Anais do 12O Encontro de Iniciação Científica e Pós-Graduação do ITA – XII ENCITA / 2006
Instituto Tecnológico de Aeronáutica, São José dos Campos, SP, Brasil, Outubro, 16 a 19, 2006

RECONSTRUÇÃO GRÁFICA TRIDIMENSIONAL DE EDIFICAÇÕES URBANAS A PARTIR DE IMAGENS AÉREAS

Daniel Moisés Gonzalez Clua
Universidade do Vale do Paraíba – UniVap
Av. Shishima Hifumi, 2911 – Urbanova – São José dos Campos, SP, 12244-000
Bolsista PIBIC-CNPq INPE
dmgc29785@yahoo.com.br

Valdemir Carrara
Instituto Nacional de Pesquisas Espacias – INPE / Divisão de Mecânica Espacial e Controle – DMC
Av. dos Astronautas, 1.758 – Jardim da Granja – São José dos Campos, SP, 12227-010
val@dem.inpe.br

Resumo. *Este trabalho tem como objetivo o desenvolvimento de algoritmos para a obtenção das dimensões de construções de edifícios a partir do processamento de imagens aéreas de alta resolução. Estas dimensões serão posteriormente utilizadas para compor objetos gráficos tridimensionais utilizando a própria textura obtida da imagem. No presente trabalho, os vértices das construções deverão ser fornecidos por um dispositivo apontador. O trabalho compreende também a eliminação da paralaxe, a correção da iluminação em função do ângulo de elevação solar, e a eliminação de sombra na textura. A metodologia a ser utilizada no desenvolvimento da presente proposta envolve a aplicação de geometria analítica e vetorial no desenvolvimento de algoritmos para compor as dimensões das construções tridimensionais. O algoritmo será desenvolvido em linguagem C ou C++, com visualização realizada por meio de OpenGl. Os principais planos para o trabalho envolvem: elaborar uma revisão bibliográfica a respeito da elaboração poligonal e geometria vetorial, familiarizar-se com técnicas de elaboração poligonal do tipo OpenGl e DirectX, desenvolver algoritmos para composição de objetos geométricos simples e para a manipulação destes em imagens, e para efetuar transformações e correções em imagens. Tais algoritmos serão implementados para compor objetos gráficos tridimensionais a partir das informações obtidas das imagens, com aplicação de texturas. Mais adiante, os códigos serão otimizados e novos recursos serão implementados.*

Palavras chave: Computação Gráfica, OpenGl, Reconstrução de Edifícios.

1. Introdução

A construção de ambientes gráficos tridimensionais (ambiente virtual) no computador tem encontrado diversas aplicações recentemente. Embora ambientes virtuais sejam freqüentes em jogos, cenários tridimensionais que representam ambientes reais são cada vez mais empregados no turismo, no urbanismo, no tratamento de distúrbios nervosos (fobias) e simuladores de vôo, veículos e máquinas. Também é usual, em tais ambientes, a reconstrução virtual de partes de cidades ou mesmo cidades inteiras. Uma vez que esta reconstrução pode ser bastante complexa e difícil, em virtude dos vários cenários existentes, buscam-se ferramentas automáticas ou semi-automáticas para tal reconstrução (VTP 2005, Carmenta 2005). Imagens aéreas e imagens obtidas por satélites com alta resolução são utilizadas usualmente, o que permite reconstruir grandes áreas com poucas imagens. O processo de reconstrução virtual pode ser dividido em duas partes: na primeira buscam-se formas automáticas ou semi-automáticas de extração de características (ou informações) das imagens para detectar as construções, e na segunda efetua-se a reconstrução propriamente dita. Pretende-se neste trabalho investir na reconstrução de ambientes urbanos (edifícios e casas) a partir de informações previamente fornecidas a um programa computacional, o que caracteriza um método semi-automático.

Portanto, o objetivo deste trabalho é elaborar um programa que permite reconstruir espacialmente e visualizar construções e edifícios a partir de imagens aéreas. A linguagem usada é C++, com recursos de OpenGL para a visualização. O OpenGL é definido como "uma interface em software para hardware gráfico". Em essência, é uma biblioteca de gráficos tridimensionais e modelagem, portátil e rápida, o que permite gerar imagens interativas em tempo-real.

O objetivo deste trabalho será, portanto, reconstruir um ambiente tridimensional com base em imagens aéreas de zonas urbanizadas de cidades, a partir de informações providas por usuários externos. Estas informações devem permitir, no mínimo, obter a geometria da construção e sua altura. Com a intenção de aumentar o realismo da cena gerada, deve-se procurar, sempre que possível, utilizar a própria textura da imagem para compor as diversas faces dos objetos gráficos reconstruídos. No caso de imagens aéreas, é provável que somente a textura do topo possa ser empregada, pois as faces laterais são vistas num perfil bastante distorcido. Dois outros requisitos são importantes nesta reconstrução. O primeiro deles é um processamento na imagem aérea para minimizar a sombra da construção, já que este ambiente poderá ter iluminação artificial gerada pelo OpenGl, o que criaria uma superposição de sombras caso a sombra natural não fosse reduzida ou eliminada. O segundo é a eliminação da projeção do edifício na imagem, para que não ocorra a impressão de haver duas construções num mesmo local numa vista aérea do ambiente tridimensional.

Adicionalmente o programa gerado deve permitir criar, modificar e salvar em disco as diversas construções. Os métodos geométricos empregados e os algoritmos desenvolvidos são descritos no próximo capítulo. No capítulo 4 apresentam-se os resultados parciais obtidos e as conclusões são apresentadas no capítulo seguinte.

2. Desenvolvimento

A reconstrução será realizada a partir de coordenadas de pontos fornecidas externamente, isto é, não são calculadas automaticamente. Estas coordenadas definem o contorno do topo das construções, o que permite reconstruir um prisma vertical sobre a imagem. A altura deste prisma será obtida com base no prévio conhecimento do ângulo de elevação do Sol sobre o horizonte na imagem, e também no comprimento da sombra do edifício, supostamente contida num plano horizontal, perpendicular ao próprio edifício. O ângulo de elevação do Sol, por sua vez, será também fornecido externamente, ou então calculado com base no conhecimento da altura real de uma construção qualquer presente na imagem. Caso a sombra de um determinado edifício seja obstruída na imagem pela presença de uma outra construção próxima, então a altura pode ainda ser estimada com base na altura deste edifício projetada na imagem, caso a linha de visada da câmera não seja totalmente ortogonal à imagem neste ponto, pois neste caso esta projeção resulta num único ponto ou apenas um pequeno segmento.

Uma vez que o OpenGl utilizado na visualização permite o uso de polígonos triangulares e quadrangulares, e dado que o contorno do topo pode eventualmente ser um polígono com mais do que quatro lados torna-se necessário construir um algoritmo para dividir um polígono de *n* lados em triângulos.

As principais funções a serem implementadas para efetuar a reconstrução virtual são, portanto:

1. Exibir a imagem aérea e permitir ao usuário o fornecimento de pontos (vértices) do topo da construção, que irá definir um prisma vertical com base não necessariamente retangular.
2. Obter parâmetros que permitam avaliar a altura da construção a partir da sombra projetada na imagem ou da própria altura projetada (em caso de vista em perfil), por meio de uma formulação geométrica.
3. Dividir o polígono fornecido do contorno do topo num conjunto de triângulos ou triangularização.
4. Extrair a textura a ser aplicada à construção da própria imagem.
5. Eliminar ou minimizar a sombra do edifício na textura do solo.
6. Armazenar as construções em disco

Estes itens são descritos nas seções seguintes.

2.1. Fornecimento dos vértices

Para a reconstrução do ambiente virtual utilizou-se funções do OpenGl descritas no livro OpenGl Super Bible (Wright, 2000) e em programas tutores encontrados na Web (Neon-Helium Productions, 2005). Para o correto funcionamento do OpenGl é necessário a inclusão no programa dos arquivos `gl.h`, `glu.h`, e `glaux.h`, além da biblioteca `windows.h` que contêm valores previamente associados a constantes. São também definidas nestes arquivos as especificações de tipos de variáveis, uma vez que o OpenGl, por ser multi-plataforma, estabelece regras para os tipos usuais, como `GLfloat`, `GLint`, etc. Estes arquivos em geral são fornecidos com o compilador C++, mas podem ser obtidos facilmente na web caso contrário. Além disso as seguintes bibliotecas devem ser incluídas no projeto para se poder utilizar os comandos de OpenGL no ambiente Windows: `OpenGL32.lib`, `GLu32.lib`, e `GLaux.lib`.

A imagem aérea será exibida numa janela do programa, o que permite ao usuário indicar, por meio do mouse, os pontos para a reconstrução dos edifícios. Para tanto esta imagem é aplicada como textura – usando o OpenGl – a um objeto retangular de mesmas proporções em vista frontal. O programa irá permitir também que o usuário mova, acerque ou distancie a imagem, através de comandos pelo teclado ou mouse. Como pode ser visto no exemplo da Fig. 1, o programa será dividido em duas janelas. Ambas mostram a mesma imagem, uma usando projeção em perspectiva e outra usando projeção paralela, sem distorção na imagem. A projeção em perspectiva irá permitir a visualização tridimensional e deve ser movimentada e vista sob diferentes ângulos. A projeção paralela permite ser acercada ou afastada, sem possibilidade de rotação, já que é nela que é feita a leitura de pontos para construção do edifício. Uma vez construído, a representação do edifício será exibida na janela com projeção em perspectiva.

O usuário deve fornecer então um número qualquer de pontos para o topo do edifício (os pontos são seqüenciados internamente no programa a partir de P_1: P_1, P_2, ..., P_n), segundo o contorno deste topo, mostrado em vermelho na Fig. 2. Um clique do botão direito do mouse permite que o programa reconheça quando foi fornecido o último ponto do contorno. Os próximos dois pontos a serem fornecidos representam o vetor da altura do edifício (h) (em verde na Fig. 2) e finalmente dois outros pontos que representam o comprimento do vetor da sombra s (em roxo). Uma vez que a leitura das coordenadas destes pontos, efetuada pelo mouse, estão em unidades de pixel da janela, é então necessário convertê-las para unidades de pixel da imagem por meio de uma transformação linear:

$$T_{xi} = a\ x_i + d_x$$

$$T_{yi} = a\, y_i + d_y$$

onde a é a ampliação (fator de escala) utilizada na visualização. Estes valores são então atribuídos a dois vetores, um para as coordenadas no eixo x e outra para as do eixo y. Ambos possuem os valores de todos os pontos que representam o topo do edifício e também os dos vetores altura e da sombra. O vetor da altura serve para calcular o deslocamento que o programa deve aplicar ao objeto para que ele se origine da base do edifício na imagem e não do topo. Já o vetor da sombra é necessário para calcular a altura do edifício, através de um cálculo com o ângulo de elevação do Sol, conforme será mostrado na próxima seção. Passados estes pontos, o programa constrói o objeto que representa o edifício, usando os valores armazenados das coordenadas, como visto nas Figuras 2 e 3.

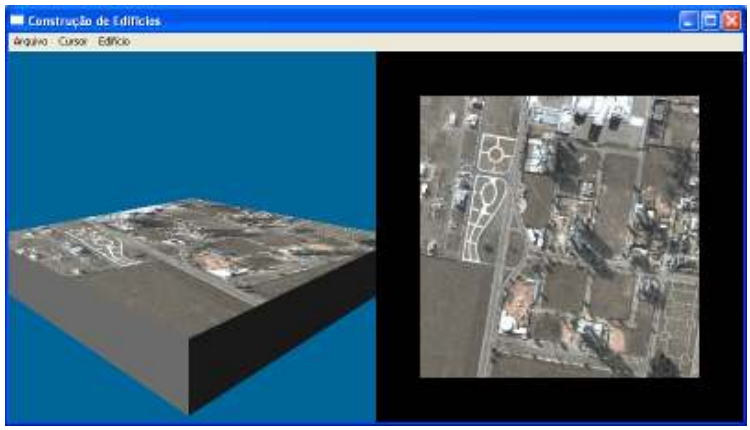

Figura 1. Janela de visualização (à esquerda) e edição (à direita).

Figura 2. Pontos recebidos para a construção do edifício.

Figura 3. Edifício reconstruído.

Para possibilitar a reconstrução de vários edifícios, deve ser criada uma classe para representá-los. Deste modo, podem ser instanciados diversos objetos desta classe, sendo que cada um conterá todas as suas informações necessárias,

como a sua posição no plano, os vértices fornecidos pelo usuário, a altura, etc. Além de facilitar a construção dos edifícios na cena, a classe irá permitir que, mais adiante, o usuário possa selecionar um edifício já construído e modificá-lo.

2.2. Formulação geométrica

A formulação geométrica possui por objetivo determinar a altura dos edifícios a partir de informações da própria imagem. Tomando como exemplo a imagem mostrada na Fig. 4, deseja-se obter a altura do edifício em unidades de pixel, representada pela magnitude do vetor *H* perpendicular ao plano da imagem, e cuja projeção no plano da imagem é o vetor *h*. É ainda mostrado na figura o vetor *s*, que indica o comprimento da sombra do edifício projetada no plano horizontal, e considerada como sendo em verdadeira grandeza.

Figura 4. Imagem aérea e os principais pontos que devem ser utilizados no processo de reconstrução. O contraste da imagem foi reduzido artificialmente

Nota-se que, embora a altura do edifício não seja vista na sua real dimensão na imagem, por outro lado a sombra tem sua verdadeira dimensão (ou bem próxima dela), se for suposto que a câmera tenha seu eixo perpendicular ao solo, como ilustra a Fig. 5. Assim, se o ângulo α que representa a elevação do Sol sobre o horizonte for conhecido, então a altura *H* do edifício pode ser calculada por meio do comprimento da sombra do edifício na imagem, ou seja, do módulo do vetor *s*, conforme mostra a Fig. 5, resultando então

$$|H| = |s| \tan\alpha \tag{1}$$

Se o comprimento *s* da sombra não estiver disponível para um dado edifício (por exemplo, se a sombra for projetada sobre uma outra construção), pode-se estimar sua altura tendo como base a altura projetada, desde que seja conhecida a altura *H'* e sua projeção *h'* de um outro edifício próximo. Neste caso emprega-se uma regra de proporcionalidade:

$$|H| = |h| \frac{|H'|}{|h'|} \tag{2}$$

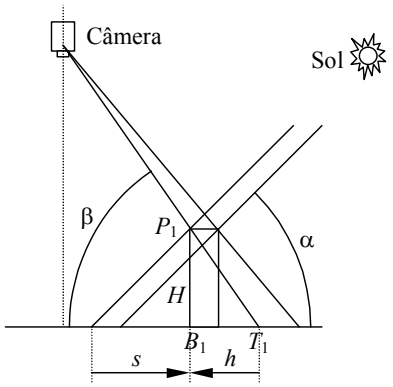

Figura 5. Vista lateral da geometria do problema.

Caso ainda o ângulo α não seja conhecido, pode-se estimá-lo desde que se conheça a altura real de alguma construção e a resolução da imagem, isto é, o tamanho de cada pixel, ou ainda o comprimento da sombra deste edifício em unidades métricas. Seja então H'_m a altura em metros de uma determinada construção conhecida e seja $|s'|$ o comprimento de sua sombra na imagem (em pixels) Se R for a resolução (em metros/pixel), então a altura do edifício H' será dada por:

$$|H'| = \frac{H'_m}{R} \text{ (em pixels).} \tag{3}$$

O ângulo de elevação do Sol pode então ser calculado invertendo-se a Eq. 1 para α.

2.3. Algoritmo para a triangularização de polígonos planos

A construção de objetos em OpenGL deve ser feita exclusivamente com triângulos planos. Portanto, para possibilitar a construção de edifícios com qualquer formato – não apenas retangulares – foi necessário desenvolver um algoritmo para realizar a triangularização do polígono definido pelos pontos passados pelo usuário, que representa o contorno do topo do edifício.

O problema da triangularização de polígonos planos consiste em obter um conjunto de triângulos T_i, com i = 1, 2, ..., $1 \le n - 2$, tal que este conjunto corresponda a uma divisão de um polígono fechado formado por uma seqüência de coordenadas na forma $P_i = (x_i, y_i)$, com i = 1, 2, ..., n, ($n \ge 3$), como mostra a Fig. 6.

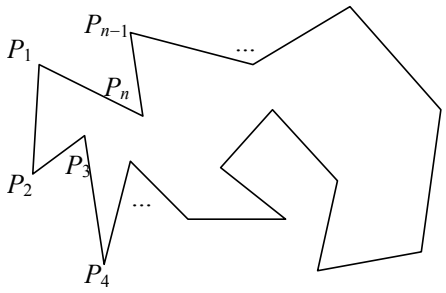

Figura 6. Pontos P_1 a P_n, que definem o polígono.

A solução adotada neste algoritmo será eliminar triângulos do polígono um a um, a partir dos triângulos externos, isto é, tais que duas de suas arestas sejam arestas consecutivas do polígono. Desta forma os triângulos escolhidos são formados por 3 vértices em seqüência, como P_4, P_5, P_6, por exemplo. Uma vez eliminado um triângulo, será constituído um novo polígono com um vértice a menos. O processo será então repetido para este novo polígono, até que reste apenas um triângulo.

O algoritmo será dividido em duas partes. Na primeira, deve-se modificar o polígono original para que satisfaça condições necessárias para a segunda parte. Na segunda, os triângulos serão removidos. A primeira destas condições é que o polígono seja formado por 3 ou mais pontos. Além disso, nenhuma aresta deve apresentar comprimento nulo, isto é, tal que P_i seja igual a P_{i+1}. Em seguida, deve-se verificar se não existem vértices "supérfluos", isto é, inseridos numa aresta, fazendo com que haja 3 ou mais pontos em seqüência alinhados a uma mesma reta. Outra condição necessária é

que o polígono não seja "dobrado", isto é, tal que não ocorra uma interseção entre duas arestas quaisquer. Finalmente, a última condição é que o sentido de percurso dos vértices seja único, horário ou anti-horário. Será adotado aqui que o sentido normal é o anti-horário, o que significa que se o percurso for horário a ordem dos vértices deverá ser invertida para gerar um polígono anti-horário, como mostra a Fig. 7.

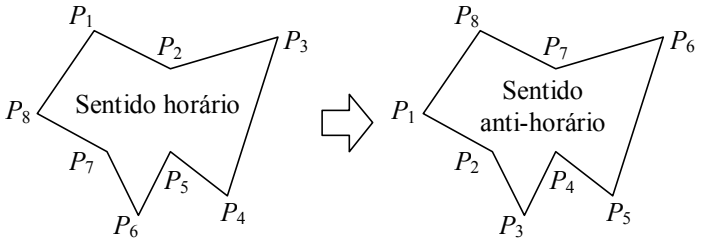

Figura 7. Inversão do sentido dos pontos que definem o polígono.

Foram desenvolvidos algoritmos para testar e corrigir todas as condições apontadas acima. A condição de que o polígono não seja "dobrado" sobre si mesmo foi inserida no código que armazena os pontos fornecidos pelo usuário. Este algoritmo impede a tentativa de se fornecer um ponto cuja aresta formada com o ponto anterior cruze qualquer outra aresta já fornecida.

A segunda parte do algoritmo consiste na subdivisão do polígono em triângulos, de tal forma que, no final, o conjunto destes triângulos forme o polígono original. Ao todo serão removidos $n - 2$ triângulos do polígono original, sendo que n é o número de vértices do polígono. A Fig. 8 ilustra um polígono e a ordem de retirada dos triângulos válidos.

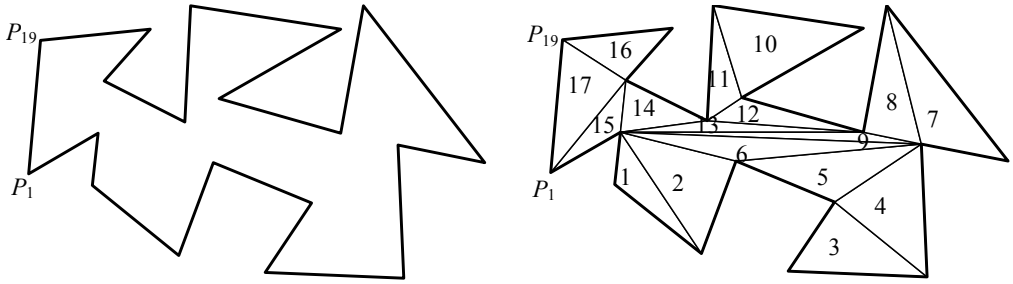

Figura 8. Subdivisão do polígono em triângulos.

Os triângulos a serem removidos são selecionados com base no produto vetorial de duas arestas consecutivas. Como os vértices percorrem um sentido anti-horário, então se o produto vetorial for positivo o triângulo a ser removido faz parte do polígono. Caso contrário trata-se de um triângulo externo e o polígono é côncavo neste local. É necessário cuidar ainda para que o triângulo a ser eliminado não contenha um vértice qualquer do polígono em seu interior, o que poderia causar erros na triangularização. Caso isto ocorra, o algoritmo procura então os próximos triângulos a serem removidos até que encontre um cujo interior não contenha vértices.

A Fig. 9 ilustra o processo de triangularização realizado pelo algoritmo num polígono qualquer. Os triângulos foram coloridos aleatoriamente na figura para facilitar a sua identificação.

2.4. Aplicação de textura no edifício

Para dar uma aparência mais real aos edifícios tridimensionais construídos, usou-se um método para aplicar texturas ao topo e às suas laterais. Como o topo to edifício está sempre visível nas imagens, esta pode ser aplicada diretamente ao edifício tridimensional reconstruído. As laterais, porém, nem sempre estarão totalmente visíveis. Por isso, usam-se outras imagens para simular estas texturas e não a própria do edifício.

Estas imagens armazenam desenhos de diferentes tipos de padrões que simulam laterais de edifícios e foram construídas com o uso de um editor de imagens comum (MS Paint). Para melhor eficiência e resultado, o tamanho destas imagens é pequeno – aproximadamente a extensão de um andar. O algoritmo repete então esta textura o número de vezes necessário para cobrir toda a área de cada face do edifício. Para dar um maior realismo, procurou-se manter uma escala compatível com a real. Por exemplo, se um edifício real possui 15 andares, espera-se que sua reconstrução digital tenha aproximadamente o mesmo número de andares. Foi necessário, então, fazer um algoritmo que calcula o tamanho da face e determina quantas vezes a textura deve ser aplicada, tanto na vertical como na horizontal. Para o comprimento, isto é feito calculando-se a distância euclidiana entre dois vértices consecutivos da base (ou do topo).

Para a altura não é necessário nenhum cálculo, já que cada edifício armazena, em uma variável, a sua respectiva altura que será a mesma para todas as suas faces. Também se deve pré-definir qual será o tamanho da textura aplicada. Isto depende do tamanho da imagem e da escala em que se está trabalhando, mas deve-se usar um valor que gere um bom resultado, de modo que a textura não fique nem muito pequena nem muito grande em relação ao edifício.

Figura 9. Processo final do algoritmo de triangularização.

Tendo o comprimento e altura tanto do edifício quanto da textura, é possível calcular quantas vezes está terá que ser repetida para cobrir toda a lateral. Como este número deve ser inteiro, faz-se um arredondamento do valor calculado para o número inteiro mais próximo. Para isto, usam-se as funções `floor()` e `ceil()` da biblioteca matemática do C++. O resultado será que a última cópia da textura será esticada ou comprimida, mas como foi calculado qual destas possibilidades está mais perto do tamanho real, a distorção não será tão significativa.

Com os valores dos números de repetições, dividem-se as distâncias vertical e horizontal calculadas anteriormente pelo número de repetições, o que resultará no tamanho que cada cópia da imagem terá. A partir do ponto inicial da lateral, e a cada aplicação da imagem, a distância na horizontal é incrementada. Quando o final da face é atingido, retorna-se ao lado inicial e incrementa-se a distância na vertical. Isso é repetido sucessivamente até que a face seja totalmente preenchida pela textura, como ilustra a Fig. 10. Este conceito de repetição de texturas permite que se construa uma lateral completa de um edifício a partir de uma ou poucas imagens elementares (contendo apenas uma janela, por exemplo). Além disso, podem ser criadas diferentes padronizações com imagens e selecioná-las de forma aleatória em cada edifício reconstruído para que não fiquem todos iguais.

a *b*

Figura 10. Exemplo de textura (*a*) e sua aplicação em um edifício (*b*).

2.5. Minimização da projeção da sombra na imagem original

Uma vez que a reconstrução dos edifícios é realizada graficamente, deseja-se que a iluminação na cena reconstruída possa ser alterada sob ação externa. Em outras palavras, deseja-se que as sombras dos objetos na cena acompanhem o movimento da posição da fonte de luz artificial quando esta for modificada, de modo que a projeção tenha um aspecto real. Isto cria um problema, pois em geral as imagens aéreas que servem como textura do solo apresentam sombras naturais dos edifícios. Para que não houvesse uma "sobreposição" de sombras (uma natural e outra artificial), o que causaria uma indefinição na direção real de iluminação, foi necessário encontrar uma maneira de se

"eliminar" a sombra natural. Admitindo-se que a área que a sombra ocupa no solo (polígono de sombra) seja conhecida, pode-se reaplicar a própria textura do solo a este local com uma claridade maior que a do solo. Isto é feito elevando-se os valores de cores vermelha, verde e azul de cada pixel. As quantidades elevadas em cada canal de cor são definidas pelo usuário e devem ser ajustadas empiricamente. Na verdade, cria-se uma nova imagem da textura do solo com os valores dos pixels todos alterados, como mostra o seguinte algoritmo:

```
for (int i=0; i<3*Image->sizeX*Image->sizeY; i++){
  int j = i%3;
  int valor = Image->data[i];
  valor+=brilho[j];
  if(valor>255)
        valor=255;
  else if(valor<0)
        valor=0;
  Image->data[i]=valor;
}
```

O laço for da primeira linha, percorre todos os pixels. Como cada pixel armazena três valores (para as cores vermelho, verde e azul), deve-se começar no primeiro, que tem índice zero e terminar em três vezes o número total de pixels, que por sua vez é obtido pela multiplicação do tamanho em *X* pelo tamanho em *Y* (a variável `Image` é um ponteiro para a imagem utilizada). A operação `i%3` na segunda linha calcula o resto da divisão de `i` por três. Isto determina a qual cor do pixel a variável `i` atual se refere e é armazenado em `j`. Em seguida, armazena-se em valor a quantidade de uma cor no pixel atual (o vetor `data[]` armazena as informações dos pixels da imagem) e soma-se a ele uma intensidade naquela cor armazenada no vetor `brilho[]`. Deve-se verificar se o valor do pixel não está acima de 255 ou abaixo de zero, para não ultrapassar os limites de cor e resultar em efeitos indesejáveis na imagem. Finalmente, o valor do pixel original é substituído pela variável valor que agora está com sua intensidade alterada. Esta variável é controlada pelo usuário e pode ser incrementada ou diminuída de modo a encontrar o melhor resultado. Esta textura é então aplicada ao polígono da sombra.

Aplicando-se a quantidade certa de nas três cores, obtêm-se o desaparecimento praticamente completo da sombra na imagem original, possibilitando a projeção de sombras dinâmicas na cena (ainda não implementado atualmente). Na presente fase do projeto, o usuário deve indicar os vértices do polígono da sombra. Essa entrada funciona de maneira semelhante à entrada de vértices do edifício, e o polígono passa pelo processo de triangularização de forma idêntica àquela realizada no topo. A Fig. 11 mostra um exemplo no qual a sombra do solo foi minimizada. Percebe-se, neste exemplo, que a completa eliminação da sombra é algo difícil, senão impossível de ser realizada. Contudo a redução do seu efeito na textura do solo irá permitir que a direção de iluminação artificial seja evidenciada sem ambigüidade.

a *b* *c*

Figura 11. Exemplo de edifício com a sombra original (*a*), seleção da área da sombra (*b*) e eliminação da sombra (*c*).

2.6. Gravação de dados

Para possibilitar a gravação dos dados de edifícios construídos para recuperação futura, foi adicionada uma opção de gravação de dados, que armazena todos os dados necessários referentes ao edifício, como seus vértices, sua altura, posição, etc. Isto é feito através de comandos como:

- `fout.open(char *Filename);` para abrir um arquivo para escrita, ou:
- `ifstream fin(char *Filename);` também para abrir um arquivo, mas desta vez para leitura.

São ainda utilizados os comandos

- `fout << char *String`
- `fin >> char *String`

para gravar e recuperar uma linha do arquivo, respectivamente. É importante lembrar que, para que este tipo de entrada e saída de dados seja possível, é necessário incluir a biblioteca `fstream.h`.

As informações são gravadas em um arquivo de extensão "`.txt`", como mostra a Fig. 12. Cada linha representa o valor das variáveis necessárias para construir os edifícios salvos. Ao ler o arquivo, os valores serão novamente armazenados em suas respectivas variáveis:

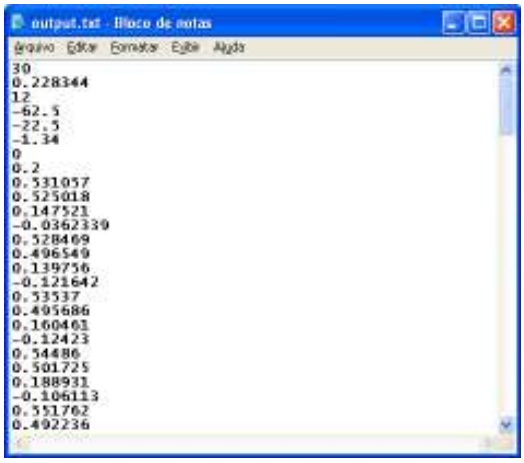

Figura 12. Informações armazenadas no arquivo "`output.txt`".

3. Resultados preliminares

Os algoritmos descritos no capítulo anterior foram implementados no programa `Edificios` cuja interface foi apresentada na Fig. 1. A versão atual permite a reconstrução de um número qualquer de edifícios presentes na imagem, com aplicação da textura real no topo e uma textura artificial nas laterais, como visualizado na Fig. 13. Nota-se na imagem que o topo do edifício mais próximo possui formato de H, ou seja, é um polígono complexo que foi corretamente dividido em triângulos.

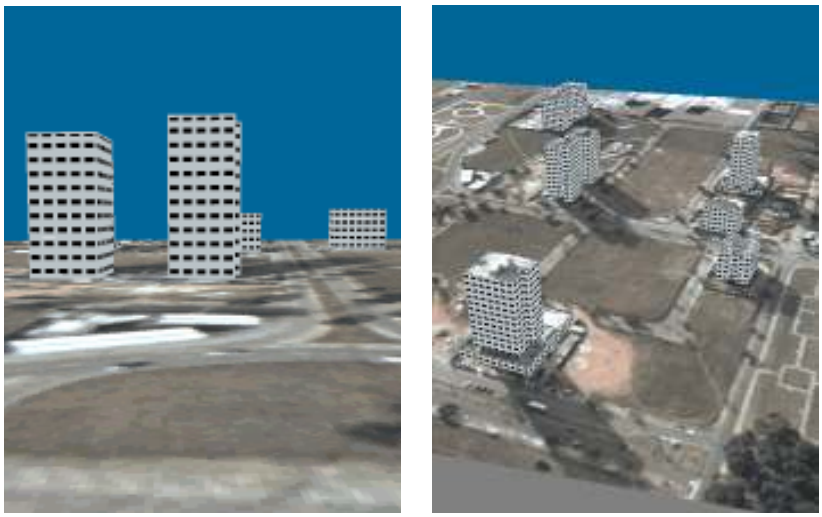

Figura 13. Exemplos de edifícios reconstruídos pelo programa.

Foi implementada a minimização da sombra (mostrada na Fig. 11), porém resta ainda a ser desenvolvido um algoritmo para a eliminação da projeção do edifício na imagem que serve como textura do solo. Resta ainda a ser implementada a iluminação artificial, por meio de uma fonte de luz do próprio OpenGl, com sombreamento de Gouraud (Gouraud, 1971), bem como uma melhoria nas texturas empregadas nas faces laterais. De fato, como pode ser

observado na Fig. 13, o número de andares não é um número inteiro (há uma quebra na textura das janelas no último andar), o que deve ser corrigido na próxima versão. Deve-se também introduzir um número maior de texturas de janelas, para dar conta da variabilidade encontrada em edifícios reais, além de um aprimoramento no fator de escala, já que o número de andares presentes na imagem virtual não é igual ao número andares do edifício real.

4. Conclusões

Tendo as posições de todos os vértices referentes à base do edifício, sabendo sua altura e com todas as texturas prontas, o edifício é reconstruído tridimensionalmente na visão em perspectiva do programa, a qual é possível movimentar livremente.

Na implementação atual, podem ser construídos vários edifícios, os quais são construídos a partir de polígonos com qualquer formato, apresentam texturas tanto na parte superior como nas laterais e têm suas sombras originais removidas. As cenas criadas podem ser salvas e carregadas. Ainda não há um processo de iluminação e os edifícios não podem ser alterados depois de construídos.

Como implementações futuras, serão desenvolvidos algoritmos para:

a) Iluminações artificiais, incluindo sombras dos edifícios projetadas no solo e em outros edifícios.
b) Melhoria na composição de texturas e melhoria na determinação do fator de escala das imagens.
c) Eliminação da projeção do perfil do edifício na textura aplicada ao solo.

5. Agradecimentos

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq, pelo auxílio financeiro de bolsa de Iniciação Científica.

Ao Instituto Nacional de Pesquisas Espaciais - INPE pela oportunidade de estudos e utilização de suas instalações.

6. Referências

Carmenta – SpatialAce, 2005, (http://www.spatialace.com).

Gouraud, H., 1971, “Continuous Shading of Curved Surfaces”. In IEEE Transactions on Computers. Vol. C-20, No. 6, pp. 623-629.

Neon-Helium Productions, NeHe., 2005, (http://nehe.gamedev.net/).

VTP - The Virtual Terrain Project., 2005, (http://www.vterrain.org/).

Wright, S. R.; Sweet, M., 2000, “OpenGL Super Bible. Second Edition”. Waite Group Press.