5 Extração de Modelos

O objetivo deste capítulo é descrever os algoritmos de extração de malhas a partir da hierarquia de células hexaédricas descrita no Capítulo 4. Considerando a regra de extração básica, o objetivo é criar algoritmos que extraiam, em tempo de exibição, malhas em diferentes resoluções, dependentes: (a) do erro da aproximação (*error-dependent*), (b) da posição da câmera (*view-dependent*) e (c) do número de células da aproximação (*polygon-budget*).

5.1 Extração Dependente do Erro Geométrico

A extração **dependente do erro geométrico** objetiva achar $\boldsymbol{m} \in \mu$', de menor complexidade, cuja diferença para o modelo original seja menor do que (ou igual a) um erro $\varepsilon \in \Re^+$ pré-definido. Neste tipo de extração, o erro ε é o único parâmetro do contexto de visualização, ou seja, o contexto é definido diretamente no espaço do objeto, não importando, por exemplo, quais são os parâmetros de câmera.

O algoritmo de extração consiste em percorrer a árvore de cima para baixo e, a cada nó da hierarquia, verificar se o erro a ele atribuído é menor do que ε (ou igual a ele). Se for menor ou igual, a célula associada ao nó é incluída na lista de células visíveis; caso contrário, o algoritmo prossegue recursivamente com os filhos do nó. Caso o nó

seja uma folha, ele será necessariamente incluído na lista, pois possui erro zero. No final, a lista de células visíveis corresponde ao corte na árvore que representa um modelo ***m***. Este algoritmo pode ser ilustrado pelo pseudo-código a seguir:

```
struct estr_célula;
typedef estr_célula tipo_célula;

struct estr_nó_hier {
                    est_nó* esquerdo; // filho esquerdo
                    est_nó* direito;      // filho direito
                    float erro;
                    tipo_célula* célula;
                    tipo_aabb* aabb; //caixa envolvente da região
};
define esrt_nó_hier  tipo_nó;

void Extração_dependente_do_erro (float ε, tipo_nó* nó) {
        if ((nó→ÉFolha()) || (nó→erro ≤ ε))
                Seleciona  (nó.célula);
        else
        {       Extração_dependente_do_erro (ε, nó→esquerdo);
                Extração_dependente_do_erro (ε, nó→direito );
        }
}
```

O procedimento *Extração_dependente_do_erro* deve receber a raiz da hierarquia no início da recursão.

Devido à propriedade de ordenação parcial da estrutura de MR (P.4.2), pode-se provar que ***m***, restrito ao espaço dos modelos μ', é ótimo. A prova é baseada em dois fatos contraditórios:

1. a expressão condicional efetuada pelo algoritmo garante que todos os nós acima do corte gerado possuem erro maior do que ε;
2. se um corte não é o de menor complexidade, existe pelo menos um par de nós irmãos que pode ser substituído pelo seu nó pai sem que o erro seja violado, ou seja, o erro associado ao pai é menor do que ε.

Consideremos o corte M_i gerado pelo algoritmo de extração parametrizado pelo erro ε. S é o conjunto formado por todos os nós acima de M_i. Do fato 1, sabe-se que todo nó p pertencente a S possui necessariamente seu erro maior do que ε. Ou seja:

$$\forall p \in S \rightarrow p.erro > \varepsilon.$$

Supondo que M_i não gere o modelo de menor complexidade, a partir do fato 2, espera-se que exista p' pertencente a S tal que o erro associado a p' seja menor do que (ou igual a) ε. Ou seja:

$$p' \in S \ \wedge \ p'.erro \leq \ \varepsilon.$$

Isso mostra que a suposição é inválida e que, portanto, se M_i é gerado pelo algoritmo de extração, então M_i é o modelo de menor complexidade que respeita o erro ε.

5.2 Extração Dependente da Câmera

A extração **dependente da câmera** é própria para aproveitar as distorções causadas por uma transformação projetiva, que valoriza mais as regiões próximas à câmera em relação às regiões distantes (Figura 5.1).

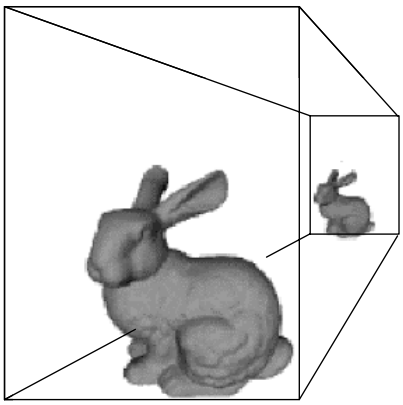

Figura 5.1 - Deformação da projeção perspectiva.

De forma geral, as regiões mais próximas necessitam ser desenhadas numa resolução mais alta do que as regiões mais distantes. Por outro lado, durante uma animação, à medida que a câmera se aproxima de uma região, causando o aumento da sua resolução, outras regiões passam a ficar fora de seu volume de visão, podendo ser desconsideradas para a visualização. Em alguns casos, os dois efeitos tendem a compensar um ao outro no sentido de que, idealmente, o número de polígonos desenhados mantém-se num valor médio constante.

São necessários dois tipos de testes para os algoritmos de busca dependentes da câmera: um para descartar os objetos que estão fora do volume de visão e outro para decidir qual a variação da resolução ao longo do domínio.

5.2.1 Descarte

O descarte é responsável pela eliminação de regiões que estão fora do volume de visão. Para realizar o descarte de forma eficiente, é necessário descartar vários polígonos ao mesmo tempo. Estruturas hierárquicas que formam conjuntos disjuntos de *clusters* são reconhecidamente bem utilizadas na realização desta tarefa [Gar98, DeRose98].

Uma vez que a estrutura de MR utilizada é uma hierarquia, a propriedade **P.4.1** garante que essa estrutura pode ser vista como uma estrutura de *clusters* disjuntos de células, na qual cada colapso (ou nó) representa a união entre dois *clusters*. Se associarmos a cada nó da hierarquia de células o volume envolvente de seu *cluster*, temos uma hierarquia de volumes envolventes que pode ser usada para realizar o descarte.

Os cálculos do volume envolvente de cada *cluster* devem ser feitos juntamente com a construção da hierarquia. Sendo assim, eles influenciam o tempo total gasto em pré-processamento, o qual, como dito anteriormente, objetivamos reduzir. Para realizá-los eficientemente, o volume envolvente utilizado neste trabalho foi a caixa alinhada com os eixos, *aabb* (*aligned-axis bounding box*), pois podemos achar as caixas envolventes mínimas em tempo linear no número de nós da hierarquia.

Além de permitir uma rápida construção, a utilização da *aabb* possui um teste de pertinência (*inside-outside test* [Tom99]) eficiente e invariável a uma possível deformação em $\boldsymbol{Z}$ na geometria do reservatório prevista pelas **características de visualização** do problema, ou seja, uma *aabb* continua sendo uma *aabb* após uma deformação em $\boldsymbol{Z}$.

5.2.2 Cálculo do Erro Projetado

Para decidir a resolução adequada de uma região do reservatório adaptando-a em relação à câmera, foi utilizado um cálculo baseado na projeção, no plano de imagem, do erro geométrico local existente na região (Figura 5.2).

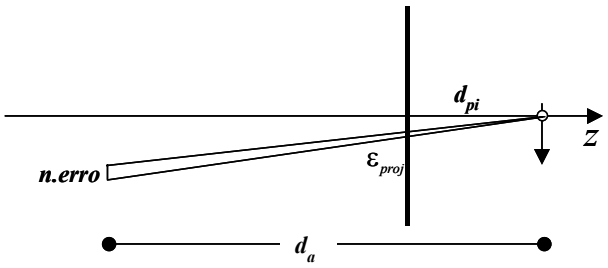

Figura 5.2 – Redução do erro em função da profundidade.

Dado um nó ***n*** da hierarquia e a célula ***c*** associada a ele, o erro projetado relativo a ***dom***(***c***) é dado pela equação:

$$\varepsilon_{proj} = n.erro \times (d_{pi}/d_a),$$

onde d_{pi} é a distância do centro do plano da imagem à câmera e d_a é a menor distância da caixa envolvente da célula à câmera.

O cálculo projetado é particularmente bom porque reproduz o comportamento da projeção cônica utilizada pela câmera (considerando uma câmera em perspectiva). Além disso, ele pode ser facilmente estendido para levar em consideração a resolução da imagem gerada. Se a imagem possui dimensões ***lx*** e ***ly***, no sistema global, e dimensões ***w*** e ***h***, no sistema de janela, então o erro projetado pode ser mapeado para o erro em *pixels* através da equação:

$$\varepsilon_{proj} = (n.erro \times (d_{pi}/d_a)) \times (max(w, h)/max(lx, ly). \qquad \textbf{(5.1)}$$

5.2.3 Algoritmo de Extração

O algoritmo de extração dependente da câmera é semelhante ao algoritmo de extração dependente do erro geométrico. A única diferença é a substituição do erro geométrico pelo erro projetado e o teste de descarte. Assim, o pseudo-código do algoritmo é:

```
estrutura estr_parâmetros_de_visualização {
                    float x, y;
                    float lx, ly;
                    float d_pi, d_a;
                    float ε;
};
define estr_parâmetros_de_visualização tipo_params_vis;

float Projeta (tipo_params_vis v, float erro) {
                    retorna (erro/(v.d_pi - v.d_a)) × (max(v.x, v.y)/ max(v.lx, v.ly));
}

void Extração_dependente_da_câmera (tipo_params_cam v, tipo_nó* nó) {
          if Descarta(nó→aabb)
                    return;
          else if ((nó→Éfolha()) || (Projeta(nó→erro) ≤ v.ε))
                    Seleciona  (nó.célula);
          else {
                    Extração_dependente_da_câmera (v, nó→esquerdo);
                    Extração_dependente_da_câmera (v, nó→direito);
          }
}
```

O procedimento *Extração_dependente_da_câmera* também deve receber a raiz da hierarquia no início da recursão. O procedimento *Descarta* é responsável por realizar o teste de pertinência do volume envolvente da região do reservatório contra o volume de visão da câmera. Maiores detalhes a respeito desse teste podem ser encontrados nos trabalhos de Thomas [Tom99] e Hofmam [Hof02].

A prova de que os modelos extraídos por este algoritmo são ótimos é idêntica à prova do algoritmo de extração dependente do erro geométrico, desde que a propriedade de ordenação parcial da hierarquia seja válida também para o erro projetado. Este fato é verdadeiro se considerarmos que a distância da *aabb* de um nó pai à câmera é sempre menor do que (ou igual a) a distância da *aabb* de seus filhos à câmera (veja equação 5.1).

5.3 Extração Dependente do Número de Polígonos

O algoritmo de extração **dependente do número de polígonos** é adequado à natureza interativa da visualização, pois possibilita o controle preciso de um dos fatores

determinantes para a interação: o número de polígonos enviados para o canal de visualização (*rendering pipeline*).

Dado um número de células ***NP***, o objetivo é achar, dentre todos os modelos de complexidade igual a ***NP*** e pertencentes a μ', aquele que possui o menor erro geométrico (ou projetado).

O algoritmo utilizado é guloso. Ele inicia com o corte mínimo da hierarquia, ou seja, com o corte cujo modelo associado é formado apenas pelo nó raiz e, a cada passo, altera o corte atual descendo localmente pelo nó que possui maior erro. A etapa de descida é repetida até que o corte atual possua ***NP*** nós ou que seja formado apenas por folhas.

Considerando que o número de polígonos pré-estabelecido deve ser limitado pelo número de polígonos do modelo original, o pseudo-código do algoritmo, em alto nível, é:

```
void Extração_dependente_do_número_de_polígonos (int NP, tipo_nó* nó) {
        tipo_nó_lista* corte;
        corte→elem = nó;
        int tamanho = 1;

        while (tamanho < NP) {
                Desce(corte);
                Tamanho++;
        }
        Seleciona (corte);
}
```

O método *Desce* recebe como parâmetro a lista que representa o corte atual e é responsável por retirar o nó de maior erro e inserir seus nós filhos. Aqui, vale uma observação quanto à implementação: deve ser utilizado um *heap* para controlar eficientemente a escolha do maior erro do corte. Com esse *heap*, pode-se calcular o erro máximo do novo corte em O(log ***NP***). Assim, a extração do modelo pode ser feita em O(***NP*** log ***NP***), perdendo a propriedade de extração linear, mas ainda permitindo a obtenção de taxas de iteração adequadas na prática.

Indutivamente, pode-se provar que os modelos gerados pelo algoritmo são ótimos. Para tanto, é importante considerar a propriedade P.4.2 e, também, que o erro associado a um corte é o máximo dentre os erros de todos os nós que formam o corte. Para ***NP*** igual a 1 e 2, é trivial verificar que os modelos são ótimos, pois, no espaço de modelos, só há um modelo composto por uma célula e um modelo composto por duas células. Para ***NP*** igual a 3, também é fácil verificar que o algoritmo guloso funciona.

Supondo que o algoritmo seja ótimo para ***NP*** igual a ***L***, pode-se verificar que a troca efetuada pela função *Desce* gera também o corte ótimo para ***L***+1, pois essa é a única troca que diminui o erro existente no corte anterior. Isso mostra que a aplicação seqüencial das trocas efetuadas por *Desce*, a começar por ***NP*** igual a 3, leva somente a modelos ótimos.

5.4 Dependência do Erro de Propriedade

Como visto no capítulo anterior, uma propriedade pode ser adicionada à estrutura de MR para ser visualizada. Portanto, é interessante que os algoritmos de extração de modelos possibilitem considerar o erro de propriedade na geração das aproximações.

Essa característica pode ser facilmente introduzida. Neste trabalho, os algoritmos dependentes do erro geométrico e da câmera incorporam-na através da efetuação de um teste duplo, permitindo que um nó seja selecionado se tanto o limite de erro geométrico quanto o de propriedade forem satisfeitos. No caso do algoritmo dependente da câmera, o erro de propriedade pode ser invariante à projeção.

No algoritmo de extração dependente do número de polígonos, a solução adotada foi um pouco diferente. Em vez de incorporar o erro de propriedade na função de avaliação de custo do *heap*, foi feita uma mescla entre as dependências: o algoritmo continua essencialmente dependente do número de polígonos, porém um limite máximo de erro de propriedade é permitido. Se não for possível respeitar esse limite, o algoritmo acaba retornando uma malha com mais polígonos do que o desejado.