não haja variação significativa de cor nos cantos, o pixel é pintado com a média das cores. Caso contrário, serão feitas mais amostras segundo uma busca binária ao longo das bordas unindo os cantos adjacentes com diferença de cor, para detectar o(s) lugar(es) onde ocorre(m) variação de cor, ou seja, detetar o lugar onde passam arestas, como mostrado na figura 1. Uma vez detectados estes pontos, o algoritmo estima a área de cada polígono real, a partir de polígonos gerados pela união destes pontos ao centróide (média dos pontos achados), conforme visto na figura 2.

Esta algoritmo calcula bem os casos mostrados na figura 3, que são os mais críticos na geração de animações. É interessante notar que no caso de uma única aresta, o resultado é exato, uma vez que o centróide passa exatamente em cima da aresta. O método, neste caso, é muito melhor que super-amostragem simples.

Este método não funciona bem em imagens contendo polígonos com largura da projeção na tela menor que 1 pixel, pois a super-amostragem é apenas realizada quando há variação de cor nos cantos. Isso pode ser crítico ou não, dependendo da cena analisada. Há possibilidade de forçar uma super-amostragem a partir do conhecimento prévio dos polígonos que tenham essa característica, no entanto isso, no caso de ray-tracing, gera um aumento muito grande no tempo de processamento.

## 4. O algoritmo original de scanline

A implementação do algoritmo de scan line z-buffer no sistema ANIMAL [Ramos, 93] é conceitualmente semelhante à descrita em [Rogers, 85]. Na varredura são utilizadas as seguintes estrutura de dados

a) buckets de polígonos: separando os polígonos por ymax

b) lista de polígonos ativos: atualizada a cada linha de varredura com polígonos provenientes do bucket de polígonos referente a esta linha.

c) listas de pares de arestas ativas: define todas as arestas que cruzam uma linha de varredura. As coordenadas registradas nesta lista tem que ser atualizadas a cada mudança de linha. Essa lista é mantida ordenada em x.

Mostramos a seguir um resumo do algoritmo:

```
inicializa tela
para cada linha y de rastreio
   inicializa o z-buffer da linha com zmax
   inicializa o buffer da linha com cor de fundo
   atualiza as listas descritas acima
   para cada coluna x da linha
      toma todos os pares de arestas que passam
                        por este ponto
      calcula coordenadas z no centro do pixel
      pinta a cor do poligono com menor coordenada
```

## 5. O algoritmo de scan line z-buffer com antialiasing

Para implementar antialiasing, temos que fazer diversas mudanças no algoritmo:

a) passar a amostrar nos cantos dos pixels

b) manter um histórico de processamento da linha anterior.

Para cada pixel é mantida uma lista contendo os pares de arestas que passam por este pixel (cada par pertencente a um polígono). Isto é necessário para que seja possível posteriormente fazer a busca binária nas bordas do pixel, para identificar o ponto em que ocorre a mudança de cor (ou seja, o lugar onde uma aresta termina).

c) verificar se é necessário super-amostrar

Após o cálculo da cor no canto de um pixel (exceto para a primeira linha e primeiras colunas), verifica-se a variação de cor deste ponto e dos outros três cantos do pixel, já calculados (2 na linha anterior e 1 nesta). Caso haja diferença significativa entre dois cantos adjacentes, realizar a super-amostragem e integrar o novo valor.

d) Super-amostrar

Este processo visa descobrir, sobre cada borda do pixel, o lugar aproximado onde ocorre mudança de cor. A idéia é realizar uma busca binária sobre cada canto do pixel descobrindo o lugar onde ocorreu uma troca de cor. Limitou-se nessa implementação, por simplicidade, a detetar apenas uma única troca de cor.

A escolha da aresta do polígono interveniente no ponto médio em cada amostragem é feita de maneira análoga ao algorítmo de scan-line z-buffer tradicional: basta verificar antes do cálculo da coordenada z de cada polígono se o ponto (x,y) amostrado realmente faz parte da projeção do polígono. Como só se faz esta verificação para os polígonos da lista, o acréscimo de tempo gerado não é significativo.

d) estimar a integração da área dos pedaços de polígonos dentro do pixel

A integração é feita tomando-se os pontos de variação de cor detetados na busca binária, calculando o centróide destes pontos e integrando as regiões através dos triângulos unindo os cantos do pixel e este centróide, como ilustrado na figura 2.

## 6. Casos particulares

Esse algoritmo apresenta alguns casos difíceis, como mostrado na figura 4. Esses casos, porém trazem um impacto quase nulo em cenas reais (podem ser até mesmo ignorados numa implementação simples). Além disso, na gravação em vídeo, algumas dessas situações são muito atenuadas pela degradação natural que ocorre no processo de gravação.

a) Polígonos menores que um pixel e totalmente interiores a ele

Mesmo que o pixel seja sobre-amostrado, o algoritmo não tomará conhecimento destes polígonos. Uma forma de solucionar este problema é um preprocessamento que crie um polígono extra, a partir da área projetada deste, e que esteja alinhado com a borda do pixel.

b) Polígonos que tem vértices dentro do pixel, mas todas as arestas estão localizadas sobre a mesma borda do pixel

Da mesma forma, o algoritmo ignorará o polígono. A solução para este caso envolve um recorte do polígono em relação ao pixel, o que é totalmente inviável, por causar um aumento excessivo de processamento. Nesse caso, entretanto, as distorções geradas não são percebidas facilmente.

c) Mais de uma aresta de polígonos diferentes, ocorrendo em uma borda de pixel

O processo de busca binária detectará apenas uma variação por borda, e um dos polígonos será ignorado.

Essa situação, que somente ocorre quando um vértice de um objeto pertence ao pixel, em geral traz pouca influência à cor calculada.

d) um polígono muito fino cortando verticalmente ou horizontalmente um pixel

O problema neste caso é mais complexo do que o item c, pois este polígono pode estar projetado no contorno do objeto. Neste caso o algorítmo pode falhar gerando algumas altas frequências no vídeo, embora em uma região muito pequena. A solução para este problema é um pre-processamento, alinhando o polígono a uma das bordas do pixel.

e) Variação de highlights menores que um pixel

Essa situação ocorre quando um highlight coincide com a silhueta do objeto. Neste caso, a variação de cor é detectada, mas o polígono não pertencerá a nenhuma das tabelas. A implementação atual simplesmente descarta esta situação. Essa situação acontece na silhueta de objetos curvos poligonalizados com malha muito fina.

f) texturas muito finas

Esta situação recai nos casos de a) a e) citados e é muito difícil de solucionar. Neste caso, entretanto, optamos por fazer uma pré-filtragem na textura, usando o método de geração de cópias da texturas pré-filtradas com tamanho decrescentemente divisivo por 2, citado por [Watt, 93], e assim virtualmente eliminar esse problema.

## 7. Resultados

A implementação deste algoritmo permitiu a criação de imagens de boa qualidade sem um grande acréscimo de tempo de processamento. Isto é muito importante no caso de gerações de imagens para animações que necessitam uma grande quantidade de quadros.

Embora não fizéssemos uma medida precisa e sistemática, realizamos alguns testes gerando diversos tipos de cenas. Executamos o algoritmo removendo o algoritmo de anti-aliasing e chegamos a um decréscimo de tempo inferior a 30% na média.

O algoritmo mostrado foi implementado em C numa estação Sun SparcStation II, como parte do sistema Animal II, destinado à geração de animações em 3 dimensões, produzido no Núcleo de Computação Eletrônica da UFRJ e distribuído sob a forma de domínio público via Internet.

**Bibliografia:**

[Cook, 89] - Cook, R. - *"Stochastic Sampling and Distributed Ray Tracing"*, em An introduction to Ray Tracing", Academic Press, 1989 - p.161-199

[Glassner, 89] - Glassner, A., *"An overview of Ray Tracing"*, em An introduction to Ray Tracing", Academic Press, 1989, p. 1-31

[Lobb, 87] - *"Antialiasing of polygons with a Weighted Filter"*, Computer Graphics, 1987. p 109-127

[Ramos, 93] - "Animal II - Um sistema para produção de animações tridimensionais", projeto de fim de Curso de Informática, UFRJ, 1993

[Rogers, 85] - Procedural Elements for Computer Graphics, McGraw Hill, 1985

[Watt, 93] - Notas de aula do curso "Introduction to 3d Computer Graphics", IMPA, 1993

[Wyvill, 89] - Wyvill, G and P. Sharp - *"Fast Antialiasing of Ray Tracing Images"*, em New Advances in Computer Graphics, Springer Verlag, R.A.Earnshaw and B.Wyvill, eds. - p.579-587

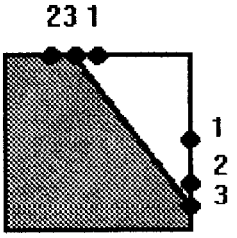


*figura 1 - Busca binária de interseção de aresta*

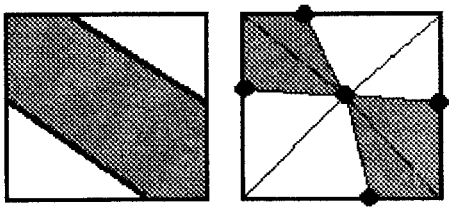

*figura 2 - Cálculo aproximado das áreas*

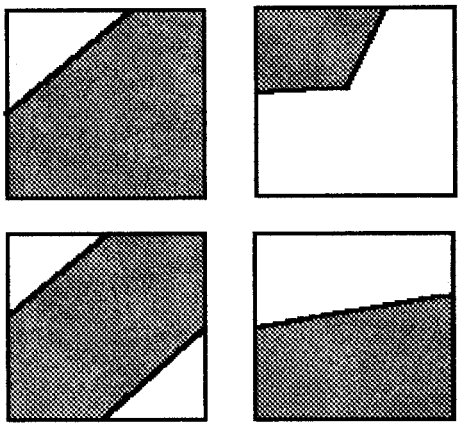

*figura 3 - Casos favoráveis*

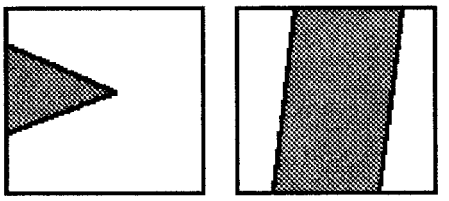

*figura 4 - Casos desfavoráveis*

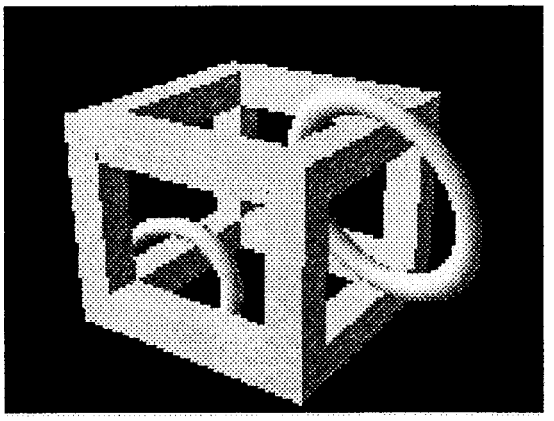

*figura 5 - Uma imagem sem anti-aliasing*

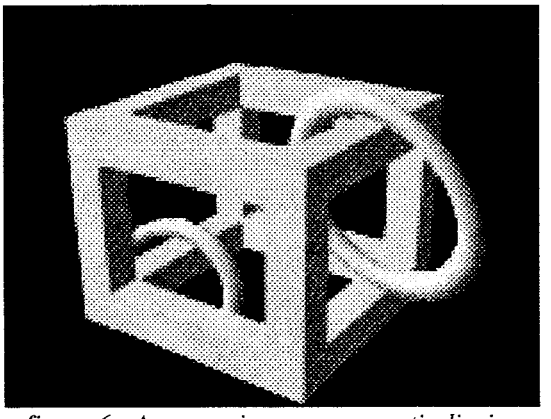

*figura 6 - A mesma imagem com anti-aliasing*