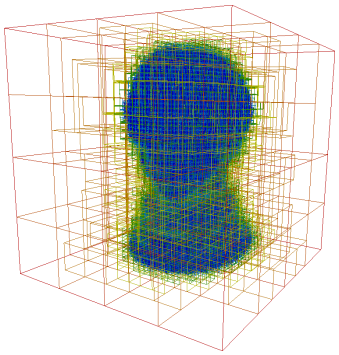
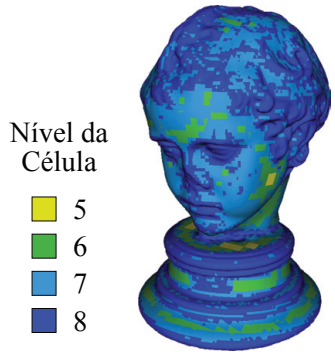
4
Visualização de ADFs em GPU

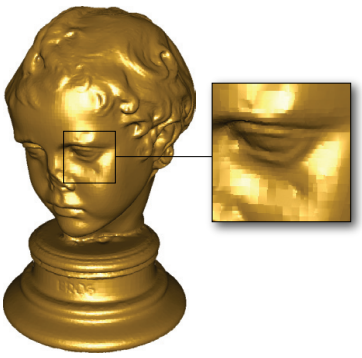
Este capítulo baseia-se no arcabouço definido no Capítulo 3 para propor um programa de fragmentos (*fragment program*) capaz de renderizar as iso-superfícies de uma ADF diretamente em GPU. A técnica proposta é um lançador de raios de propósito geral, que utiliza *traçado por esferas* (*sphere tracing*) para percorrer o campo de distância. Além da renderização em GPU, este capítulo também aborda uma solução para as descontinuidades de sombreamento das ADFs. A Figura 4.1 ilustra alguns resultados.



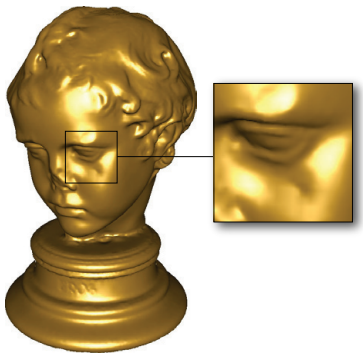
4.1(a): *Octree* da ADF, com as células coloridas de acordo com o nível.



4.1(b): Superfície colorida de acordo com o nível da célula usada na reconstrução.



4.1(c): Modelo renderizado com normais calculadas pelo método tradicional das ADFs.



4.1(d): Modelo renderizado com normais suaves pelo método proposto nesta dissertação.

Figura 4.1: Imagens renderizadas em GPU a partir de uma ADF de nível 8 do modelo "Eros". O modelo original é uma malha de triângulos e foi obtido do repositório Aim@Shape.

4.1
Traçado por Esferas

O traçado por esferas [13] é um método iterativo para o percorrimento de campos de distância. Neste método, os valores do campo são utilizados para marchar ao longo de um raio em passos que garantidamente não penetrem uma iso-superfície. Considera-se que um raio intercepta uma iso-superfície quando a distância entre os dois encontra-se dentro de uma tolerância ϵ pré-definida. Caso não intercepte a iso-superfície, um raio morre quando ele sai do espaço da ADF – definido por uma AABB (*axis-aligned bounding box*).

Para utilizar traçado por esferas, é imprescindível que se tenha controle sobre o erro de reconstrução do campo de distância, pois erros muito grandes podem fazer com que o raio "pule" e perca uma iso-superfície. Neste caso, os campos de distância com sinal são capazes de tolerar mais erros, pois um raio pode utilizar o sinal para perceber que penetrou um objeto e "voltar para trás". No entanto, erros grandes ainda podem fazer com que o raio pule superfícies muito finas ou detalhes muito pequenos de um objeto.

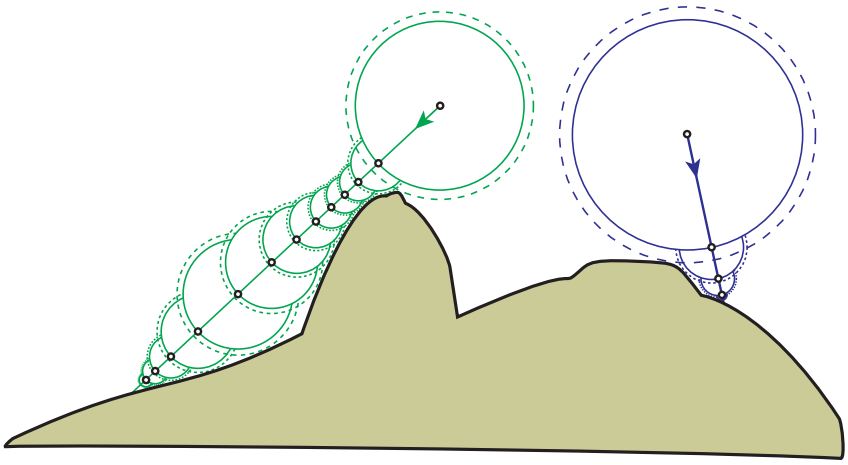
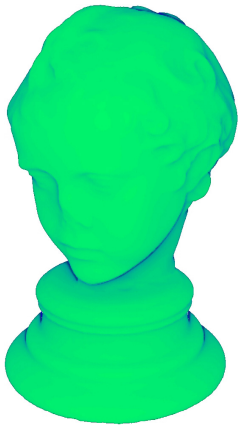


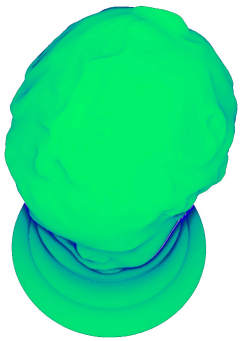
Figura 4.2: No traçado por esferas, os raios que "tangenciam" a superfície precisam de mais iterações para convergir.

A convergência do traçado por esferas é geralmente rápida, exceto para os raios que "tangenciam" a iso-superfície antes de atingi-la, como ilustrado na Figura 4.2. Este comportamento pode ser observado claramente na Figura 4.3, onde as cores indicam o número de iterações necessárias para cada raio da imagem, e as diferenças no número de iterações realçam naturalmente as arestas do modelo. Note que as imagens foram geradas utilizando apenas uma função de cor linear, sem qualquer sombreamento.

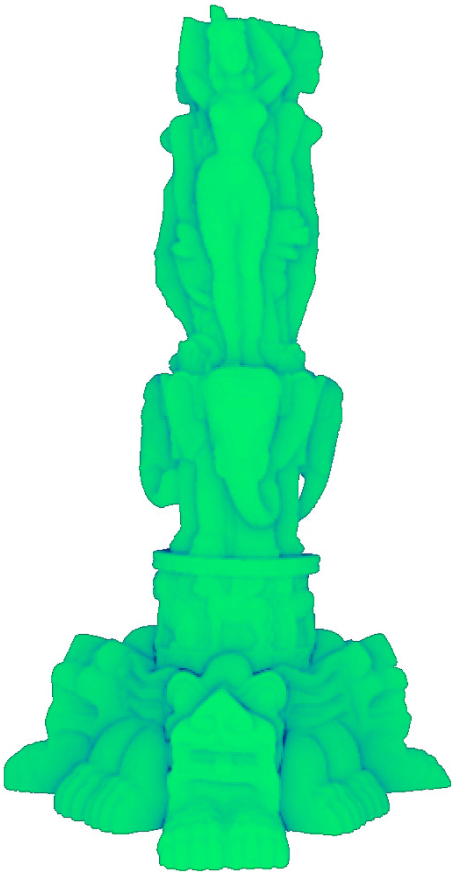
Certos modelos e ângulos de visão podem ser especialmente problemáticos para o traçado por esferas. Em casos extremos, alguns raios podem estourar o limite máximo de iterações (estabelecido pelo algoritmo ou pelo



4.3(a): Vista lateral da estátua de Eros.



4.3(b): Estátua de Eros vista de cima.



4.3(c): Vista frontal da "Estátua Tailandesa" (*Thai Statue*) de Stanford.

Figura 4.3: Imagens coloridas de acordo com o número de iterações necessárias para que os raios interceptem as superfícies. As cores variam do verde claro (1 iteração) até o azul escuro (64 iterações).

hardware) e alguns *pixels* da imagem podem ficar vazios. Em geral, porém, o método é bastante eficiente e atrativo como um lançador de raios de propósito geral. Como demonstrado no Capítulo 5, alguns tipos de aplicações (que manipulam o campo de distância diretamente) não funcionam com o método de lançamento de raios tradicional das ADFs (Seção 2.2.5); nestes casos, o traçado por esferas é provavelmente a melhor opção.

4.2 Traçado por Esferas para ADFs Globais

O método de traçado por esferas funciona sob a condição de que a tolerância ϵ seja maior que o erro de reconstrução do campo de distância. Em geral, isto é fácil de garantir para as *ADFs globais* (Seção 2.2.2), já que elas são construídas de modo que o erro de reconstrução não ultrapasse um certo limite pré-determinado.

O algoritmo de traçado por esferas para ADFs globais, tal como utilizado

neste trabalho, encontra-se a seguir em pseudo-código. Um raio é definido por uma origem $\mathbf{r}_0$ e uma direção $\mathbf{r}_d$. A variável escalar c é o iso-valor da superfície que deve ser desenhada, enquanto que ϵ indica a tolerância usada nos testes de interseção.

Algoritmo 4.1 Traçado por esferas para ADFs globais.

```
function CAST_RAY(r_0, r_d, c, ε)
    (t_near, t_far) ← INTERSECT_AABB(r_0, r_d)
    t ← t_near                          ▷ projeta o raio até a face da AABB da ADF
    ℓ ← 1
    p ← r_0 + t r_d
    while t < t_far do
        (v, ℓ) ← QUERY(ℓ, p)
        d ← GET_DISTANCE(v, p) − c
        if d < ε then                   ▷ raio atingiu a iso-superfície?
            n ← GET_NORMAL(v, p)
            return SHADE(p, n, −r_d)
        end if
        t ← t + d                       ▷ caminha ao longo do raio
        p ← r_0 + t r_d
    end while
    discard                             ▷ este raio não intercepta a iso-superfície
end function
```

Observe que esta função explora coerência espacial ao longo do raio, pois as consultas ao campo de distância — feitas através da função $\textsc{query}(\ell, \mathbf{p})$ — sempre retomam a partir do nível onde a consulta anterior parou. Uma vez que o raio tenha atingido a iso-superfície, uma normal é reconstruída e a superfície é iluminada com sombreamento de Phong (*Phong shading*). Este algoritmo é utilizado em um programa de fragmentos que é acionado ao se desenhar a AABB de uma ADF. A implementação real é feita em GLSL e encontra-se disponível nos apêndices.

4.3 Traçado por Esferas para ADFs de Fronteira

Ao contrário das ADFs globais, as ADFs de fronteira não oferecem limites para o erro de reconstrução nos pontos localizados fora das células que interceptam a iso-superfície de interesse. Pontos distantes da iso-superfície, especialmente aqueles próximos das faces da AABB da ADF, contêm muito erro. Portanto, o traçado por esferas é sempre exposto a grandes erros nas primeiras iterações.

Esta dissertação propõe uma alteração simples para as ADFs de fronteira, a fim de oferecer limites para os erros de reconstrução em todos os pontos do domínio e, ao mesmo tempo, reter a maior parte da eficiência dessas ADFs.

Durante a construção de uma ADF de fronteira, uma célula que não intercepte a iso-superfície de interesse só deve ser descartada caso o erro de reconstrução estimado para o seu pai seja menor que uma *porcentagem* da menor amostra de distância armazenada na célula. Com esta alteração, garante-se que o erro de reconstrução em um ponto é sempre proporcional ao valor do campo de distância.

Para distinguir as ADFs de fronteira originais daquelas com a alteração proposta acima, estas últimas serão chamadas deste ponto em diante de *ADFs superficiais*. As ADFs superficiais utilizadas nesta dissertação toleram erros de até 10% do valor do campo. Em média, isto traz um aumento de 25% no número de células de uma ADF, o que no total é apenas uma fração do número de células das ADFs globais.

O traçado por esferas funciona bem para ADFs superficiais, contanto que as distâncias usadas para caminhar ao longo de um raio sejam reduzidas a uma *margem de segurança*. Por exemplo, para lidar com erros de até 10%, é necessário multiplicar todas as distâncias reconstruídas por 0.9. Esta margem de segurança encontra-se ilustrada na Figura 4.2: as distâncias reconstruídas são representadas pelos círculos tracejados, enquanto que as distâncias usadas para caminhar ao longo do raio são representadas por círculos contínuos. Com base nesta solução, a renderização de ADFs superficiais mostrou-se quase 15% mais lenta que a renderização (com erros) de ADFs de fronteira. Porém, no caso das ADFs superficiais, não é possível notar nenhuma diferença de imagem em comparação com as ADFs globais.

4.4 Descontinuidades

As ADFs possuem dois tipos de descontinuidades (Seção 2.2.3) que podem deteriorar a qualidade das imagens renderizadas:

1. Descontinuidades C^1 (nas normais) nas fronteiras das células, devido ao uso de interpolação trilinear;

2. Descontinuidades C^0 (geométrica) nas faces e arestas onde células-folha de diferentes níveis se encontram.

Ambas as descontinuidades causam artefatos de sombreamento (Figura 4.1(c)), enquanto que as descontinuidades C^0 causam pequenas rupturas na superfície, visíveis com *zoom*.

4.4.1 Tratamento para Rupturas

No traçado por esferas, os pontos de interseção não são exatos. Tudo que se sabe é que o ponto está, no máximo, a uma distância ϵ da superfície. Um efeito colateral desta tolerância é que a superfície adquire uma certa espessura. Assim, se ϵ for maior que o diâmetro da ruptura, ela será "coberta".

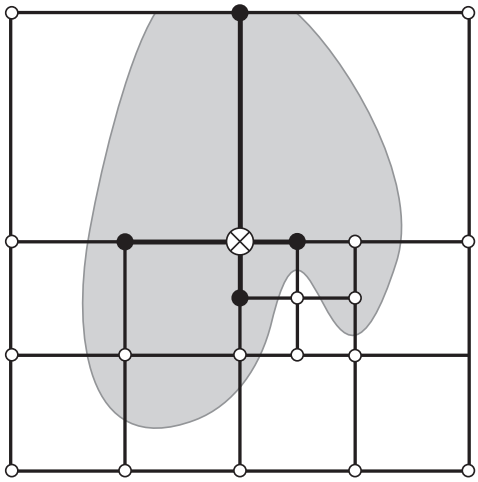
Este efeito soluciona a maior parte das rupturas das ADFs globais: basta escolher um ϵ aceitável. Embora nenhum problema tenha sido observado, no caso das ADFs de fronteira os erros podem ser bem maiores, de forma que não haja valor de ϵ plausível que solucione todas as rupturas. Nesses casos, uma solução possível seria subdividir as maiores folhas da ADF, que causam as maiores descontinuidades, até que as rupturas tornem-se menores que ϵ.

4.4.2 Solução para as Descontinuidades de Sombreamento

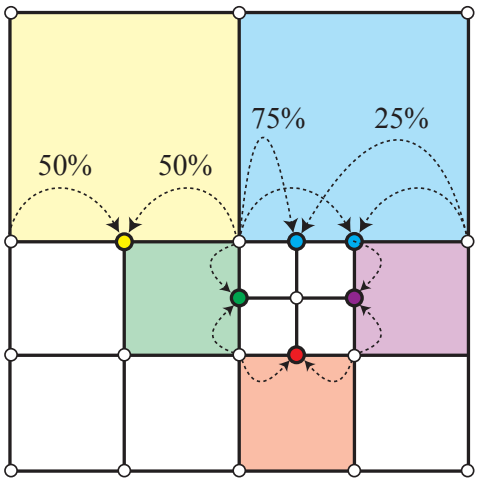
Para renderizar iso-superfícies com sombreamento suave, uma solução eficiente encontrada consiste em calcular e armazenar *normais suaves* em uma textura de *voxels* adicional, denominada "textura de normais". Neste esquema, a normal de um ponto pode ser reconstruída de forma simples, por interpolação trilinear com as normais de uma célula. Cada normal é codificada em um *texel* de 24 bits, o que implica em um aumento de 75% no custo de memória por *voxel*. Por outro lado, a reconstrução de normais deste modo é mais rápida que o cálculo de gradientes por diferenças finitas.

A construção da textura de normais acontece em uma fase de pré-processamento. Em um primeiro passo, as normais são estimadas para cada *voxel* através de diferenças centrais, com base nas amostras de distância dos *voxels* vizinhos mais próximos, como ilustrado na Figura 4.4(a).

Em um segundo passo, as normais são re-estimadas para que não ocorram descontinuidades nas fronteiras entre células de tamanhos diferentes. Esta re-estimativa é feita percorrendo-se as células da *octree* em largura, de cima para baixo. As normais nos *voxels* posicionados entre arestas ou sobre faces de uma célula-folha, como ilustrado na Figura 4.4(b), devem ser re-calculadas por interpolação linear (ou bilinear, para faces) com as normais da célula-folha.



4.4(a): A normal do *voxel* ⊗ é calculada por diferenças centrais utilizando os *voxels* marcados com círculos pretos.



4.4(b): As normais nos pontos indicados são re-estimadas por interpolação linear com as normais da célula sobrejacente.

Figura 4.4: Construção da textura de normais.