Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Curso de Pós-Graduação em Engenharia de Eletricidade

RONALD SILVA SERRÃO

ALGORITMO DE OTIMIZAÇÃO PARA VISUALIZAÇÃO DE MODELOS URBANOS COM BANCO DE DADOS GEOGRÁFICOS

São Luís
2008

RONALD SILVA SERRÃO

ALGORITMO DE OTIMIZAÇÃO PARA VISUALIZAÇÃO DE MODELOS URBANOS COM BANCO DE DADOS GEOGRÁFICOS

Dissertação apresentada ao Curso de
Pós-Graduação em Engenharia de Eletricidade da
Universidade Federal do Maranhão como parte
dos requisitos necessários para obtenção do
grau de Mestre em Engenharia Elétrica.

Orientador: Anselmo Cardoso de Paiva

Co-orientador: Aristófanes Corrêa Silva

São Luís
2008

Silva Serrão, Ronald

Algoritmo de Otimização para Visualização de Modelos Urbanos com Banco de Dados Geográficas / Ronald Silva Serrão. - São Luís, 2008.

96f.:il.

Impresso por computador (Fotocópia).

Orientador: Anselmo Cardoso de Paiva.

Dissertação (Mestrado) - Universidade Federal do Maranhão, Programa de Pós-Graduação em Engenharia de Eletricidade, Centro de Ciências Exatas e Tecnologia, 2008.

1. Realidade Virtual 2. Modelo Virtual Urbano 3. Banco de Dados Geográficos 4. Sistema de Informação Geográfica. I. Título.

CDU 007.51

ALGORITMO DE OTIMIZAÇÃO PARA VISUALIZAÇÃO DE MODELOS URBANOS COM BANCO DE DADOS GEOGRÁFICOS

Ronald Silva Serrão

Dissertação aprovada em 19 de junho de 2008.

Prof. Anselmo Cardoso de Paiva, Dr.
(Orientador)

Prof. Aristófanes Corrêa Silva, Dr.
(Co-orientador)

Prof. Alberto Barbosa Raposo, Dr.
(Membro da Banca Examinadora)

Prof. Mário Antonio Meireles Teixeira, Dr.
(Membro da Banca Examinadora)

"A verdade não é, de modo algum, aquilo que se demonstra,.
mas aquilo que se simplifica."
Antoine de Saint-Exupéry

À minha família e amigos, pelo apoio e companheirismo.

Agradecimentos

Em primeiro lugar, a Deus.

Aos meus pais e meus irmão, por toda a dedicação, confiança e amor.

Ao meu amor Ádilla pelo carinho, compreensão, incentivo e apoio.

Ao meu orientador, Profº. Dr. Anselmo Cardoso de Paiva, e ao meu co-orientador, Profº. Dr. Aristófanes Corrêa Silva, pela compreensão, colaboração e orientação segura.

Aos amigos pela força que sempre me emprestam quando eu necessito.

A todos que contribuíram de maneiras diversas para a realização deste trabalho.

Resumo

Sistemas de Realidade Virtual têm sido utilizados em diversas áreas nos últimos anos, apresentando vários benefícios. Uma classe desses sistemas é a de manipulação e visualização de modelos virtuais urbanos, que podem ser usados para planejamento urbano, planejamento arquitetônico, visualização e jogos. Apresenta-se a concepção e o desenvolvimento de um algoritmo otimizado de visualização de modelos virtuais urbanos com *cache* baseado em Banco de Dados Geográficos. A estratégia de *cache* proposta possibilita um bom desempenho no caminhar (*walktrough*) em modelos virtuais urbanos. Para testar o algoritmo proposto foi construido um modelo virtual urbano baseado no centro histórico da cidade de São Luís.

Palavras-Chave: Realidade Virtual. Modelo Virtual Urbano. Banco de Dados Geográficos. Sistema de Informação Geográfica.

Abstract

Virtual Reality Systems have been used in diverse areas in recent years, providing many benefits. A class of these systems is for visualization and manipulation of virtual urban models, that can be used for urban planning, architectural, visualization and games. This work presents the conception and development of an optimized algorithm for urban virtual models visualization with cache, based in Geographical Database. The proposed cache strategy provides a good performance for urban virtual models walktrough. To test the proposed algorithm it was built an urban virtual model based on São Luís historical town.
Keywords: Virtual Reality. Urban Virtual Model. Geographical Database. Geographic Information System.

Artigo Científico Publicado pelo Autor Relacionado à Dissertação

SERRÃO, R. S. ; PAIVA, A. C. ; SILVA, A. C. . Architecture Based on Virtual Reality Techniques and Geographic Database for Storage and Visualization of Urban Virtual Models. In: 26th Urban Data Management Symposium (UDMS 2007), 2007, Stuttgart, Germany. Proceedings of the 26th Urban Data Management Symposium - UDMS 2007, 2007.

Lista de Tabelas

Lista de Figuras

Lista de Abreviaturas e Siglas

API	*Application Programming Interface*
AVC	Ambientes Virtuais Colaborativos
BLOB	*Binary Large OBject*
CAD	*Computer-Aided Detection*
DEM	*Digital Elevation Model*
DML	*Data Manipulation Language*
FPS	*Frames Per Second*
GID	*Numeric Geometry Identifier*
GLSL	*OpenGL Shading Language*
GPU	*Graphics Processing Units*
HLSL	*High Level Shading Language*
LOD	*Level of Detail*
MBR	*Minimum Bounding Rectangle*
OCCI	*Oracle C++ Call Interface*
OGC	*Open Geospacial Consortium*
OSG	OpenSceneGraph
ROAM	*Real-time Optionally Adapting Meshes*
RV	Realidade Virtual
SGBD	Sistema de Gerenciamento de Banco de Dados
SGBD-OR	Sistema de Gerenciamento de Banco de Dados Objeto Relacional
SIG	Sistema de Informação Geográfica
SQL	*Structured Query Language*
SRID	*Spatial Reference System Identifier*
TDE	Tipo de Dados Espaciais
VRML	*Virtual Reality Modeling Language*

Sumário

Capítulo 1

Introdução

Uma das áreas que tem utilizado largamente tecnologia de Realidade Virtual (RV) é a modelagem de ambientes urbanos, que possibilita a reconstrução realista de lugares existentes no mundo, ou inserção de ferramentas para a sua visualização interativa e a simulação de diversas situações possíveis de acontecer no mundo real. Além disso, a construção antecipada destes ambientes virtuais, simulando ambientes urbanos reais a serem construídos, também tem sido realizada a fim de prever e evitar problemas.

Os modelos virtuais urbanos têm sido amplamente empregados como interfaces para sistemas de informação. A RV possibilita a construção de modelos bastante interativos e a associação de outros componentes como banco de dados, páginas Web e componentes multimídia. Esses modelos favorecem seu uso intuitivo como repositório de informações, sendo útil em ambientes virtuais nas áreas de turismo, arte e cultura, planejamento urbanístico, sistemas de navegação, testes mecânicos e estruturais, impacto ambiental e visual, meteorologia, entre tantas outras. Por exemplo, pode-se construir ruas, bairros e, até mesmo, cidades e disponibilizar via Web para o turismo virtual.

Outro aspecto importante que pode ser explorado pela construção de modelos virtuais urbanos é a área de preservação do patrimônio histórico. Assim, pode-se usar esses modelos urbanos como ferramenta para a documentação científica de áreas urbanas de interesse histórico. Outra alternativa de utilização é a reconstrução virtual de monumentos ou regiões não preservadas ou parcialmente preservadas. Nesses casos, os modelos são usados como uma metáfora para uma

interessante interface para a navegação através de um conjunto de informações sobre a cultura e monumentos históricos em diversas mídias como fotografias, vídeos e hipertexto.

Os monumentos arquitetônicos são parte importante do nosso patrimônio cultural. Mas, enquanto que outros elementos desse patrimônio podem ser protegidos por trás de um vidro em um museu, monumentos arquitetônicos são amplamente utilizados e ameaçados por estar sob grandes influências como tráfego, a poluição do ar ou eventos destrutivos que causaram graves danos, como terremotos, incêndios ou guerras. Mas, por todos os meios, quando monumentos estão seriamente danificados, ou completamente destruídos, a quantidade e qualidade de qualquer documentação sobrevivente torna-se extremamente importante (Duran and Toz 2001).

Com as tecnologias de RV é possível a reconstituição de ambientes, personagens e cenários históricos, que podem ser criados usando gráficos tridimensionais em tempo real ou apenas fotos e vídeos, e permitem envolvimento e interação com um cenário real. A demonstração desses tipos de cenários em circunstâncias de interação serve como suporte para o desenvolvimento educacional, cultural e de caráter turístico.

Para atender essas demandas os ambientes virtuais, no caso os modelos virtuais urbanos, devem propiciar impressões semelhantes às que ocorrem no mundo real. Entretanto, devido ao tamanho desses modelos, sua construção requer, em geral, a utilização de *hardware* de alta performance e alto custo, o que restringe a utilização dessas tecnologias.

1.1 Objetivos

O propósito deste trabalho é a concepção e desenvolvimento de um algoritmo que propicie a aceleração da visualização de modelos virtuais urbanos, com *cache*, baseado em um *buffer* georeferenciado, que utiliza banco de dados geográficos como suporte para o armazenamento, recuperação de dados, e que permite a visualização e navegação em modelos urbanos de larga escala em tempo real, em uma plataforma de *hardware* convencional em termos de recursos computacionais. Além disso, propõe-se a construção de uma interface 3D para uma base de

dados espacial, que poderá ser utilizada para adicionar conteúdo multimídia, possibilitando o fornecimento de informações a respeito dos monumentos históricos presentes na região representada.

Foi utilizado um modelo urbano construído de maneira *ad hoc* a partir de um mapa da área do centro histórico da cidade de São Luís, tombada como patrimônio histórico da humanidade.

1.2 Organização do Trabalho

O trabalho está organizado da seguinte maneira:

No Capítulo 2, apresentam-se os conceitos básicos necessários ao entendimento das tecnologias que foram utilizadas no desenvolvimento deste trabalho. Sendo expostos detalhes sobre as técnicas de otimização da visualização de grandes modelos, grafo de cena, sistema de informações geográficas, e por fim banco de dados geográficos. Além disso, são apresentadas as tecnologias envolvidas na implementação deste trabalho.

No Capítulo 3, é feita a exposição de alguns trabalhos relacionados à otimização do *rendering* e visualização de modelos urbanos e com o emprego de banco de dados espaciais aliados à realidade virtual. Estes contribuíram para o embasamento teórico deste trabalho e motivaram o desenvolvimento de soluções até então não empregadas.

No Capítulo 4, é apresentado em detalhes o algoritmo de otimização de visualização baseado em cache com o protótipo desenvolvido para que seja possível a visualização do modelo construído.

E finalmente, no Capítulo 5, conclui-se o trabalho com uma reflexão sobre os resultados obtidos, além de sugerir trabalhos futuros a serem desenvolvidos.

Capítulo 2

Fundamentação Teórica

Neste capítulo, apresenta-se a fundamentação teórica em que se baseou o desenvolvimento deste trabalho. São mostrados os conceitos básicos relacionados ao assunto, que permitem o entendimento das tecnologias empregadas, as quais são apresentadas na seqüência.

2.1 Conceitos Básicos

Nessa seção, apresentam-se alguns conceitos básicos necessários ao entendimento das tecnologias que foram utilizadas no desenvolvimento deste trabalho. Destacando-se entre eles, algumas técnicas de aceleração do *rendering*, grafo de cena, sistema de informação geográfica e banco de dados geográficos.

Antes de conceituar as técnicas de aceleração propriamente ditas, faz-se necessária a explicação do que é *pipeline* gráfico. O *pipeline* gráfico é um conjunto de etapas realizadas pelo *hardware* para gerar a visualização de uma imagem de uma cena, a partir de alguns parâmetros desta. Os parâmetros da cena incluem os vértices, polígonos, dados sobre iluminação (fontes de luz, vetores normais das superfícies), transformações geométricas, entre outros.

Em (Tortelli and Walter 2007) há uma descrição mais detalhada do *pipeline* gráfico da nova geração de *hardware*s gráficos, que possuem estágios programáveis (*Vertex Shader*, *Geometry Shader* e *Pixel Shader*), vistos na Figura 2.1. Os *shaders* são conjuntos de comandos que aceitam recursos gráficos de entrada, executam uma série de instruções sobre esses recursos e apresentam o resultado.

Os *shaders* são escritos em linguagens específicas, tais como: HLSL (*High Level Shading Language*) para Direct3D, ou, GLSL (OpenGL *Shading Language*) para OpenGL (OpenGL 2006), que permitem a programacão de *shaders* em nível de algoritmos. A implementacão de *shaders* em uma aplicação gráfica em tempo real permite maior flexibilidade ao programador para tratar os dados dentro do *pipeline*, e criar novos efeitos, aumentando a qualidade visual das cenas, proporcionando uma experiência de imersão com muito mais realismo.

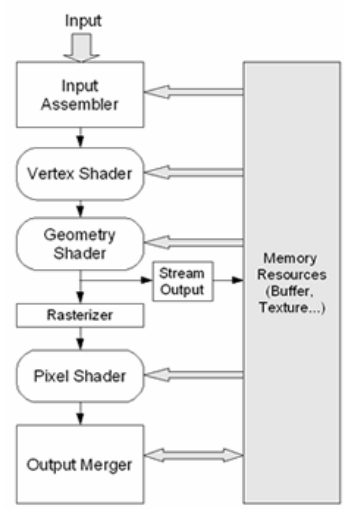


Figura 2.1: *Pipeline* gráfico. Fonte: (Tortelli and Walter 2007)

As GPUs (*Graphics Processing Units*) mais modernas possuem três tipos de processadores: de vértices, de geometria e de fragmentos (Coutinho *et al.* 2007), um para cada tipo de *shader*. O processador de vértices recebe como entrada um vértice juntamente com seus atributos: normalmente a posição, cor, coordenada de texturas e a sua normal. Para cada um destes vértices, o processador executa uma determinada seqüência de instruções, que consiste no *vertex shader*. Este pequeno programa fará alterações nos parâmetros do vértice, de acordo com a sua lógica, possibilitando criar efeitos complexos em tempo real (Clua 2004). Ao terminar de processar o *vertex shader*, o vértice é encaminhado para o restante do *pipeline* gráfico, no caso, para o *geometry shader* que é responsável por especificar como as primitivas serão processadas, além de permitir criar novas primitivas. Já o *pixel shader* consiste em um conjunto de instruções que especificam como os

pixels gerados são processados.

No *pipeline* gráfico, o estágio de rasterização processa um fragmento individualmente, calculando sua cor através de uma interpolação da cor dos vértices do polígono a que pertence, bem como das coordenadas da sua textura. A saída de um programa deste tipo consiste numa cor. Tal recurso permite que alguns efeitos de iluminação, antes impossíveis para visualização em tempo real, possam ser eficientemente implementados (Clua 2004).

Para gerar uma cena é necessário configurar as variáveis de estado que definem os atributos gráficos dos objetos que a compõem. Portanto, o estado define as características de um objeto: material, textura, iluminação, etc. Por exemplo, pode-se aplicar uma iluminação sobre os objetos de uma cena. Desta forma, diz-se que o estado de iluminação está "ligado". Embora essa afirmação possa não aparentar nenhuma complexidade, quando se determina que o *hardware* gráfico deva utilizar iluminação, este é configurado para realizar uma série de operações especiais, podendo ser bastante custoso quanto à utilização de recursos de máquina. O *pipeline* gráfico é projetado para realizar uma tarefa da melhor forma possível, de acordo com a sua configuração. Ao se alterar um estado talvez seja necessário interromper o *pipeline* para que este se ajuste aos novos parâmetros. É possível perceber que, dependendo do estado que for alterado, o *pipeline* pode ter que descartar todo o trabalho feito anteriormente, para poder se reconfigurar e retomar o trabalho com a nova configuração. Um exemplo de um estado que poderia causar esse efeito é o de iluminação.

Além da iluminação, existem vários outros tipos de estados, como configurações de materiais (textura, cores) e configuração do sistema de coordenadas.

O *pipeline* envolve a realização de inúmeras operações que demandam por muito esforço de processamento e em razão disso é necessária a busca por técnicas que minimizem o esforço computacional evitando cálculos desnecessários e otimizando o processamento para a visualização da cena.

Entre as diversas técnicas propostas para realização de otimização do *rendering* podemos destacar o descarte (*culling*) e nível de detalhe (LOD sigla do inglês *Level of Detail*), as quais serão detalhadas nas seções seguintes.

2.1.1 Descarte

Descarte é o processo de eliminação de dados que contribuem pouco para a imagem final. Esses dados incluem objetos que não podem ser vistos pelo observador por alguns motivos. O descarte pode ser feito em diversas etapas no processo de *rendering*. Três tipos interessantes de descarte são o descarte das faces de trás (*backface culling*), descarte por volume de visualização (*frustum culling*) e o descarte por oclusão (*occlusion cullling*).

Descarte das Faces de Trás

Este tipo de descarte é feito no espaço dos objetos, analisando os polígonos. Um polígono possui dois lados: o lado da frente e o lado de trás. Normalmente, o observador visualiza apenas um desses lados, o lado da frente. Assim, polígonos que seriam visualizados por trás podem ser descartados. Desta forma, ao utilizar o descarte das faces de trás, a complexidade da cena é reduzida, já que o lado dos polígonos voltado para trás não serão processados.

Existem algumas maneiras de se determinar o descarte de faces. Uma das mais aceitas consiste em calcular o ângulo formado entre o vetor normal ao plano da face do polígono e o vetor partindo da câmera em direção a este mesmo plano, se o ângulo formado estiver entre 0º e 90º esta face será visível, no caso desse ângulo ser maior, a face não será visível, sendo descartada do processo de *rendering*. A Figura 2.2 mostra esse procedimento.

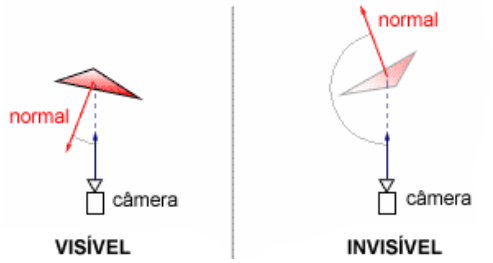


Figura 2.2: Esquema do descarte de faces de trás.

Descarte por Volume de Visualização

Ao contrário do descarte das faces de trás, o descarte por volume de visualização pode eliminar objetos inteiros (compostos por vários polígonos). O processo baseia-se em comparar se os objetos são visíveis, ou seja, se estão dentro do volume de visualização conhecido como *frustum*. O *frustum* é um volume tridimensional (volume de visualização), relacionado com uma determinada projeção (perspectiva ou ortográfica). Em uma projeção em perspectiva, esse volume corresponde a uma pirâmide imaginária limitada pelos planos delimitadores *near* e *far* (plano próximo e distante.), que correspondem a distâncias relativas ao observador, na direção de sua linha de visão. Este esquema é ilustrado na Figura 2.3.

Normalmente, os objetos possuem geometrias arbitrárias e complexas. Desta forma, para se decidir se um objeto é visível ou não, são utilizadas formas geométricas mais simples para aproximar o volume ocupado pelo objeto. O objetivo de se utilizar formas geométricas mais simples é padronizar e facilitar o cálculo da interseção. Essas formas geométricas, que são conhecidas como volumes envolventes (*bounding volumes*), são usualmente de dois tipos: esferas (esfera envolvente, menor esfera que envolve o objeto) e paralelepípedos (caixa envolvente, menor paralelepípedo que envolve o modelo).

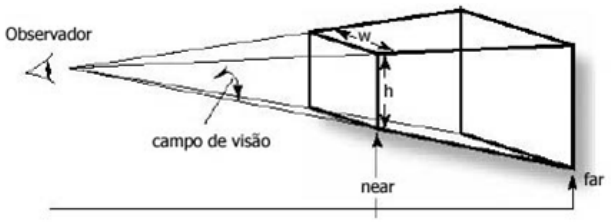


Figura 2.3: Volume de visão. Fonte: (Valente 2004)

A Figura 2.4 ilustra um esquema para o descarte por volume de visualização, Todos os objetos que se encontram no interior do volume são visíveis.

A vantagem de se usar o *frustum culling* é que uma grande quantidade de dados (polígonos) pode ser excluída do processo de renderização, antes de serem enviados ao *hardware*. Se um objeto está totalmente incluído no *frustum*, ele é enviado para o *hardware*. Se um objeto está parcialmente dentro do *frustum*, diversas estratégias podem ser aplicadas. Os objetos que se encontram fora do

frustum são descartados imediatamente.

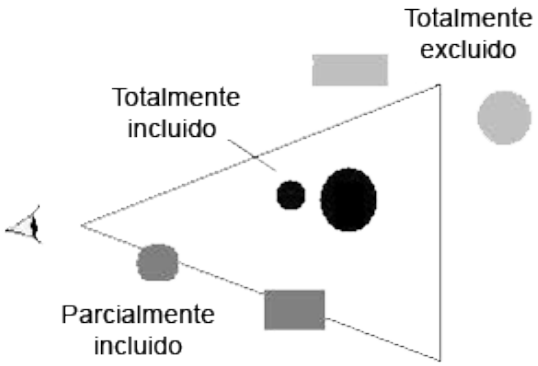


Figura 2.4: Esquema do descarte por volume de visualização. Fonte: (Valente 2004)

Descarte por Oclusão

É uma variação mais sofisticada de descarte onde os objetos que são encobertos por outros são eliminados do processamento gráfico (Figura 2.5). Como exemplo, um observador que está localizado em frente a um prédio. Atrás deste, existem diversos objetos complexos. Nesse exemplo, uma grande quantidade de processamento computacional pode ser poupada ao se descartar inicialmente todos os objetos que se encontram por trás do prédio, já que estes não podem ser visualizados de maneira nenhuma.

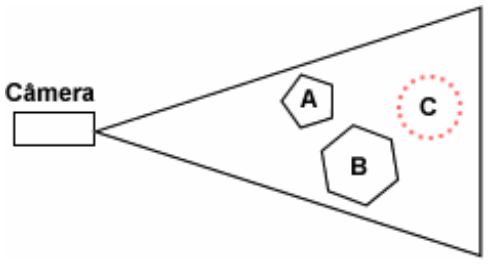


Figura 2.5: Esquema do descarte por oclusão

Como pode ser visto na Figura 2.5, o objeto C está totalmente encoberto pelos outros dois, pois ao se traçar qualquer segmento de reta partindo da câmera nenhum alcançará o objeto C. Portanto, este objeto será descartado do processo de visualização.

2.1.2 Nível de Detalhe

Em computação gráfica, nível de detalhe (LOD) significa a diminuição da complexidade da representação de um objeto 3D à medida que o observador se afasta do mesmo, de acordo com alguma métrica, como o tamanho do objeto (Funkhouser *et al.* 1992), velocidade de deslocamento do observador ou posição (Watson *et al.* 1996). Os objetos mais distantes são substituídos por versões mais simples. Com isso, as técnicas de LOD aumentam a eficiência do *rendering*, diminuindo a carga de trabalho nas fases do processamento gráfico. A qualidade visual reduzida do modelo é muitas vezes imperceptível quando estão distantes ou em movimento rápido. A Figura 2.6 mostra a representação do mesmo objeto com vários níveis de detalhes, nesse caso, a representação mais detalhada tem uma quantidade bem maior de polígonos.

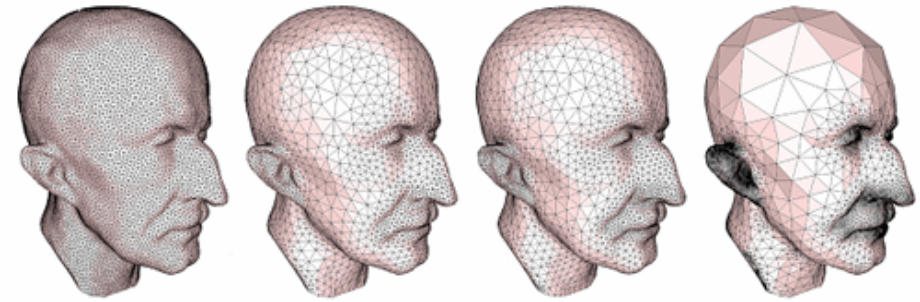


Figura 2.6: Exemplo da utilização de LOD. Fonte: (Alliez *et al.* 2002)

Embora as técnicas de LOD sejam aplicadas em geral apenas a geometrias bem detalhadas, o conceito básico pode ser generalizado.

2.1.3 Grafo de Cena

Em computação gráfica, uma "cena"se refere a uma coleção de objetos, seus relacionamentos, e a definição da forma como o observador vê esses objetos na

cena. Em matemática, um grafo discreto representa um sistema construído a partir de duas entidades fundamentais: nós e arestas. Ambas as definições nos permitem explicar o que é um grafo de cena. Um grafo de cena é um grafo discreto, dirigido, acíclico, no qual não há nenhuma ligação de um nó a ele mesmo, e nem um caminho até alguns de seus predecessores no grafo (Tenenbaum *et al.* 1995). Geralmente, um grafo de cena é uma organização hierárquica de formas, grupos de formas, e grupos de grupos que coletivamente definem o conteúdo de uma cena. O grafo de cena permite ordenar dados em forma hierárquica onde nós pai afetam seus nós filho. Ele pode ter quantos filhos quiser, assim como uma árvore. Mas não se trata de apenas uma árvore, ele representa também alguma ação a ter lugar antes de proceder aos objetos filho. As formas e as sub-árvores podem ser compartilhadas entre múltiplos grupos (Sola *et al.* 2005).

Computacionalmente falando, um grafo de cena é uma estrutura de dados que permite a organização hierárquica de objetos que constituem uma cena (Zeev 2003). Por exemplo, quando se representa um veículo com quatro rodas, é desejável que a alteração da posição ou da orientação do carro seja transmitida para as rodas. Outra relação explorada no grafo de cena é a hierarquia de volumes envolventes, onde objetos próximos são agrupados para que, durante a etapa de descarte, sejam eliminados com apenas um teste feito no topo da hierarquia, evitando a necessidade de visitar cada um de seus filhos.

Um grafo de cena é uma estrutura de dados que utiliza uma abordagem de alto nível para modelagem e gestão de cenas, ao contrário das interfaces de programação de aplicativos (API) gráficas básicas (OpenGL e Direct3D). Os usuários desse tipo de estrutura de dados não necessariamente precisam conhecer os detalhes de implementação de baixo nível para utilizá-la.

Um dos conceitos centrais a respeito de um grafo de cena é que este implementa uma estrutura hierárquica (Valente 2004). Em um grafo de cena todos os nós são ligados a um nó raiz, que é usado para definir o começo dos dados da cena.

Para gerar a imagem da cena, o grafo é percorrido a partir da raiz. Alteração de dados ou estrutura do grafo, assim como a especificação de algoritmos de travessia diversos, pode produzir imagens diferentes ao final do processo.

Uma diferença entre grafos de cena e grafos matemáticos é que os grafos de cena são heterogêneos, isto é, os nós têm tipos diferentes. Como pode ser visto

na Figura 2.7, em um grafo de cena, a distinção principal está entre os nós de grupo e os nós folha. As folhas carregam a geometria visualizável, os demais nós do grafo são estruturados em grupos lógicos (Reiners 2002).

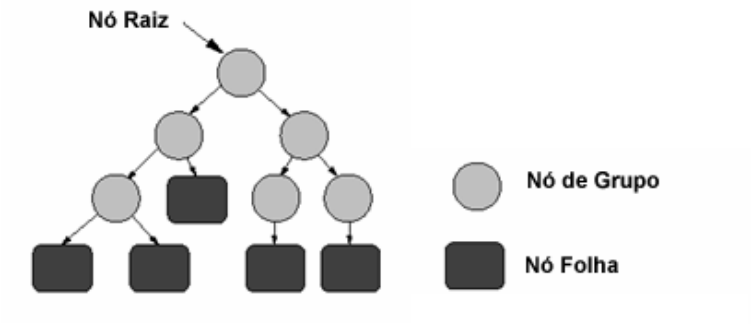


Figura 2.7: Estrutura de um grafo de cena. Fonte: (Reiners 2002)

2.1.4 Sistema de Informação Geográfica

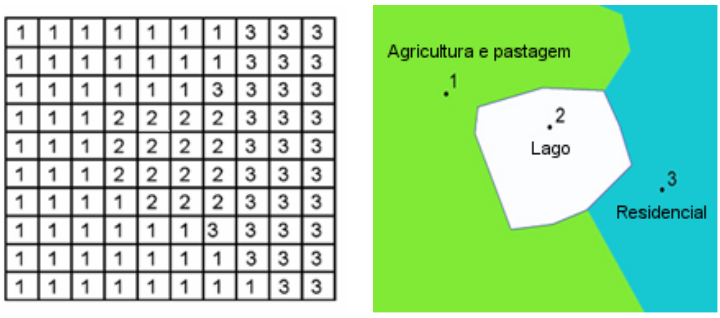
Um Sistema de Informação Geográfica (SIG) é um sistema computacional capaz de capturar, armazenar, analisar, e exibir informações referenciadas geograficamente; isto é, dados identificados de acordo com a localização sobre a superfície terrestre (USDI 2007). Um SIG também pode ser definido como um sistema de *hardware*, *software*, informação espacial e procedimentos computacionais, que permitem e facilita a análise, gestão ou representação do espaço e dos fenômenos que nele ocorrem.

Essa tecnologia pode ser usada para investigações científicas, estudos de impacto ambiental ou de prospecção de recursos, e auxílio ao planejamento de desenvolvimento urbano. Sendo de grande importância por constituir um sistema espacial de apoio a decisão, que disponibiliza informações referentes à localização; ou à análise de rotas, onde é possível o cálculo de caminhos ótimos entre dois ou mais pontos; ou à geração de modelos explicativos a partir do comportamento observado de fenômenos espaciais; ou até a utilização em material jornalístico, onde pode ser usado para aprofundar coberturas jornalísticas onde a espacialização é importante.

Existem vários modelos de dados aplicáveis em SIG. Por exemplo, o SIG pode

funcionar como uma base de dados com informação geográfica que se encontra associada aos objetos gráficos de um mapa digital. Desta forma, selecionando-se uma área pode-se saber o valor dos seus atributos geográficos, e inversamente, selecionando um registro da base de dados é possível saber a sua localização e apontá-la num mapa.

Os modelos de dados mais comuns em SIG são o modelo *raster* ou matricial e o modelo vetorial. O modelo de SIG matricial centra-se nas propriedades do espaço, fragmentando-o em células regulares (habitualmente quadradas, mas podendo ser retangulares, triangulares ou hexagonais). Cada célula representa um único valor (Figura 2.8a). Quanto maior for a dimensão de cada célula (resolução) menor é a precisão ou detalhe na representação do espaço geográfico. No caso do modelo de SIG vetorial (Figura 2.8b), o foco das representações centra-se na precisão da localização dos elementos no espaço. Para modelar digitalmente as entidades do mundo real utilizam-se essencialmente três formas espaciais: o ponto, a linha e área (ou polígono). As estruturas vetoriais são utilizadas para representar as coordenadas das fronteiras de cada entidade geográfica (USDI 2007).



(*a*) (*b*)

Figura 2.8: Exemplo de estruturas de arquivos. (a) modelo matricial. (b) modelo vetorial. Fonte: (USDI 2007)

Na tentativa de chegar a uma padronização dos citados tipos de dados, foi criado o Open Geospatial Consortium (OGC 2007), que é um consórcio internacional com mais de 250 companhias, agências, universidades participando no desenvolvimento de soluções conceituais disponíveis publicamente que podem ser úteis a todos os tipos de aplicações que gerenciam dados espaciais. O objetivo é incentivar os desenvolvedores de *software* de SIG e Geoprocessamento adotarem as

especificações OpenGIS (OGC 2007), onde se encontram os padrões que permitem a interoperabilidade entre as tecnologias de geoprocessamento.

2.1.5 Banco de Dados Geográficos

A pesquisa na área de Banco de Dados, já há algum tempo, passou a preocupar-se com o suporte a aplicações não convencionais (Schneider 1997). Uma aplicação é classificada como não convencional quando trabalha com outros tipos de dados, além dos tradicionais (inteiros, reais, caracteres), como tipos de dados espaciais, temporais e espaço-temporais.

Ao longo dos anos, as implementações de SIGs colaboraram para o surgimento de diferentes arquiteturas, distinguindo-se principalmente pela estratégia adotada para o armazenamento e recuperação de dados espaciais. Mais recentemente, tais arquiteturas evoluíram para utilizar, cada vez mais, recursos de Sistemas de Gerenciamento de Banco de Dados (SGBD) (Casanova *et al.* 2005), surgindo com isso as extensões espaciais dos principais SGBDs disponíveis no mercado. Com isso, torna-se comum a perspectiva da construção de SIG onde tanto os atributos como as geometrias de dados espaciais sejam gerenciados pelo SGBD.

Existem basicamente duas formas principais de integração entre os SIGs e os SGBDs, que são a arquitetura dual e a arquitetura integrada (Casanova *et al.* 2005). A arquitetura dual armazena as componentes espaciais dos objetos separadamente. A componente convencional, ou alfanumérica, é armazenada em um SGBD relacional e a componente espacial é armazenada em arquivos com formato proprietário, o que ocasiona dificuldade no controle e manipulação das componentes espaciais, e também dificuldades de interoperabilidade, já que cada sistema trabalha com arquivos em formatos proprietários.

A arquitetura dominante atualmente é a integrada, que consiste em armazenar todos os dados num SGBD, ou seja, tanto a componente espacial quanto a alfanumérica. Sua principal vantagem é a utilização dos recursos de um SGBD para controle e manipulação de objetos espaciais, como gerência de transações, controle de integridade, concorrência e linguagens próprias de consulta. Com isso, a manutenção de integridade entre a componente espacial e alfanumérica é feita pelo SGBD.

Esta arquitetura pode ainda ser subdividida em três abordagens: baseada em

campos longos, em extensões espaciais e combinada.

A arquitetura integrada baseada em campos longos utiliza BLOBs (*Binary Large OBject*) para armazenar a componente espacial dos objetos. Nesse caso, o SGBD trata um BLOB como uma cadeia de bits sem nenhuma semântica adicional. Portanto, ao codificar dados espaciais em BLOBs, esta arquitetura torna a sua semântica opaca para o SGBD. Ou seja, passa a ser responsabilidade do SIG implementar os operadores espaciais, capturando a semântica dos dados, e métodos de acesso que possam ser úteis no processamento de consultas, embora seja bastante difícil incorporá-los ao sistema de forma eficiente.

A arquitetura integrada com extensões espaciais consiste em utilizar extensões espaciais desenvolvidas sobre um SGBD Objeto Relacional (SGBD-OR). Esta arquitetura oferece algumas vantagens, entre as quais podemos destacar:

- Permite definir tipos de dados espaciais, equipados com operadores específicos (operadores topológicos e métricos);
- Permite definir métodos de acesso específicos para dados espaciais.

Exemplos desta arquitetura são o Oracle Spatial (Murray 2003), PostGIS (PostGIS 2007) e a extensão espacial do SGBD MySQL (MySQL 2006). Embora largamente baseados nas especificações do OpenGIS, estas implementações possuem variações relevantes entre os modelos de dados, semântica dos operadores espaciais e mecanismos de indexação.

As extensões espaciais utilizadas nas arquiteturas integradas possuem as seguintes características:

- Fornecem tipos de dados espaciais (TDE) em seu modelo de dados, e mecanismos para manipulá-los;
- Estendem a linguagem SQL para incluir operações sobre TDEs, transformando-a de fato em uma linguagem para consultas espaciais;
- Adaptam outras funções de nível mais interno ao SGBD para manipular TDEs eficientemente, tais como métodos de armazenamento e acesso, e métodos de otimização de consultas.

No caso de aplicativos SIG que manipulam objetos com geometrias tanto matriciais quanto vetoriais, é possível a utilização de uma arquitetura integrada combinada, formada pela combinação da arquitetura baseada em campos longos e, em extensões espaciais. Ou seja, as geometrias vetoriais são armazenadas utilizando-se os recursos oferecidos pelas extensões e as geometrias matriciais são armazenadas em BLOBs.

As atuais extensões espaciais possibilitam o melhor aproveitamento do SGBD com dados deste tipo, pois quando se utiliza extensão espacial pode-se trabalhar com tipos de dados espaciais definidos por elas, tais como ponto, linha e polígono. Estas extensões permitem que tais dados sejam manipulados como qualquer outro tipo de dado de SGBD (Campos *et al.* 2007). Além desta característica, há a extensão da linguagem SQL ofertando operações e funções para consultar relações espaciais.

Nas seções a seguir serão detalhados conceitos inerentes aos SGBDs com extensões espaciais, tais como: tipo de dados espaciais, indexação espacial, operações espaciais, funções espaciais, predicados espaciais, métricas espaciais e funções construtoras.

Tipos de Dados Espaciais

Tipo de dados espacial permite que o banco de dados reconheça pontos, linhas e polígonos como objetos espaciais dentro da base de dados. O banco de dados nativamente tem a capacidade de reconhecer tipos de dados, tais como reais, inteiros, e caracteres. No entanto, um novo tipo deve ser incluído a fim de que o banco de dados possa compreender os objetos espaciais (Weinberger 2002). O Open GIS Consortium (OGC) definiu uma estrutura para o tipo de dados espaciais. Este foi definido como um tipo *Geometry*, como visto no modelo simplificado na Figura 2.9.

O tipo *Geometry* é composto por ponto, linha, área e outros que podem ser combinados para representar praticamente qualquer objeto geométrico (MySQL 2006). Neste modelo, cada objeto geométrico tem as seguintes propriedades gerais:

- É associado com um Sistema de Referência Espacial, que descreve a coordenada espacial, na qual o objeto é definido;

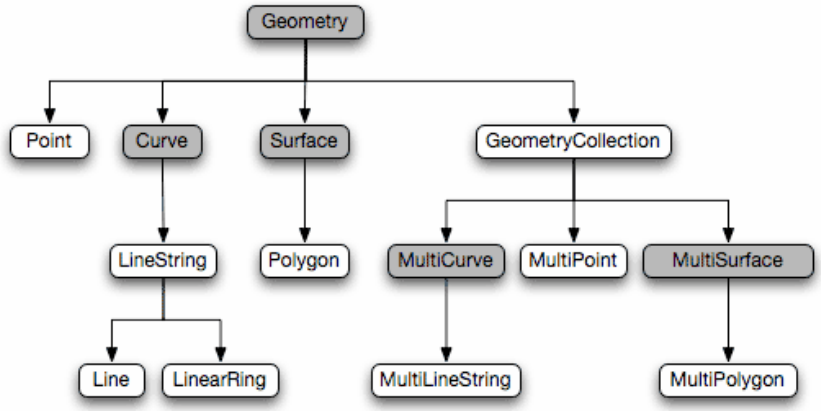


Figura 2.9: Estrutura do tipo de dados espacial proposta pelo OGC. Fonte: (OGC 2007)

- Pertence a alguma classe geométrica.

Algumas destas classes são abstratas (não-instanciável). Isto é, não é possível criar um objeto desta classe. Outras classes são instanciáveis e os seus objetos podem ser criados. Cada classe tem propriedades e podem ter declarações (regras que definem instâncias de classes válidas).

A classe abstrata *Geometry* é a base da hierarquia, sendo que as subclasses instanciáveis de *Geometry* são restritas a objetos geométricos com zero, uma ou duas dimensões que existem no espaço de coordenadas bidimensional. Em *Geometry* está presente um conjunto de propriedades que são comuns às suas subclasses.

Um objeto da classe *geometry* terá as seguintes propriedades:

- Tipo: cada geometria pertence a uma das classes instanciáveis na hierarquia.
- Identificador de Referência Espacial (SRID): este valor identifica o Sistema de Referência Espacial associada da geometria, o qual descreve a coordenada espacial onde o objeto geométrico está definido.
- Coordenadas em seu Sistema de Referência Espacial: todas as geometrias não-vazias incluem pelo menos um par de coordenadas (X,Y). Coordenadas estão relacionadas ao SRID. Por exemplo, em sistemas de coordenadas

diferentes, a distância entre dois objetos podem diferir mesmo quando os objetos têm as mesmas coordenadas, porque as distâncias no sistema de coordenadas planar e a distância no sistema geocêntrico (coordenadas na superfície da Terra) são coisas diferentes.

- Interior, limite e exterior: todas as geometrias ocupam alguma porção no espaço. O exterior de uma geometria é todo espaço não ocupado pela geometria. O interior é o espaço ocupado pela geometria. O limite é a interface entre o interior e o exterior

- Mínimo Retângulo Envolvente (MBR do inglês *Minimum Bounding Rectangle*) da geometria: isto é, a geometria delimitadora, formada pelas coordenadas de mínimo e máximo (X,Y):

- Propriedade fechado ou não-fechado: valores geométricos de alguns tipos (*LineString*, *MultiString*) podem ser fechado ou não-fechado. Cada tipo determina a sua própria afirmação de ser fechado ou não-fechado.

- Propriedade vazia ou não-vazia: uma geometria é vazia se ela não tem nenhum ponto. Exterior, interior e limite de uma geometria vazia não estão definidos (são representados por valores NULL).

- Dimensão: uma geometria pode ter uma dimensão de -1, 0, 1 ou 2:

  - -1 usado para geometrias vazias
  - 0 usado para geometrias sem tamanho e sem área.
  - 1 usado para geometrias com tamanho diferente de zero e sem área.
  - 2 usado para geometrias com área diferente de zero.

A classe base *Geometry* tem as subclasses: *Point*, *Curve*, *Surface* e *GeometryCollection*:

- *Point* representam objetos sem dimensão.

- *Curve* representam para objetos de uma dimensão, e tem a subclasse *LineString*, com subclasses *Line* e *LinearRing*.

- *Surface* é criado para objetos bidimensionais e tem a subclasse *Polygon*.
- *GeometryCollection* tem classes de coleção com zero, uma e duas-dimensões chamadas *MultiPoint*, *MultiLineString* e *MultiPolygon* para modelagem geométrica correspondente a coleções de *Points*, *LineStrings* e *Polygons* respectivamente. *MultiCurve* e *MultiSurface* são introduzidas como superclasses abastratas que generalizam a interface de coleção para tratar *Curves* e *Surfaces*.

Geometry, Curve, Surface, MultiCurve e *MultiSurface* são definidos como classes não instanciáveis. Eles definem em conjunto de métodos comuns para suas subclasses incluídos por razões de extensebilidade.

Point, LineString, Polygon, GeometryCollection, MultiPoint, MultiLineString, MultiPolygon são classes instanciáveis.

Indexação Espacial

Outro importante componente das extensões espaciais é o índice espacial. Praticamente todos os bancos de dados incluem sistemas de indexação para permitir a rápida localização e recuperação dos dados. No entanto, esses sistemas são projetados para trabalhar com os tipos de dados nativos. Essas estruturas tradicionais de dados de apenas uma dimensão como tabelas *hash* (estrutura de dados especial, que associa chaves de pesquisa (*hash*) a valores) ou árvores-B (B-trees) não funcionam para dados de mais de uma dimensão, pois essas estruturas trabalham com comparações de igualdade entre variáveis enquanto aplicações espaciais necessitam realizar comparações entre intervalos (Oliveira *et al.* 2007). Índices espaciais são estruturas de dados que permitem promover acesso aos dados de mais de uma dimensão de maneira eficiente. Em um esforço para permitir que se recuperem rapidamente dados espaciais, foram desenvolvidas diversas estruturas de dados com o propósito de indexar dados de mais de uma dimensão, entre elas estão: Quadtrees, R-trees, hB-trees, TV-trees, SS-trees (Kothuri *et al.* 2002). Recentemente, tem havido um movimento de migração na utilização de outros tipos de estruturas para R-trees como o mecanismo preferido de indexação de dados espaciais nas ferramentas disponíveis no mercado (Weinberger 2002).

Os dados espaciais devem ser indexados para que se possa utilizar os operadores e funções espaciais sobre os campos do tipo geográficos com um melhor tempo de resposta. Há casos, em que para utilizar operadores específicos é obrigada a indexação espacial da tabela a ser utilizada.

Operações Espaciais

Tipos de dados espaciais e indexação espacial permitem que os dados sejam armazenados e recuperados a partir de dados relacionais. No entanto, isto é inútil, a menos que você tenha um modo de manipular os dados. A manipulação desses dados ocorre através da utilização de operadores espaciais. Os operadores espaciais dividem-se em várias categorias, entre os quais se incluem funções espaciais, predicados, medição e construtor. O OGC define todas estas funções, mas nem todas são implementadas nas ferramentas e alguns SGBD espaciais oferecem mais recursos do que outros. Todas essas funções podem ser acessadas usando comandos SQL.

As funções espaciais permitem aos usuários executar operações que geram novas geometrias, como a criação de um *buffer* em torno de um objeto (gera uma nova geometria a partir de uma distância de um objeto específico). Funções espaciais devem ter uma entrada, que será utilizada para efetuar a operação, e em seguida disponibilizar como resultado uma nova geometria.

Os predicados espaciais são desenvolvidos a partir de relacionamentos espaciais entre feições geográficas, o que permite aos usuários consultas ao banco de dados que retornam como resposta o resultado verdadeiro ou falso. Os predicados espaciais podem representar os relacionamentos topológicos *disjoint, intersects, inside, contains, touches, covers, covered-by, equal* e *crosses*.

Já as métricas espaciais fazem exatamente o que o nome sugere: cálculo de área, comprimento ou perímetro e distância entre geometrias. Uma medição espacial pode retornar a área ou comprimento de uma dada geometria.

As funções construtores permitem aos usuários criar novos objetos espaciais usando comandos SQL. Fornecendo-se as coordenadas X, Y um ponto pode ser criado. Um conjunto de pontos pode servir como entrada para criar uma linha ou polígono.

2.2 Tecnologias Envolvidas

Nessa seção, são apresentadas as tecnologias envolvidas no desenvolvimento do modelo de arquitetura e na construção do protótipo. Toda a implementação deste trabalho foi feita em C++ em conjunto com a biblioteca gráfica OpenSceneGraph (Osfield and Burns 2006), sendo utilizado para o armazenamento e gerenciamento da base de dados a extensão espacial do SGBD Oracle (Oracle 2005).

2.2.1 OpenSceneGraph

O OpenSceneGraph (OSG) é uma biblioteca gráfica *open source*, usada por desenvolvedores de aplicações nos campos tais como simulação visual, jogos eletrônicos, realidade virtual, visualização científica e modelagem. Escrita inteiramente em *Standard* C++ e OpenGL, sendo independente de plataforma, podendo rodar sobre as plataformas Windows, OSX, o GNU / Linux, IRIX, Solaris, HP-Ux, AIX FreeBSD (Osfield and Burns 2006).

No desenvolvimento desse trabalho, decidiu-se utilizar o OSG por ser um *toolkit* de alta performance e por oferecer um pacote completo com grande parte do que se pode esperar de um gerenciador de grafo de cena. Já traz implementadas várias otimizações, que permitem boa representação de ambiente virtual 3D e renderização eficiente (Sola *et al.* 2005), fundamentais quando se objetiva bom desempenho em modelos grandes.

A construção de aplicações gráficas com o OSG possibilita ao desenvolvedor concentrar-se em rotinas de nível mais alto, já que a biblioteca livra o desenvolvedor de implementar e otimizar chamadas gráficas de baixo nível, com isso, agiliza o desenvolvimento dessas aplicações e poupa o desenvolvedor de programar rotinas complexas de computação gráfica 3D e renderização (Sola *et al.* 2005).

Dentre as características do OSG, podem-se listar algumas das mais interessantes para o desenvolvimento de aplicações gráficas:

Maximização do desempenho: O OSG é uma biblioteca que maximiza o desempenho durante a visualização. Para isso são utilizadas duas técnicas: *culling* e propriedades de estado. A técnica *culling* é aplicada quando se deseja excluir do processamento partes geométricas de objetos 3D que não são visualizados naquele

momento, economizando tempo e capacidade de processamento da CPU e da placa gráfica. A técnica de propriedades de estado se aplica quando diversos objetos possuem características iguais, tais como luzes, texturas e materiais. Nesse caso, as propriedades iguais são armazenadas em um único nó e compartilhadas para todos os objetos 3D que usam essas propriedades em um ambiente virtual. Isso auxilia a economizar memória relativa às estruturas de dados necessárias para representar o ambiente virtual em uso.

Melhoria da Produtividade: O OSG auxilia a reduzir o tempo e o trabalho requerido para programar aplicações gráficas de alto desempenho. Ele esconde a complexidade do gerenciamento de todas as partes gráficas, reduzindo a quantidade de linhas de código e chamadas para a biblioteca gráfica de baixo nível, OpenGL. Por ter uma estrutura baseada em orientação a objetos, o OSG pode ser considerado como uma biblioteca flexível que permite aumentar o reuso de código. Assim, auxilia o desenvolvedor a criar aplicações com facilidade diminuindo o tempo de análise e desenvolvimento e, conseqüentemente, a manutenção do código dessas aplicações.

Portabilidade: O OSG encapsula muitos detalhes relativos a rotinas gráficas de baixo nível, processamento gráfico, e a leitura e escrita de dados, coibindo a necessidade de criar código específico para uma plataforma. Sendo assim, para portar a aplicação em outras plataformas é necessário apenas recompilar o código para outra plataforma.

Além da infra-estrutura para gerenciamento dos dados, vantagens adicionais são oferecidas na forma de gerenciamento de detalhes e problemas. Uma cena complexa em três dimensões contém muitos elementos diferentes, incluindo fontes de luz, um modelo de câmera usado para visualizar a cena, os objetos na cena, entre outros. Detalhes adicionais a serem gerenciados incluem os planos de *clipping*, o mecanismo para otimizar a utilização dos *buffer*s de cor e profundidade em cada *frame*, os controles dos *viewport*. Manter um seguimento adequado de toda esta informação, além de prover *defaults* razoáveis, pode ser amplamente simplificado por um bom sistema de grafos de cena (Sola *et al.* 2005). Estas características foram importantes na decisão pela utilização do OSG para o desenvolvimento da arquitetura proposta neste trabalho.

2.2.2 Oracle Spatial

O conhecido SGBD Oracle possui a extensão desenvolvida sobre o modelo objeto-relacional para ser empregado do desenvolvimento de SIG. O Oracle Spatial é uma extensão espacial, que é baseada nas especificações do OpenGIS. Esta extensão contém um conjunto de funcionalidades e procedimentos que permitem armazenar, acessar, modificar e consultar dados espaciais em um banco de dados Oracle.

O Oracle Spatial é formado pelos seguintes componentes (Oracle 2005):

- Um modelo próprio de dados chamado MDSYS que define a forma de armazenamento, a sintaxe e semântica dos tipos espaciais suportados;
- Mecanismo de indexação espacial;
- Um conjunto de operadores e funções para representar consultas, junção espacial e outras operações de análise espacial;
- Aplicativos administrativos.

A seguir será detalhado o modelo conceitual do Oracle Spatial, além disso, como é a sua representação de geometrias, o seu objeto espacial para a representação de dados geográficos, o seu conjunto de operações implementadas, a sua semântica das operações e os seus métodos de indexação.

Modelo Conceitual

O modelo de dados do Oracle Spatial consiste em uma estrutura hierárquica de elementos, geometrias e camadas de informação (*layers*). Cada camada é formada por uma coleção de geometrias que possuem um mesmo conjunto de atributos, que por sua vez são formadas por um conjunto de elementos (Sharma 2001). Cada elemento é associado a um tipo espacial primitivo, como ponto, linha ou polígono (*Point, LineString* ou *Polygon*). Uma geometria pode ser formada por um único elemento ou por um conjunto homogêneo (*MultiPoint, MultiLinesString* ou *MultiPolygon*) ou heterogêneo (*Collection*) de elementos.

Um elemento é o componente básico de construção das geometrias do Oracle Spatial. Eles são os tipos primitivos de dados suportados pelo Oracle e que serão apresentados adiante. Os elementos são construídos utilizando coordenadas,

podendo ter um ou vários pares de coordenadas dependendo do elemento (Silva 2002).

Uma geometria é uma representação de um objeto espacial, modelada com um conjunto ordenado de elementos primitivos (homogêneos ou heterogêneos). Cada geometria possui um identificador, conhecido como *Numeric Geometry Identifier* (GID), que é único e que associa a geometria com o correspondente conjunto de atributos (Silva 2002).

Um *layer* é uma coleção heterogênea de geometrias que compartilham o mesmo conjunto de atributos (Silva 2002).

Representação da Geometria

O Oracle Spatial suporta três tipos primitivos de geometrias (ponto, linha ou polígono) e uma coleção de outras geometrias (Figura 2.10) formadas a partir desses tipos primitivos, tais como: arcos circulares, círculos, linhas compostas e polígonos compostos.

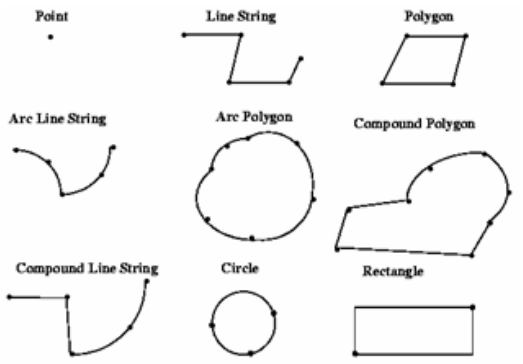


Figura 2.10: Tipos de dados espaciais do Oracle Spatial. Fonte: (Oracle 2005)

Os tipos espaciais bidimensionais são compostos por pontos formados por duas ordenadas X e Y. A extensão também suporta o armazenamento e indexação de tipos tridimensionais e tetradimensionais, mas as funções e operadores só funcionam para os tipos bidimensionais.

Os pontos são elementos compostos de duas coordenadas, X e Y, freqüentemente correspondentes à longitude e latitude. Linhas são compostas de um ou mais pares de pontos que definem o segmento de linha. Polígonos são

compostos de linhas conectadas que formam um anel fechado, cujo interior do polígono é implícito. Polígonos podem conter "buracos"que são construídos, por definição, internos aos polígonos. Neste caso, o anel exterior e o anel interior do polígono são considerados como dois elementos distintos que juntos formam um polígono complexo.

Objeto Espacial

Baseado no modelo objeto-relacional, o Oracle Spatial define um tipo de objeto, para representar dados espaciais a serem manipulados, chamado SDO_GEOMETRY. Este objeto contém a geometria em si, suas coordenadas, e informações sobre seu tipo e projeção. Em uma tabela espacial, os atributos alfanuméricos da geometria são definidos como colunas de tipos básicos (VARCHAR2, NUMBER, DATE, dentre outros), e a geometria como uma coluna espacial (ou *layer*) particular da tabela do tipo MDSYS.SDO_GEOMETRY. Em uma tabela espacial, cada instância do dado espacial é armazenada em uma linha, e o conjunto de todas as instâncias dessa tabela forma uma camada de informação.

Conjunto de Operações Implementadas

O Oracle Spatial fornece um conjunto de operadores espaciais (*spatial operators*) e funções espaciais (*geometry functions*), que são utilizados juntamente com a linguagem SQL, para suportar consultas espaciais. A diferença básica entre essas duas abordagens é que, as funções não utilizam índices nas tabelas espaciais, caso existam. Já para a utilização dos operadores, é necessária obrigatoriamente a existência de índices nas tabelas espaciais e, por conta disso, as consultas efetuadas com operadores são mais eficientes. Outra diferença é que as funções podem ser utilizadas tanto na cláusula SELECT como na cláusula WHERE, enquanto que os operadores só podem ser utilizados na cláusula WHERE.

A Tabela 2.1 apresenta algumas dessas operações com uma breve descrição de suas funcionalidades.

A Tabela 2.2 mostra as funções espaciais implementadas com uma resumida descrição do que fazem.

Tabela 2.1: Operadores Espaciais do Oracle Spatial. Fonte: (Oracle 2005)

Operadores Espaciais	**Descrição**
SDO_NN	Determina os vizinhos mais próximos a uma geometria
SDO_NN_DISTANCE	Determina a que distâncias estão os objetos retornados pelo operador SDO_NN de uma dada geometria
SDO_RELATE	Determina se duas geometrias se interagem de algum modo
SDO_WITHIN_DISTANCE	Determina se uma geometria está a uma dada distância da outra

Semântica das Operações

Para consultar relações topológicas entre duas geometrias é utilizado o operador SDO_RELATE ou a função SDO_GEOM.RELATE. Estes implementam o Modelo de 9-Interseções definido em (Egenhofer and Herring 1991). Este modelo considera as interseções, vazia (0) ou não vazia (1), entre os interiores, fronteiras e exteriores de duas geometrias. O SDO_RELATE e a SDO_GEOM.RELATE recebe como parâmetro o tipo de relação topológica que deve ser computada. Os possíveis parâmetros são: *Equal, Disjoint, Touch, Inside, OverlapBdyIntersect, OverlapBdyDisjoint, Anyinteract, Contains, On, Covers e Coveredby* (Oracle 2005). A seguir são apresentadas as descrições de cada relacionamento topológico:

- EQUAL: dois objetos são iguais quando eles possuem a mesma fronteira e o mesmo interior.
- DISJOINT: dois objetos são disjuntos quando nem o interior nem a fronteira de ambos se interceptam, ou seja, não há relacionamento entre eles.
- TOUCH: dois objetos se tocam quando suas fronteiras se interceptam, mas o interior não. Isto é, suas fronteiras compartilham pelo menos um ponto comum, mas sem que haja nenhum ponto comum a ambos os interiores.
- INSIDE: ocorre quando o primeiro objeto está totalmente dentro do segundo e suas fronteiras não se tocam.
- OVERLAPBDYINTERSECT: dois objetos têm um relacionamento desse

Tabela 2.2: Funções Espaciais do Oracle Spatial. Fonte: (Oracle 2005)

Funções Espaciais	**Descrição**
SDO_GEOM.RELATE	Determina como duas geometrias se interagem
SDO_GEOM.SDO_AREA	Calcula a área de um polígono de duas dimensões
SDO_GEOM.SDO_BUFFER	Gera um *buffer* (nova geometria) ao redor de uma geometria
SDO_GEOM.SDO_DIFFERENCE	Retorna a geometria correspondente à diferença topológica entre duas geometrias
SDO_GEOM_DISTANCE	Calcula a distancia entre duas geometrias
SDO_GEOM.SDO_INTERSECTION	Retorna a geometria correspondente à interseção topológica entre duas geometrias
SDO_GEOM.SDO_LENGTH	Calcula o comprimento ou perímetro de uma geometria
SDO_GEOM.SDO_UNION	Retorna a geometria correspondente à união topológica entre duas geometrias
SDO_GEOM.VALIDATE_GEOMETRY	Determina se uma geometria é válida
SDO_GEOM.VALIDATE_LAYER	Determina se todas as geometrias armazenadas em uma coluna espacial são válidas
SDO_GEOM.WITHIN_DISTANCE	Determina se uma geometria está a uma distância específica (distância Euclidiana) de outra

tipo (*Overlap Boundaries Intersect*) quando a fronteira e o interior de ambos se interceptam. É aplicável quando ambos os objetos são do tipo polígono.

- OVERLAPBDYDISJOINT: dois objetos têm um relacionamento desse tipo (*Overlap Boundaries Disjoint*) quando o interior de um objeto intercepta a fronteira e o interior de outro, mas as duas fronteiras não se interceptam. É aplicado quando o teste e efetuado entre objetos do tipo linha e polígono.
- ANYINTERACT: dois objetos têm algum tipo de interação quando não são disjuntos.
- CONTAINS: ocorre quando o segundo objeto está totalmente dentro do primeiro e suas fronteiras não se tocam.
- ON: ocorre quando o interior e a fronteira do primeiro objeto estão na fronteira de outro objeto (e o segundo objeto abrange o primeiro objeto).
- COVERS: ocorre quando o segundo objeto está totalmente dentro do primeiro e suas fronteiras se tocam em um ou mais pontos.
- COVERBY: ocorre quando o primeiro objeto está totalmente dentro do segundo e suas fronteiras se tocam em um ou mais pontos.

Na Figura 2.11 são exibidas essas relações topológicas.

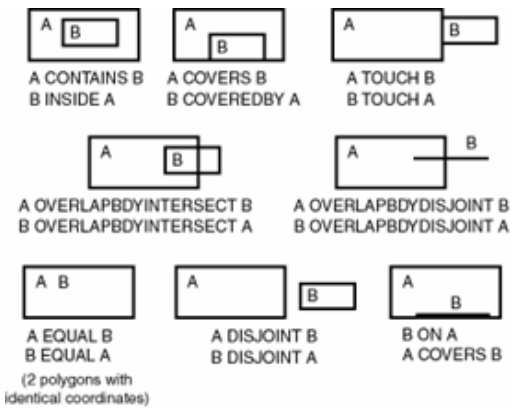


Figura 2.11: (Relações Topológicas implementadas no Oracle Spatial. Fonte: (Oracle 2005)

Métodos de Indexação

O Oracle Spatial dá suporte à criação de índices para dados espaciais, podendo ser de dois tipos: R-tree e Quadtree. Esses índices espaciais podem ser criados com extensões de sintaxe SQL. Cada um desses índices é apropriado para diferentes situações e podem ser usados simultaneamente para indexar uma mesma coluna com geometria. Segundo Sharma (2001), o índice R-tree pode ser utilizado em lugar do Quadtree, ou em conjunto com ele.

As R-trees no Oracle spatial logicamente estão implementadas como árvores e internamente estão implementadas em tabelas do Banco de dados. As buscas envolvem SQL recursivos para se chegar da raiz aos nós da árvore Kothuri (2002). Essa abordagem resulta em buscas mais eficientes devido a uma melhor preservação de aproximações espaciais, mas pode tornar-se lenta para atualizações e para a criação dos índices (Oliveira *et al.* 2007).

Para uma camada de geometrias, um índice R-tree contém um índice hierárquico no mínimo retângulo envolvente (MBR) da geometria na camada, como visto na Figura 2.12.

A Quadtree realiza aproximação das geometrias através de aproximações com quadrantes, que são resultantes da subdivisão recursiva do espaço, também conhecido como tesselamento (Figura 2.13).

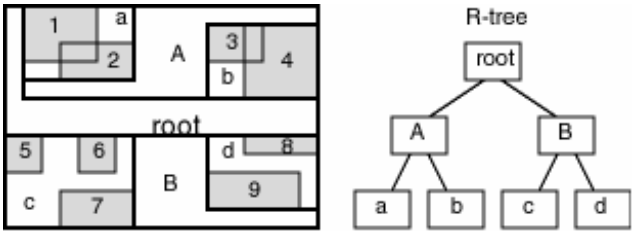


Figura 2.12: Índice hierárquico R-tree. Fonte: (Oracle 2005)

A quadtree utiliza índices e árvores-B para realizar buscas espaciais e outras operações de manipulação de dados (DML do inglês *Data Manipulating Language*). Essa abordagem acarreta em uma criação simplificada para os índices, atualização rápida dos índices e uma herança de um controle de concorrência de árvore B (Kothuri *et al.* 2002).

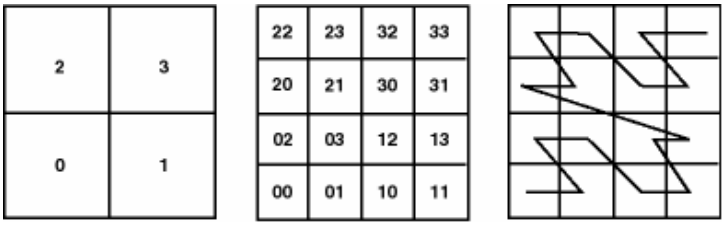


Figura 2.13: Decomposição Quadtree. Fonte: (Murray 2002)

Para o desenvolvimento deste trabalho optou-se pela extensão espacial do SGBD Oracle pela robustez da ferramenta, por conta da extensa documentação e por ter a licença gratuita para uso não comercial.

Capítulo 3

Trabalhos Relacionados

O uso da Realidade Virtual na construção de grandes modelos urbanos em conjunto com sistemas de informação pode servir como um importante veículo de informação. Surge com isso, a necessidade do desenvolvimento de técnicas que otimizem a visualização desses modelos, que sejam associadas a um banco de dados geográfico, permitindo o fornecimento de informações mais fiéis à realidade. Dessa maneira a visualização se transforma em uma excelente metáfora de interface para análise de dados no banco.

Em (Porto *et al.* 2004) é abordado o problema de recuperação de modelos tridimensionais de objetos em SGBD Objeto Relacional, buscando otimizá-lo com o emprego de um algoritmo de oclusão desenvolvido especificamente para este sistema. O trabalho se insere em um contexto maior de desenvolvimento de um sistema completo de apoio à recuperação e exibição de objetos 3D em aplicações de AVC (Ambientes Virtuais Colaborativos) e SIG-3D. Entretanto, não foram utilizados dados geográficos reais e nem foi construído um módulo de visualização, sendo gerados arquivos de saída em AutoLISP para visualização no AutoCAD.

Pode-se observar que não são tratadas as questões relacionadas à visualização e nem de deslocamento dentro do ambiente virtual, e apesar de apresentar técnicas de otimização em SGBD Objeto-Relacional, não utiliza dados de uma área extensa para os seus testes.

Em (Wilmott *et al.* 2001) é descrito um pacote que traz um conjunto de técnicas de renderização e otimização para a visualização de grandes e complexos ambientes urbanos com taxas de quadros por segundo interativas. O

pacote foi construído baseado em uma estrutura de Grafo de Cena proprietária desenvolvido para o projeto CHARISMATIC (Havemann and Fellne 2005). O trabalho apresenta a combinação de Adaptação em Tempo-real de Malhas (ROAM), Descarte de Objetos por Volume de Visualização (*View Frustum Culling*), Descarte por Oclusão (*Occlusion Culling*) e Níveis de Detalhe (LOD) para as edificações em uma única aplicação com o objetivo de conseguir um caminhamento em tempo-real no modelo virtual. Como resultado, é apresentado uma comparação das médias de quadros por segundo no caminhamento do modelo composto de 544.002 polígonos, sendo que sem a utilização das técnicas de otimização foi obtida uma taxa de quadros por segundos de 1,6 quadros por segundo (fps), enquanto que com o emprego delas foi obtida uma taxa de 35,54 fps. Na Figura 3.1 são mostrados os resultados de aplicação de técnicas de texturas.

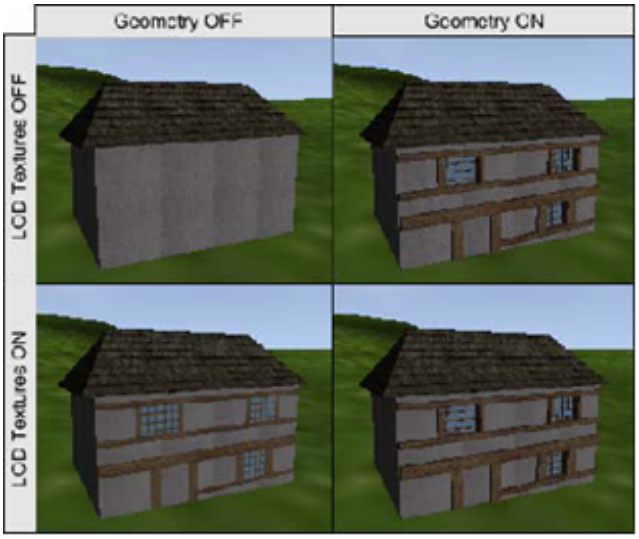


Figura 3.1: Comparação de casa com ou sem textura. Fonte: (Wilmott *et al.* 2001)

Apesar dos bons resultados apresentados em relação ao desempenho, não há um banco de dados geográficos incorporado a sua arquitetura e por isso não apresenta uma maneira prática de adicionar informações úteis a respeito das construções que compõem todo o modelo.

Em (Döllner *et al.* 2005) é apresentada uma técnica que proporciona a visualização expressiva de representações de grandes modelos 3D de cidades,

inspirada em ilustrações artísticas e visualizações cartográficas utilizando a visão panorâmica do modelo. São definidos uma coleção de componentes do modelo da cidade e um algoritmo de múltiplas etapas em tempo-real que atinge representações aceitáveis do modelo, baseado em realce das bordas das principais construções do modelo, efeito de profundidade utilizando cor e sombra, e utilização de texturas nas fachadas. Com essa apresentação estilizada, busca-se facilitar o reconhecimento, a navegação, exploração e análise de informação espacial urbana. O algoritmo tem quatro fases de *rendering*: Fase 1 gera uma codificação de textura das regiões sombreadas da imagem; Fase 2 renderiza a cena com os realces das bordas, sombreamento e fachadas com texturas; Fase 3 renderiza as bordas estilizadas; e Fase 4 renderiza o restantes dos componentes do modelo 3D da cidade. A Figura 3.2 mostra um exemplo de um modelo ilustrativo da cidade 3D criado pela técnica apresentada.

Figura 3.2: Visualização de modelo 3D com técnica ilustrativa. Fonte: (Döllner *et al.* 2005)

Não são detalhadas as técnicas empregadas na geração dos prédios, e nem o emprego de técnicas de otimização do *rendering*. Existe um custo razoável para o pré-processamento antes da visualização final. Nesse trabalho não é mostrada como se dá a integração com uma base de dados espacial.

Um sistema de simulação urbana que tem como objetivo criar uma simulação imersiva em grande escala da cidade de Dublin é descrito em (Hamill and O'Sullivan 2003). A navegação pelo ambiente é feita em primeira pessoa dando

ao usuário liberdade para ir onde desejar. Como foram incluídas praticamente todas as construções da cidade foi empregado LOD, Descarte por Oclusão e um algoritmo de gerenciamento de texturas para otimizar o sistema. O sistema foi desenvolvido a partir de um modelo CAD existente com as bibliotecas OpenGL e OpenAL (OpenAL 2006), esta última para áudio 3D que pode ser utilizados em várias aplicações, tais como jogos. Os modelos utilizados para simular as construções foram feitos com o 3D Studio Max (AutoDesk 2008), têm em média 500 polígonos, utilizam aproximadamente 1 MB de textura cada, sendo que o maior tem 13500 polígonos e 6 MB de texturas (Figura 3.3). A carga e posicionamento dos modelos no ambiente são controlados por um simples arquivo texto. Diante disso, a taxa de quadros por segundos fica entre 25 e 60 fps, dependendo da complexidade da cena atual.

Figura 3.3: Vista do Trinity College em Dublin. Fonte: (Hamill and O'Sullivan 2003)

Nesse caso, pode-se observar que apesar dos bons resultados apresentados em relação ao desempenho, também não há no modelo um banco de dados geográficos incorporado a sua arquitetura e, portanto, não são aproveitados os benefícios da utilização de bases de dados espaciais.

Em (Netto *et al.* 2005) é apresentado o desenvolvimento de uma interface gráfica 3D utilizando a tecnologia de ambientes virtuais com o objetivo de auxiliar a tomada de decisão em um sistema computacional para redução de perdas em redes de distribuição de energia elétrica. Essa interface 3D foi

construída para a visualização de uma grande quantidade de dados em um sistema de distribuição, com redes de grande porte, além de facilitar a interpretação (avaliação) das soluções propostas pelo sistema e de permitir uma fácil editoração do sistema de distribuição com o objetivo de planejamento dessas redes. Nesse trabalho, foi criado um SIG capaz de contemplar o maior número de informações, ligadas ao georeferenciamento, onde houve a necessidade de armazenamento de inúmeros dados e informações sobre a carga da rede de distribuição de energia elétrica, tal como tensão nos postes, carregamento dos cabos da rede, capacidade da subestação e dos alimentadores de energia; dados sobre localização e posicionamento dos postes e subestações em relação à cidade; informações sobre os diferentes tipos de postes, cabos e chaves de abertura e fechamento encontradas na rede elétrica, etc. Para dar suporte a esse grande volume de informações foi necessário a utilização de um banco de dados espacial. A construção do modelo baseou-se em um mapa 2D no formato CAD e imagens aéreas da cidade de São Carlos (SP). A partir do mapa 2D foi gerado o mapa tridimensional. Na Figura 3.4, uma vista de quarteirões modelados.

Figura 3.4: Exemplo de modelagem 3D dos quarteirões da cidade. Fonte: (Netto et al. 2005)

Nesse trabalho não foram citadas a utilização de técnicas de otimização do processo de *rendering*.

Em (Schilling and Zipf 2003) são tratados problemas relacionados à geração de mundos 3D dinamicamente, e tenta resolver os problemas de integração de dados 2D e 3D automaticamente. São introduzidos alguns mecanismos que permitem

a geração automatizada de conjuntos de dados geográficos em 3D, que são mostrados no protótipo implementado, onde são exibidas animações otimizadas do caminhamento por áreas pré-selecionadas, sendo possível a geração de modelos em VRML [1] (VRML 2005) dessas regiões. A cidade de Heidelberg (Figura 3.5), na Alemanha, foi utilizada para demonstração. É proposta a utilização de um Servidor de Realidade Virtual (VR-Server) como camada intermediária para prover interfaces específicas para a carga de diferentes fontes geográficas de dados em 2D e 3D, onde as diversas bases de dados geográficas têm seus sistemas de referências convertidos para um único tipo comum a todos. É utilizado modelo digital de elevação (DEM - *digital elevation model*), com base nas alturas armazenadas de cada construção, para a geração da superfície da área. Os modelos são gerados por extrusão e reposicionados de acordo com sua elevação. Os objetos mais próximos da câmera são trocados por modelos VRML mais detalhados.

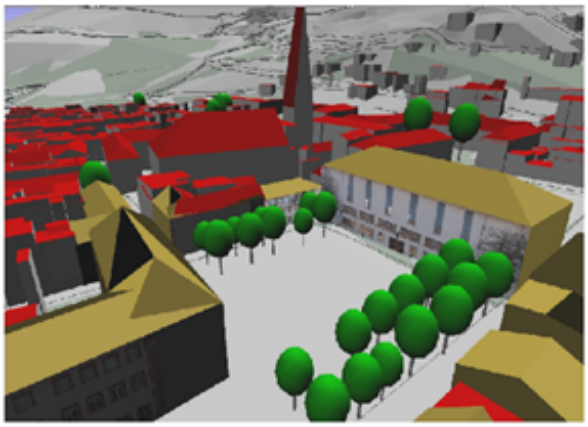


Figura 3.5: Vista de uma quadra cidade de Heidelberg. Fonte: (Schilling and Zipf 2003)

Nesse caso, apesar da boa integração com bases de dados geográficas, no quesito interação com o modelo, há restrições quanto à navegação no modelo, já que é necessária uma seleção prévia de parte da área para que seja gerado o modelo, e por conta disso ainda há um pré-processamento para cada nova seleção de regiões.

Nos trabalhos citados podem-se observar as diferentes soluções para os

[1] Virtual Reality Modeling Language - linguagem utilizada para a modelagem de ambientes virtuais.

problemas relacionados à visualização de modelos urbanos. No entanto, todos apresentam alguma carência. O primeiro trabalho apresenta técnicas de otimização em SGBD Objeto-Relacional, mas não utiliza dados de uma área extensa para os seus testes, nem são tratadas as questões relacionadas à visualização e deslocamento dentro do ambiente virtual. Apesar dos bons resultados apresentados em relação ao desempenho, em (Wilmott *et al.* 2001) e (Hamill and O'Sullivan 2003), os modelos não incorporam um banco de dados geográficos a sua arquitetura e por isso não apresenta uma maneira prática de adicionar informações úteis a respeito das construções que compõem todo o modelo. Em (Döllner *et al.* 2005) não é citado o emprego de técnicas de otimização da renderização, existindo um custo razoável de pré-processamento antes da renderização final, nem é mostrado como se dá a integração com uma base de dados espaciais. Já (Netto *et al.* 2005) implementa SIG, mas não aborda técnicas de otimização em relação a renderização e visualização. O último trabalho apresenta uma interessante integração com a base de dados, mas possui limitações quanto a navegação no ambiente urbano, e alto custo de pré-processamento para a geração das áreas selecionadas a serem visualizadas. Na Tabela 3.1 é mostrado um comparativo resumido entre os trabalhos pesquisados.

Tabela 3.1: Resumo comparativo dos trabalhos pesquisados

	(Porto *et al.* 2004)	(Wilmott *et al.* 2001)	(Döllner *et al.* 2005)	(Hamill and O'Sullivan 2003)	(Netto *et al.* 2005)	(Schilling and Zipf 2003)
Tamanho	164 objetos	544.002 polígonos	N/D*	80 modelos	N/D	N/D
Taxa de quadros por segundos	N/D	35,54	N/D	Entre 25 e 60	N/D	N/D
SGBD espacial	Não	Não	N/D	Não	Sim	Sim
Técnica de Otimização	Descarte por oclusão	Descarte por visão e por oclusão, LOD, ROAM	N/D	Descarte por oclusão, LOD, gerenciamento de textura	N/D	LOD
Visualização	Não	Sim	Sim	Sim	Sim	Sim
Deslocamento no modelo	Não	N/D	Sim	Sim	Sim	Sim

* N/D - Informação não disponível

Na metodologia proposta neste trabalho, buscou-se utilizar as técnicas que permitissem obter um melhor desempenho na visualização e navegação em ambiente virtual urbano sem que se fizesse necessária a utilização de *hardware* com alto poder de processamento. Além disso, o modelo gerado é associado a uma base de dados geográficos, o que permite atrelar importantes informações ao modelo.

Capítulo 4

Estratégia de Otimização para Aceleração da Visualização de Modelos Virtuais Urbanos

Para realizar a visualização de um grande modelo 3D que represente uma área urbana e que possibilite um deslocamento por dentro dele com um bom desempenho, são necessárias várias etapas que empregam técnicas de otimização desde a recuperação dos dados espaciais ao gerenciamento do grafo da cena, bem como na visualização do ambiente virtual construído.

O problema estudado neste trabalho concentra-se na otimização da visualização de um modelo urbano construído a partir de uma base de dados espacial associada a um conjunto de arquivos com os modelos 3D, os quais representam as construções de uma área urbana. Com a utilização de um banco de dados geográficos é dado suporte ao armazenamento, recuperação e visualização desse modelo virtual urbano em tempo real.

Neste capítulo, será detalhado o algoritmo de visualização, a arquitetura de visualização proposta e apresentado os resultados obtidos nos testes realizados com a aplicação desenvolvida.

4.1 Técnica de Aceleração

O algoritmo proposto neste trabalho gerencia o grafo de cena, com base nos dados acessados no banco espacial, utilizando uma estratégia de *cache* baseada em um *buffer* em volta da posição da câmera. Com isso, é possível selecionar um conjunto específico de objetos a uma determinada distância do observador que serão adicionados ao grafo, limitando assim a quantidade de nós no grafo, já que não são carregados todos os objetos do modelo. Com isso, obtém-se uma redução considerável no consumo de recursos computacionais, imaginando um cenário onde seja necessária a visualização de modelos de grandes proporções que podem ter algumas centenas de objetos ou até mesmo milhares ou milhões de objetos.

Inicialmente, pensou-se em utilizar um *buffer* com área triangular com um vértice situado por trás da câmera e os outros dois à frente, como podem ser visto na Figura 4.1a. Nesse caso, seriam pesquisados e carregados no grafo aqueles objetos que estivessem dentro da região cinza claro mais os que estivessem na área cinza escuro, sendo estes últimos carregados mesmo não sendo visíveis, funcionando apenas como uma margem para o caso da câmera ser rotacionada, onde o usuário mudaria o seu campo de visão para os lados.

(*a*) (*b*)

Figura 4.1: Possíveis formas de *buffers*. (a) *buffer* triangular. (b) *buffer* circular.

No entanto, foi adotado um *buffer* circular (Figura 4.1b) em volta da câmera, optando-se por essa forma ao invés de uma forma triangular para minimizar o número de inserções e remoções de objetos no grafo quando a câmera for "girada"rapidamente e em um ângulo não contemplado pela região triangular.

Por conta disso, os objetos situados atrás da câmera também são carregados, desde que estejam dentro da área delimitada pelo *buffer*. Este *buffer* tem um raio que é passado como parâmetro e que define a sua área total.

Para obter-se a lista de objetos que são carregados ou descartados do grafo de cena é feita a interseção da área contida no *buffer* com as geometrias contidas nas tabelas espaciais. Essa operação é realizada utilizando os operadores espaciais adequados em conjunto com as funções espaciais necessárias. Para agilizar o acesso dos dados espaciais foram criados índices espaciais para cada uma das tabelas que contêm dados geográficos. No desenvolvimento deste trabalho foi utilizado o Oracle Spatial, que obriga a se criar índices espaciais para que se possam usar os operadores espaciais. Abaixo é mostrada a sintaxe de criação de um dos índices utilizados. Nesse caso, está sendo criado o índice espacial square_idx para o campo *GEOMETRY* da tabela *square*.

```
CREATE INDEX square_idx ON square(GEOMETRY)
INDEXTYPE IS mdsys.spatial_index;
```

Neste trabalho, foi utilizado o operador espacial SDO_RELATE do Oracle Spatial utilizando a máscara ANYINTERACT e as funções espaciais SDO_GEOM.SDO_DIFFERENCE e SDO_GEOM.SDO_BUFFER.

O operador espacial SDO_RELATE é utilizado para determinar se duas geometrias interagem entre si, e com a utilização da máscara ANYINTERACT são retornados todos os objetos que tenham alguma relação topológica entre eles. A função espacial SDO_GEOM.SDO_DIFFERENCE fornece como resultado de sua chamada, uma geometria que corresponde à diferença topológica entre duas geometrias. Já a função SDO_GEOM.SDO_BUFFER cria uma nova geometria ao redor da geometria passada como parâmetro. Abaixo um exemplo de uma das consultas espaciais realizadas usando a sintaxe dos operadores e funções espaciais para o Oracle Spatial.

```
SELECT lot.idlot, square.idsquare, street.name, lot.geometry, lot.path
FROM lot , square, street
WHERE lot.idsquare = square.idsquare
   AND lot.idstreet = street.idstreet
   AND SDO_RELATE(lot.geometry,
```

```
    SDO_GEOM.SDO_DIFFERENCE(
      SDO_GEOM.SDO_BUFFER( MDSYS.SDO_GEOMETRY( 3001, 1,
      MDSYS.SDO_POINT_TYPE("oldCoordX", "oldCoordY", 0), NULL,
      NULL ), sizeBuffer, 0.01, 'unit=m arc_tolerance=0.005' ),
      SDO_GEOM.SDO_BUFFER( MDSYS.SDO_GEOMETRY(3001,1,
      MDSYS.SDO_POINT_TYPE("coordX", "coordY", 0) , NULL, NULL ),
      sizeBuffer, 0.01,'unit=m arc_tolerance=0.005' ),
      0.001
    ) ,
    'mask = ANYINTERACT querytype = WINDOW'
) = 'TRUE'
```

Na primeira iteração, são consultadas todas as geometrias contidas nas tabelas espaciais que têm qualquer interação com a área delimitada pelo *buffer*, e então são carregados todos os objetos 3D correspondentes. A partir do primeiro deslocamento da câmera é feita a diferença entre o *buffer* atual e o *buffer* anterior para que se possa obter as áreas que contêm os objetos que deverão ser removidos e inseridos no grafo de cena. Com isso não é necessário fazer a interseção das tabelas espaciais com toda a área do contemplada pelo *buffer*, mas somente com as áreas resultantes das diferenças entre os *buffers* da posição atual e anterior. Estas consultas são realizadas a partir da execução do método de remoção de objetos e na seqüência pelo método responsável pela inserção dos novos objetos no grafo.

Este procedimento é mostrado na Figura 4.2, onde estão representadas as áreas resultantes dessa operação, mostrando-se o deslocamento da câmera do ponto A ao ponto B. Nesse exemplo, a área em amarelo envolve todos os objetos a serem descartados, os quais são determinados pela diferença entre o *buffer* da posição anterior (A) e o novo *buffer* (B) da posição atual. Já os objetos que estão na área de interseção são mantidos e aqueles que estão dentro da área em azul serão incluídos no grafo, sendo estes obtidos pela diferença entre o *buffer* atual (B) e o *buffer* antigo (A).

A cada inserção é verificado se os objetos que estão no *buffer* consultado já estão ou não no grafo. Em caso afirmativo eles serão desconsiderados, caso

contrário, serão inseridos no grafo. Com essa checagem impede-se a redundância na carga dos objetos. Essa verificação é feita através de listas de objetos com os índices das geometrias dos nós inseridos anteriormente no grafo, onde a cada inserção no grafo é também inserido um objeto de identificação na lista de índices, contendo a identificação espacial (id) desta geometria.

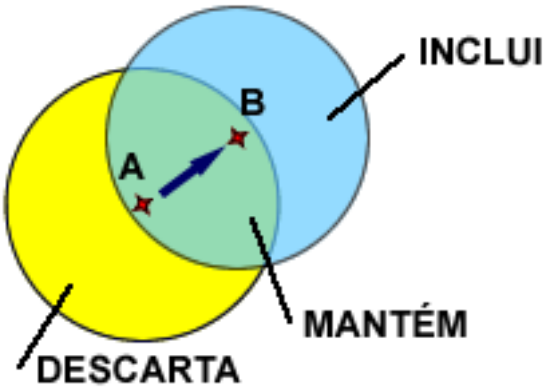


Figura 4.2: Áreas cobertas pelos *buffers*

Como mais uma estratégia de redução de interações com o grafo da cena, o que acarreta um considerável consumo de recursos de *hardware*, definiu-se um mecanismo baseado em uma tolerância no deslocamento feito pelo usuário da câmera. Isto é, ao invés de realizar a cada deslocamento todo o procedimento de busca no banco e atualização do grafo, é checado se a posição da câmera encontra-se dentro da área circular da tolerância, e em caso afirmativo não será realizada qualquer atualização, como pode ser visto na Figura 4.3.

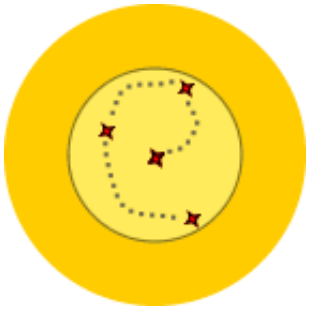


Figura 4.3: Área de tolerância no deslocamento da câmera

Enquanto a câmera for deslocada dentro da área em amarelo claro nenhum outro objeto será inserido ou removido do grafo. As junções das áreas internas e mais externas representam a região onde estão todos os objetos contidos no grafo naquele momento. Essa área mais interna, ou área de tolerância, também tem o seu valor parametrizado.

No diagrama de classes mostrado na Figura 4.4, são exibidas as principais classes implementadas do modelo, bem como os seus atributos e métodos mais relevantes. No diagrama exposto, a classe *Application* corresponde ao arquivo fonte onde se encontra a função principal do protótipo implementado.

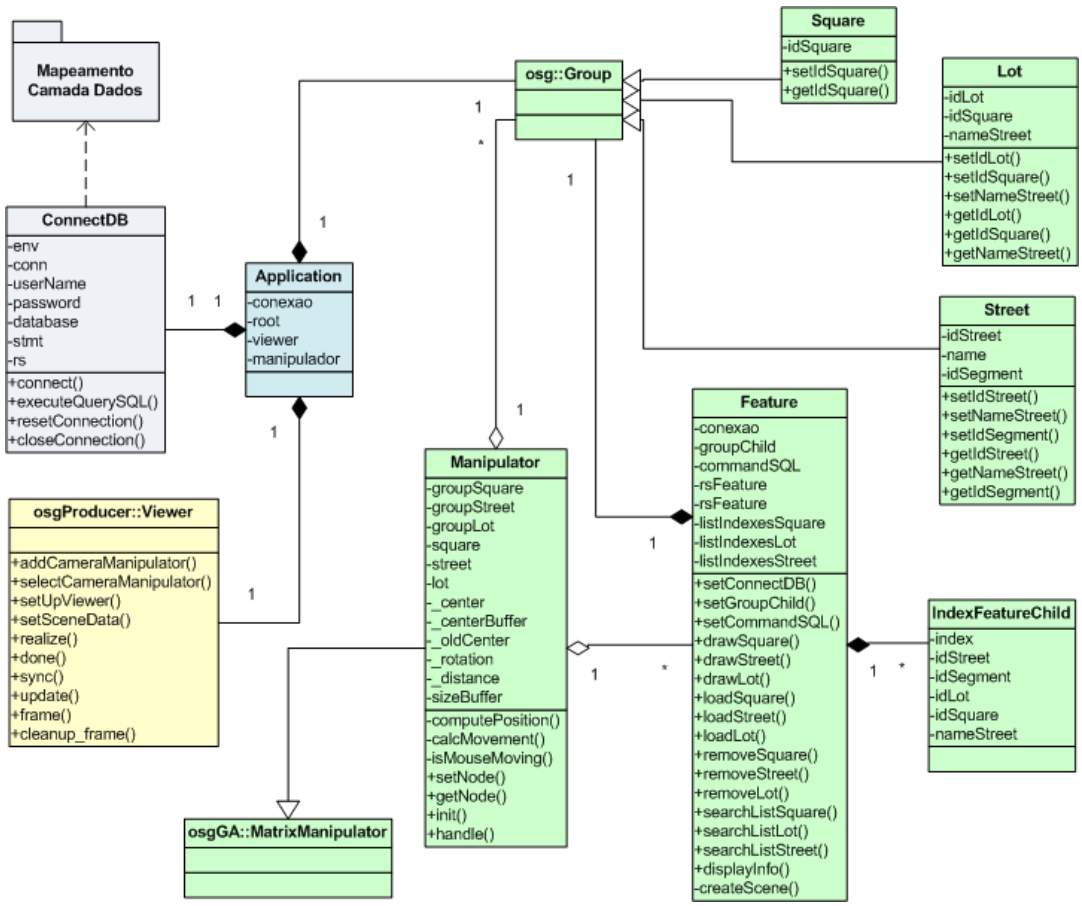


Figura 4.4: Diagrama de classes da aplicação desenvolvida

Em *Application* são instanciados os objetos: de conexão ao banco de dados, a raiz do grafo de cena (do tipo osg::Group), o viewer (do tipo osgProducer::Viewer) e o manipulador da câmera (do tipo Manipulator::osgGA::MatrixManipulator).

A classe *Manipulator* tem uma referência para o objeto *root*, com ela pode-se anexar outros grupos de objetos ao grafo. No caso, são instanciados três objetos do tipo osg::Group (groupSquare, groupStreet e groupLot), um para cada tipo espacial utilizados no trabalho, nos quais são adicionados os objetos 3D ou 2D correspondentes. Nessa classe, são instanciados objetos do tipo *Feature*, também um para cada entidade espacial. Ainda nesta classe, são definidos os parâmetros da câmera o do *buffer*, bem como os métodos responsáveis pelo cálculo da posição e de movimento da câmera, de acordo com a utilização do *mouse*.

A classe *Feature* é responsável pelo gerenciamento de um grupo específico de geometrias, por exemplo, as quadras do modelo. Por conta disso, ela instancia a classe osg::Group e utiliza polimorfismo com as classes *Square, Street* e *Lot*. São criados três objetos da classe *Feature* em *Manipulator* para que se possam executar as consultas espaciais e os métodos de remoção e inserção de objetos no grafo, por isso necessita também da referência do objeto raiz (*root*) do grafo. Nela também, estão as listas de índices dos objetos presentes no grafo para cada entidade, por exemplo, o objeto que representa a *feature Square* tem uma lista de índices dos nós do mesmo tipo inseridos anteriormente no grafo, utilizada para consultas no momento de novas inserções. Esta lista é do tipo IndexFeatureChild, nela estão os atributos necessários para o gerenciamento dos objetos no grafo de cena, tais como o índice de nó no grafo e os atributos utilizados como chave nas buscas na lista.

As classes *Square, Street* e *Lot* são responsáveis pelo mapeamento das entidades no banco espacial. Nelas, podemos incluir dados não geográficos específicos a cada tipo.

4.2 Arquitetura de Visualização Proposta

Para realizar a visualização de modelos urbanos desenvolveu-se a modelagem do problema com um banco de dados espacial 2D, sobre o qual são realizadas consultas espaciais para que sejam recuperados os objetos (*features*) que representam as quadras, os segmentos de rua e os lotes dos prédios, e a partir disso são carregados os objetos 3D que representam as construções arquitetônicas, ou então são desenhadas essas *features*.

Na abordagem adotada neste trabalho, optou-se pela navegação em primeira pessoa, por ser mais próximo da visão real e por se adequar muito bem a estratégia de *cache* proposta, onde somente os objetos dentro de um determinado raio em volta da câmera são carregados. O modelo aqui apresentado baseia-se na premissa de que modelos urbanos são densos, no que diz respeito à quantidade de objetos nele contidos, quando visualizados do solo e, em como conseqüência, os prédios mais próximos obstruem a visualização dos demais. Por conta disso, em todas as direções de visada os espaços são preenchidos pelos prédios sem a necessidade de ser carregado um grande volume de objetos. Caso fosse utilizada outra alternativa de visualização, como de visão panorâmica, seria necessária a implementação de outras estratégias de otimização.

Na Figura 4.5, expõe-se uma visão geral da arquitetura de visualização proposta, a qual está organizada em três camadas: apresentação, aplicação e dados.

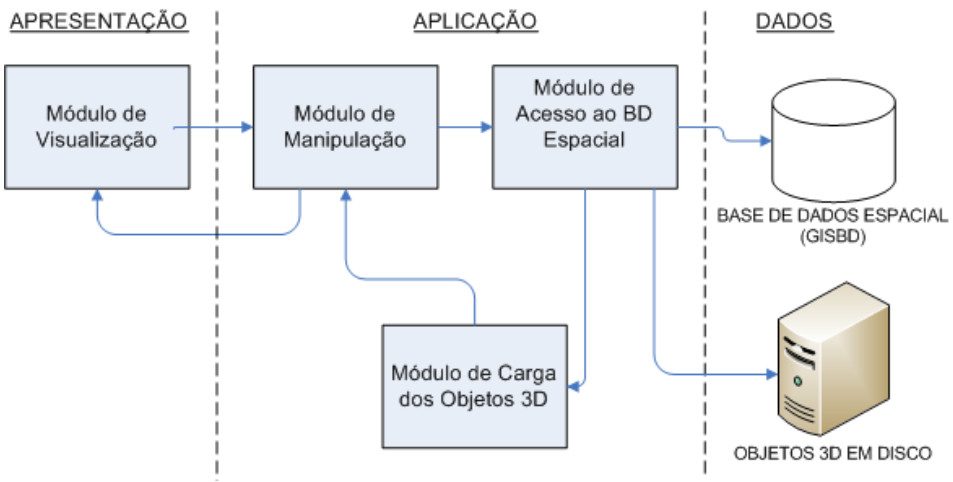


Figura 4.5: Visão geral da arquitetura

A camada de apresentação é responsável pela visualização da cena. Nela há o laço de visualização que é executado por todo o período de execução da aplicação utilizando métodos do objeto *viewer* para gerar a cena. Também é definida e parametrizada a câmera da cena em conjunto com o objeto *manipulator*, este último responsável por tratar os eventos de entrada e atualizar os parâmetros da câmera. Esta camada caracteriza-se como a interface de interação entre o usuário

e a aplicação desenvolvida.

Na camada de aplicação fica hospedada a lógica de negócios do sistema, na qual é implementada a classe *Manipulator*, onde é utilizada uma referência ao objeto de conexão com a base de dados para realizar as consultas, são processados os resultados e implementada a estratégia de *cache* baseada nas consultas espaciais para a obtenção dos dados geográficos relevantes à visualização. A partir disso, é feita a inserção e exclusão de objetos no grafo de cena, bem como o gerenciamento dos dados relacionado aos modelos 3D das construções armazenados em disco que são carregados nas cenas.

Por fim, a camada de dados que encapsula o acesso à base de dados, tanto dos dados armazenados no banco de dados como nos arquivos dos objetos 3D que se encontram em disco. A seguir o detalhamento de cada uma delas.

4.2.1 Camada de Apresentação

Na camada de apresentação encontra-se o módulo de visualização que é responsável pela exibição da cena gerada a partir do grafo. Para isso é criado o objeto de visualização da cena, o *viewer*, que recebe como um de seus argumentos a câmera, bem como o objeto *manipulator*, responsável pelo tratamento dos eventos do *mouse* e a atualização dos parâmetros da câmera. Além disso, é nessa camada que é criado o objeto raiz do grafo de cena, responsável por agrupar todos os demais nós que compõem a cena. Faz-se necessário também a utilização de um laço responsável pelo *rendering* que garante a atualização da cena a cada mudança ocorrida na mesma ou então na câmera. Sendo exibidas também algumas informações sobre cada nó do grafo à medida que se passa o cursor do *mouse* sobre eles, tais como a identificação do objeto dentro do grafo de cena, o nome do arquivo em disco e as coordenadas do objeto dentro do modelo, como visto na Figura 4.6.

A interface entre o usuário e a aplicação se dá através do *mouse*, que ao mover o *mouse* com o botão esquerdo pressionado aplica uma rotação em relação ao eixo vertical da câmera, enquanto que ao mover o *mouse* com o botão direito pressionado, este aplica à câmera um deslocamento horizontal em todas as direções.

Inicialmente são carregados os objetos de uma localização pré-determinada e a partir do primeiro movimento de câmera é executado o fluxo de tarefas

ilustrado resumidamente no diagrama de seqüência apresentado na Figura 4.7.

Figura 4.6: Imagem com as informações do objeto 3D sendo exibidas

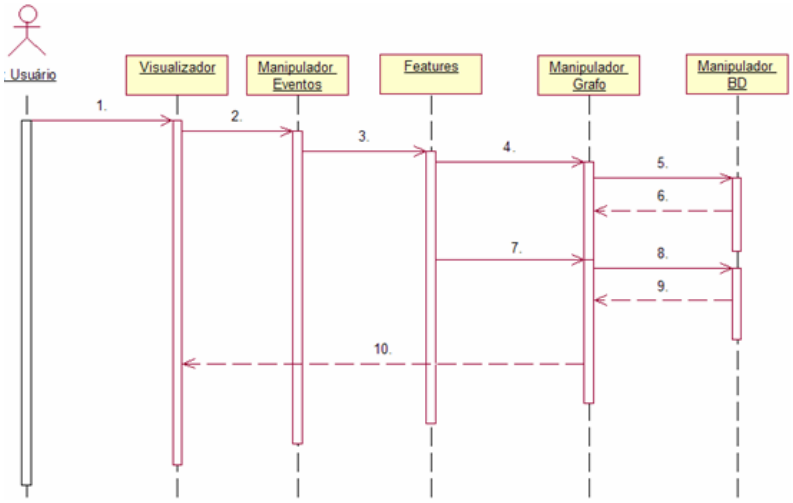


Figura 4.7: Cenário de atualização do grafo de cena e a visualização

1. Usuário movimenta o *mouse*;
2. Tratamento do evento do *mouse*, no caso o botão direito pressionado e arrastando para realizar uma translação;
3. Chamada de método que calcula a nova posição da câmera;

4. Executado o método de remoção dos objetos;
5. São consultados no banco os objetos a serem removidos de acordo com o *buffer*;
6. Retornam os objetos removidos;
7. Executado o método de carga dos objetos;
8. São consultados no banco os objetos a serem adicionados no grafo de acordo com o *buffer*;
9. Retorna objetos adicionados;
10. Grafo atualizado e cena gerada para a visualização.

4.2.2 Camada de Aplicação

A camada de aplicação é responsável pela lógica de negócios. Nela é implementada a classe *Manipulator*. Esta classe recebe e trata os eventos repassados pela interface com o usuário, realiza as transformações espaciais sobre a câmera, obtém uma referência para o objeto de conexão à base de dados, instancia objetos da classe *Feature* para realizar as consultas espaciais de acordo com a estratégia de *cache* utilizando um *buffer* em volta da posição da câmera. Na seqüência, processa os resultados das consultas para obter os objetos que serão incluídos ou excluídos do grafo de cena, realizando a atualização do grafo de cena do modelo.

Como resultado dos procedimentos realizados nesta camada, obtém um grafo de cena estruturado que é utilizado para a visualização da cena na camada de apresentação. Na Figura 4.8, observa-se como está estruturado o grafo de cena desenvolvido neste trabalho, nele se encontra o nó raiz que tem como nós filhos três nós de agrupamento. Um para cada grupo de entidades espaciais contidas no banco de dados (*Street, Square* e *Lot*). Por fim, a esses nós estão ligados os nós que armazenam os objetos 3D ou desenhos das geometrias.

Esta camada é composta pelos seguintes módulos: acesso ao banco de dados espacial, manipulação e carga dos objetos 3D.

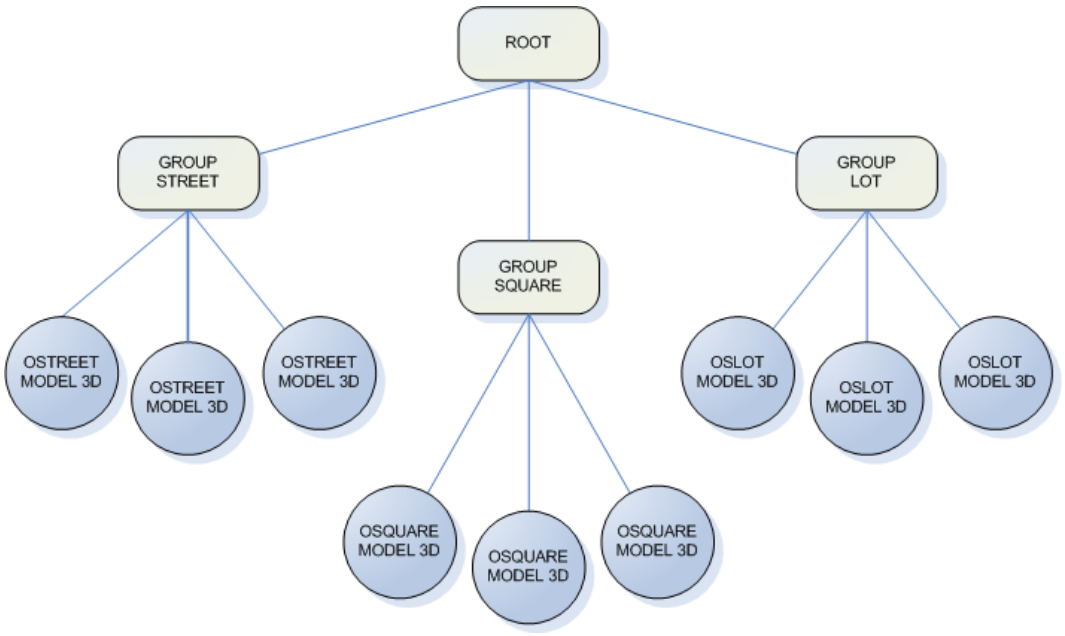


Figura 4.8: Grafo de cena do modelo

O módulo de acesso ao banco de dados espacial é a interface responsável por conectar a aplicação ao banco de dados, através de uma referência à instância do objeto de conexão à base de dados, permitindo que os dados geográficos sejam reconhecidos e utilizados na mesma. Foi criada uma classe genérica de acesso ao banco de dados com todos os métodos necessários à interação da aplicação com a base de dados geográficos. Como citado anteriormente, foi utilizada a extensão espacial do Oracle, através da biblioteca OCCI [1] (Oracle 2007) onde estão definidos métodos específicos para acesso aos dados espaciais desse SGBD. Mas com a estrutura de classes desenvolvida torna fácil a extensão para usar outros SGBDs espaciais que sejam compatíveis com o padrão Open Geospatial (OGC 2007).

O módulo de manipulação do grafo de cena é responsável por construir e atualizar o grafo da cena, baseado nos dados fornecidos pela camada de acesso aos dados, e tratar os eventos enviados pela camada de apresentação, realizando as transformações geométricas (translação e rotação) sobre a câmera utilizando uma estratégia de *cache* baseada em um *buffer* em volta da câmera. Através

[1] Oracle C++ Call Interface - Interface de Chamadas Oracle para linguagem de programação C++.

desta estratégia é permitido selecionar os objetos que estão a certa distância do observador, os quais serão adicionados ao grafo.

Já o módulo de carga dos objetos 3D realiza a inserção ou descarte de objetos 3D no grafo de cena, sendo verificado a cada inserção se os objetos que estão no *buffer* consultado já estão ou não no grafo. Em caso afirmativo eles serão desconsiderados, caso contrário, serão inseridos no grafo. Com essa checagem impede-se a redundância na carga de objetos.

4.2.3 Camada de Dados

A camada de dados é representada por uma instância de um banco de dados espacial, denominada de fonte de dados, bem como um conjunto de arquivos armazenados em disco dos objetos 3D modelados. Essa camada é a responsável pela independência de bancos de dados na manipulação de dados geográficos. Como se tem a camada de dados implementada como estrutura de classes de forma isolada, apesar de utilizar o Oracle Spatial, a arquitetura pode ser estendida para lidar com diferentes fontes de dados, por exemplo, Postgresql, MySQL e IBM DB2. Para o desenvolvimento deste trabalho optou-se pela utilização do Oracle pelas vantagens descritas anteriormente.

A base de dados é composta por um conjunto específico de tabelas para a representação dos dados geográficos no SGBD. O esquema de dados foi construído levando em consideração o relacionamento das entidades espaciais e não-espaciais que representam o espaço urbano modelado. Todas as tabelas com atributos espaciais (*SEGMENT, SQUARE* e *LOT*) têm os campos chamados *Geometry* e *Path*. Este último utilizado para guardar a referência para os objetos 3D (construções arquitetônicas) em disco, enquanto que o primeiro armazena a geometria bidimensional da entidade espacial. É necessária também a definição das tabelas não-espaciais STREET e SEGMENT_STREET, sendo a primeira utilizada para armazenar os códigos e nomes de cada rua, e a segunda para materializar o relacionamento N-N entre as tabelas SEGMENT e SQUARE. A partir do relacionamento da tabela STREET com as tabelas SEGMENT e LOT, é possível realizar consultas para se identificar, por exemplo, todos os lotes que pertencem a uma determinada rua, ou então, a partir de um dado segmento de rua obtêm-se todos os lotes por onde passa a rua a qual este segmento pertence.

A Figura 4.9 mostra o diagrama do esquema de dados desenvolvido para este trabalho.

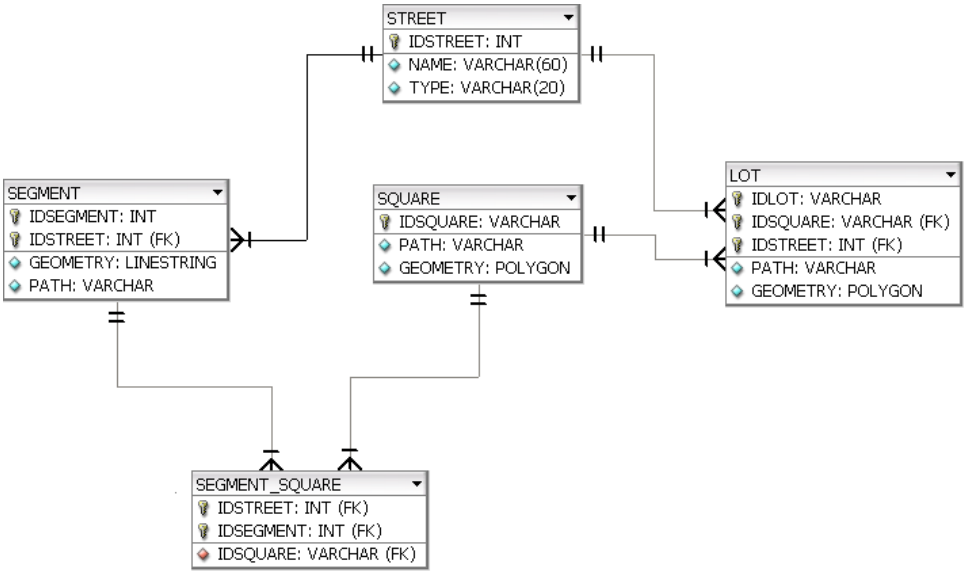


Figura 4.9: Modelo de dados desenvolvido

As informações não-espaciais, a exemplo de conteúdo informativo sobre os monumentos históricos, podem ser armazenadas em outras tabelas que devem ser relacionadas com as tabelas com atributos espaciais (*SEGMENT, SQUARE* e *LOT*), tornando possível a associação de todo o tipo de informação sobre os monumentos.

Vale ressaltar que os objetos 3D que são visualizados na aplicação não são armazenados no banco, mas somente uma referência para eles. Estes são armazenados em disco com uma nomenclatura padronizada, onde o nome do arquivo deverá ser composto pelo nome da classe de objetos (*SEGMENT, SQUARE* e *LOT*), concatenado com os valores que compõem as chaves de cada tabela mais a extensão do arquivo. Por exemplo, tem-se o arquivo lot1_66_3.3DS que é um lote da quadra de código 1, como código do lote 66 e o código 3 que identifica a respectiva rua.

4.3 Protótipo Desenvolvido - RevVir

O RevVir é o nome da aplicação desenvolvida neste trabalho, onde pode-se observar os reultados do algoritmo proposto sendo executado. Consiste em um visualizador que além de exibir os objetos 3D no ambiente virtual também possibilita a exibição de um conjunto de informações de cada objeto, bem como o caminhar dentro do modelo urbano. Nele o usuário tem o modo de visualização em primeira pessoa por padrão, mas é possível ter-se uma visão panorâmica de todo o modelo, entretanto, para este caso não foram implementadas estratégias de otimização. Também foram desenvolvidos métodos de desenho a partir das geometrias armazenadas no banco, possibilitando com isso a visualização de um mapa 2D de toda a área com todas as geometrias armazenadas, como pode ser visto na Figura 4.10.

Figura 4.10: Ilustração criada com os métodos de desenho a partir das geometrias no banco de dados

As linhas em amarelo são as ruas e as verdes, as quadras e lotes presentes nesta região. Neste caso, em vez de utilizar os métodos de carga dos objetos 3D, foram executados os métodos de desenho em 2D.

4.3.1 Aquisição de Dados

Como referência para a modelagem do ambiente urbano a ser utilizado na aplicação, foi utilizado como base o centro histórico da cidade de São Luís, capital do Estado do Maranhão no Nordeste do Brasil que foi declarada patrimônio da humanidade pela UNESCO. O centro histórico é composto por 115 quadras, 53 ruas que foram divididas em 205 segmentos de rua e 962 lotes. Cada segmento de uma rua é delimitado pelo cruzamento com as outras ruas. Na Figura 4.11, podem-se visualizar em azul as ruas da área usada como referência para o modelo.

Figura 4.11: Visualização do mapa 2D da área modela da aplicação

A aquisição dos dados espaciais utilizados para povoar o banco com os dados referentes ao centro histórico foi feita a partir de um modelo CAD. Após a construção da base de dados foi implementada a aplicação responsável por se comunicar com o banco de dados, acessar à base de dados espacial, fazer as pesquisas espaciais e com os seus resultados gerar a visualização da área georeferenciada e dos modelos 3D.

4.3.2 Modelagem de Objetos 3D

Foi dada prioridade à modelagem dos objetos em 3D que representam as quadras e as construções que são carregadas no modelo urbano. Dentre estes foram modeladas 115 calçadas com as dimensões das quadras, 962 prédios e as praças, além disso, 205 segmentos de rua. Totalizando 1282 objetos para serem visualizados na cena, com 42228 primitivas e 94236 vértices.

Foram feitos testes somente com objetos simplificados para representar estes volumes. Não foram aplicadas texturas ao modelo. Para a realização da modelagem dos objetos 3D foi utilizado o *software* 3D Studio Max. A Figura 4.12 mostra a visualização de parte dos objetos modelados em duas áreas distintas do modelo.

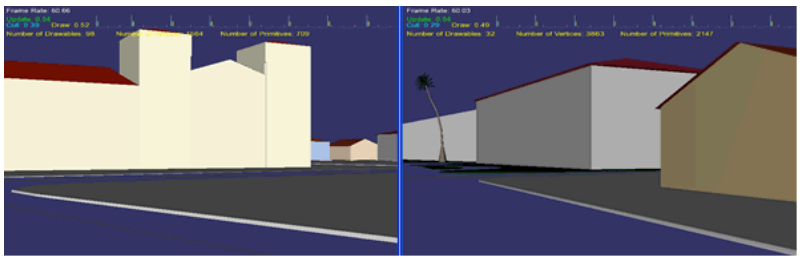


Figura 4.12: Modelos 3D das construções visualizadas na aplicação

Na Figura 4.13, observa-se os possíveis modos de visualização, a forma padrão de visualização no RevVir, enquanto que na Figura 4.14, uma alternativa de visão panorâmica do modelo.

Figura 4.13: Modo de visualização em primeira pessoa

4.3.3 Resultados Obtidos

No desenvolvimento do protótipo foram realizados alguns testes de desempenho a cada etapa do trabalho. Inicialmente, sem a implementação da estratégia de *cache* de visualização, a taxa média de quadros por segundos (fps) ficava em torno

de 45 fps, pois não havia nenhuma política de gerenciamento do grafo de cena que otimizasse o processo, sendo necessário carregar todos os 1282 objetos do modelo antes da visualização começar. Por conta disso, eram consumidos razoáveis recursos de *hardware* e demandava um maior tempo de pré-processamento antes que se pudesse visualizar a cena. A configuração de *hardware* onde foram realizados esses testes foi a seguinte: processador um AMD Athlon 64 3000+ de 1.8 GHz de *clock*, 1024 MB de memória RAM, placa aceleradora de vídeo NVidia GeForce FX 5500 de 128 MB, rodando o sistema Operacional Windows XP.

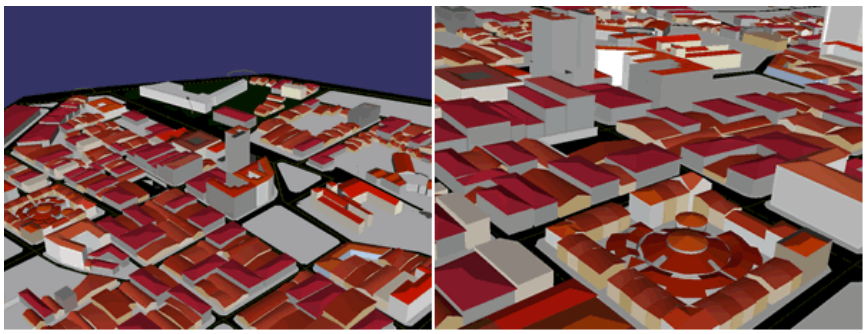


Figura 4.14: Visão panorâmica da área urbana

Após a inclusão da técnica de *cache* utilizando o *buffer*, que limitou o número de objetos no grafo de cena, a taxa média de quadros por segundos passou a ser de 68 fps aproximadamente, no *hardware* citado acima, sendo reduzido consideravelmente o número de objetos no grafo e o consumo de recursos de máquina. Na Figura 4.15, pode-se verificar a taxa de quadros por segundos em uma região do modelo.

Nos testes que foram registrados, determinou-se uma trajetória em que foram obitidas a taxa de quadros por segundo em alguns pontos desta. Na Figura 4.16, são destacados a trajetória e os pontos (representados por letras) de conferência dos fps. O gráfico da Figura 4.17 mostra os números que ilustram os testes realizados, com a variação média das taxas de quadros por segundos nas duas situações. Em azul, a representação da navegação sem otimização, enquanto que na cor laranja têm-se os fps com otimização.

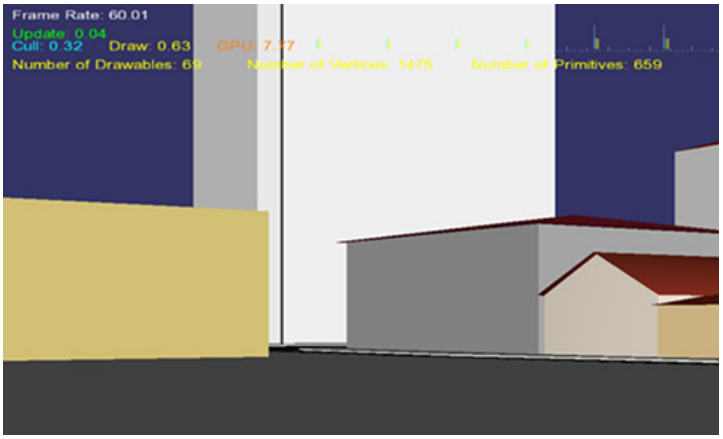


Figura 4.15: Visão do modelo com a taxa de quadros por segundos

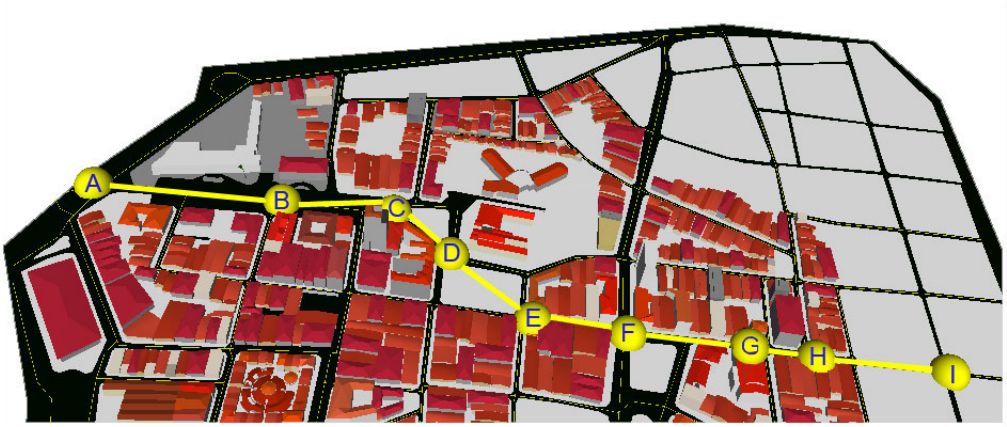


Figura 4.16: Trajetória de navegação utilizada para testes

No ponto E, observa-se a menor taxa devido maior nível de detalhamento, e também, por ter uma grande concentração de construções. A partir do ponto G, começa a diminuir o número de prédios e a aumentar o fps, sendo que na posição I não há prédios e a câmera está direcionada para fora dos limites do modelo urbano.

Para se chegar a um valor de parâmetro para o tamanho do raio do *buffer*, foram realizados alguns testes a fim de garantir um bom desempenho. Na Figura 4.18, são mostradas as taxas para tamanhos de raios entre 100 e 300.

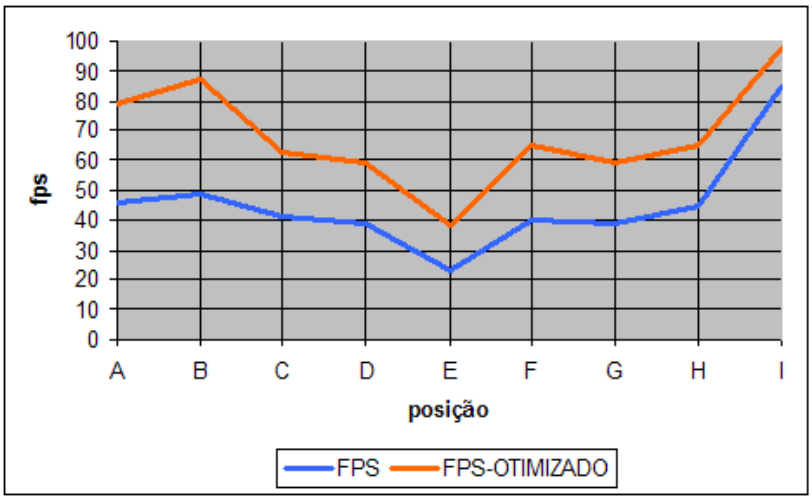


Figura 4.17: Gráfico de comparação de fps

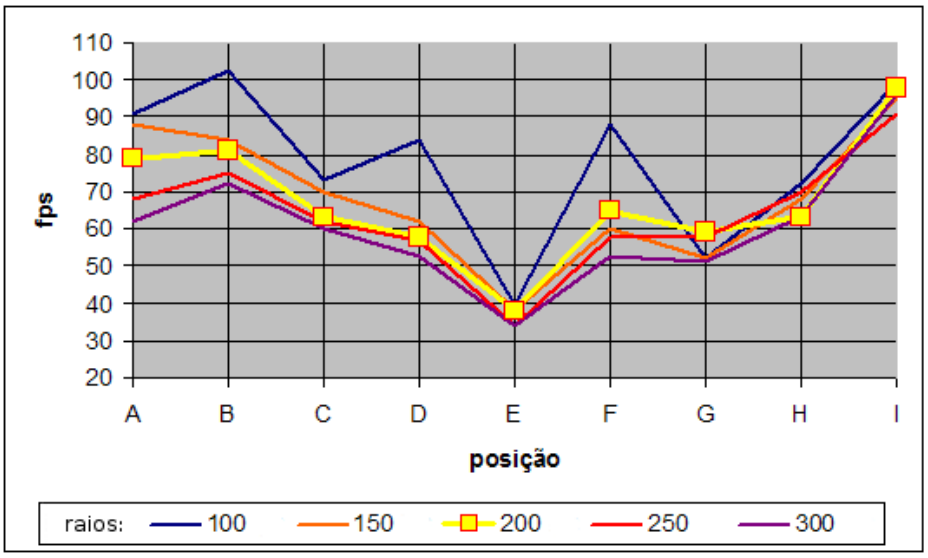


Figura 4.18: Gráfico de comparação de fps para diferentes raios do *buffer*

Optou-se pela utilização do *buffer* com raio de 200 unidades como valor padrão, devido a melhor relação entre performance e representação aceitável de um modelo urbano. Nos testes realizados com o raio de 100, resultaram em taxas mais altas, mas com pequena quantidade de objetos, assim como o caso de raio de 150 unidades. Utilizando os raios com 250 e 300 unidades o fps diminui pouco.

Foram realizados também testes em duas outras máquinas, com as seguintes configurações: o primeiro, com processador Pentium 4 de 3.0 GHz de *clock*, 1024 MB de memória RAM, placa aceleradora de vídeo NVidia Ge-Force 6200 de 128 MB. O segundo, com processador AMD Athlon 64 3000+ de 1.8 GHz de *clock*, 1024 MB de memória RAM e placa aceleradora de vídeo ATI Radeon 9600 Pro de 256 MB. Ambas as máquinas com o sistema Operacional Windows XP. A primeira obteve a mesma taxa de quadros por segundos obtidos pela máquina citada anteriormente, enquanto que a segunda houve um ganho considerável, atingindo taxas de aproximadamente 200 quadros por segundo. Atribui-se estes últimos resultados à placa aceleradora de vídeo presente na segunda máquina.

Na Figura 4.17 pode-se observar uma visão da planta do modelo. Nesse caso, o agrupamento de objetos corresponde aqueles que têm alguma interação com o *buffer* com um raio de 200 unidades em torno da posição da câmera.

Figura 4.19: Objetos carregados no *cache*

Foram feitos, também, testes com o banco de dados e a aplicação de visualização em máquinas distintas para que fossem observadas possíveis variações nos resultados. No entanto, não foram notadas variações consideráveis na taxa de quadros por segundo.

Capítulo 5

Conclusão

Os modelos virtuais urbanos têm sido empregados para vários fins. Com o uso de realidade virtual é possível realizar passeios turísticos por mundos virtuais, planejamento urbanístico de cidades, sistemas de navegação e localização, entre outros. Por isso, é crescente o número de pesquisas relacionadas ao desenvolvimento de grandes modelos virtuais urbanos. Entretanto, as aplicações desenvolvidas para esse fim, normalmente necessitam de *hardware* poderoso e de alto custo.

Durante o levantamento bibliográfico realizado nesta pesquisa, pode-se observar o emprego de várias técnicas para a solução do problema de visualização de modelos urbanos nos trabalhos relacionados. Alguns destes trabalhos empregam SGBDs espaciais e concentram-se em técnicas de otimização no *rendering*, tais como *culling* e LOD, no entanto, não foi observada nesses trabalhos a utilização em conjunto de base de dados espaciais, otimização no banco de dados e técnicas de otimização na visualização neles, necessitando, por isso, do emprego de *hardware* de alto desempenho e custo.

O principal objetivo do algoritmo proposto neste trabalho é possibilitar a visualização de grandes modelos urbanos em sistemas com restrições de *hardware*, sem sacrificar a sensação de continuidade no deslocamento pelo ambiente virtual. Além disso, incorporar ao ambiente virtual um grupo de informações contidas em uma base de dados geográfica, propiciando uma interface homem máquina. Possibilitando a utilização da arquitetura para a construção de uma possível interface de interação com vários tipos de informações multimídia do ambiente

urbano, especialmente interessante para a construção de aplicações dedicadas à popularização, o ensino e à preservação de informação cultural associado às cidades históricas.

Neste trabalho foi desenvolvido um algoritmo de aceleração da visualização de modelos virtuais urbanos, com um *cache* de objetos que se baseia em um *buffer* espacial em banco de dados geográficos, que garante o suporte ao armazenamento, recuperação e a visualização desses modelos urbanos em tempo real. O algoritmo desenvolvido apresentou um bom desempenho, baseado em operações realizadas no banco de dados, através de índices, operadores e funções espaciais, que foram utilizados para implementar a estratégia de *cache*, que possibilita consultas eficientes dos objetos que devem ser mantidos ou removidos do *cache*. Com disso, obteve-se um gerenciamento mais eficaz do grafo de cena, com a simplificação e redução de tamanho deste, possibilitando a execução da aplicação desenvolvida sobre uma plataforma de *hardware* convencional em termos de recursos computacionais.

Os resultados obtidos podem ser considerados satisfatórios devido ao desempenho médio conseguido durante os testes, onde se mantiveram aceitáveis taxas de quadros por segundo durante o caminhar por dentro do modelo urbano. Entretanto, houve reduções mais acentuadas na taxa de atualização em áreas do modelo com alta concentração de objetos, ou então quando estes tinham um maior nível de detalhes. Não foram aplicadas texturas nestes objetos, já que não foi implementada nem uma otimização para a utilização das mesmas. Foram realizados testes com o SGBD instalado local e remotamente, entretanto, não foram percebidas grandes diferenças no desempenho, já que o servidor estava presente em uma rede local.

Como trabalho futuro, pretende-se estender a arquitetura, incorporando outras técnicas de otimização a fim de aumentar a eficiência e usabilidade do sistema (LOD e *prefetch*), e também expandir o módulo de acesso a banco de dados para possibilitar acesso a fontes de dados em outros SGBD's espaciais. Além disso, empregar a utilização de modelos digitais de terreno (DEM) para a visualização do relevo, levando em consideração dados de elevação de terreno para a geração de superfícies mais fiéis a realidade. Pretende-se ainda, desenvolver estratégias de otimização para a visualização panorâmica também. Finalmente, pretende-se

incorporar textura aos objetos que representam as construções arquitetônicas a fim de tornar mais agradável e realista a navegação no modelo, e se for o caso, desenvolver algoritmos de otimização no gerenciamento de memória utilizada para o mapeamento dessas texturas. Com isso, permitir a construção de uma biblioteca digital multimídia de áreas urbanas de interesse do patrimônio histórico cultural.

Referências Bibliográficas

Alliez, P., M. Meyer and M. Desbrun (2002). Interactive Geometry Remeshing. *SIGGRAPH 2002 conference proceedings* pp. 347–354.

AutoDesk (2008). Autodesk. Disponível em http://usa.autodesk.com.

Campos, Samuel R. S., Adriana Z. Martinhago, Thomas C. de A. Oliveira, Luca Araújo Egas Prieto, Ronaldo A. Silva, Aleksander M. França and Ivayr D. F. Netto (2007). Integração do Sgbd Oracle Spatial e do Google Earth para disponibilizar informações relacionadas ao Inventário Florestal de Minas Gerais. *IX Brazilian Symposium on GeoInformatics* pp. 227–232.

Casanova, M., G. Câmara, C. Davis, L. Vinhas and G.R. Queiroz (2005). *Bancos de Dados Geográficos*. MundoGEO. Curitiba, Brasil.

Clua, Esteban W. G (2004). Impostores com Relevo. Master's thesis. PUC Rio.

Coutinho, Bruno Barcellos S., Gilson Antônio Giraldi, Antônio L. Apolinário Jr. and Paulo Sérgio S. Rodrigues (2007). Um Modelo de Simulação para Escoamento Superficial de Águas sobre Terrenos Baseado em GPU. Disponível em http://arquivosweb.lncc.br/pdfs/GPU%20Flow%20Simulation.pdf.

Döllner, J., H. Buchholz, Nienhaus and F. M., Kirsch (2005). Illustrative Visualization of 3D City Models. *Proceedings of Visualization and Data Analysis 2005 (Electronic Imaging 2005,SPIE Proceedings)* pp. 42–51.

Duran, Z. and G. Toz (2001). Obtaining 3D Information of Historical Monuments by Means of Photogrammetry. *Proceedings of Fourth International Symposium - Turkish-German Joint Geodetic Days* **1**, 277–285.

Funkhouser, T. A., C. H. Sequin and S. J. Teller (1992). Management of Large Amounts of Data in Interactive Building Walkthroughs. *Computer Graphics (1992 Symposium on Interactive 3D Graphics)* **25**, 11–20.

Hamill, J. and C. O'Sullivan (2003). Virtual dublin - a Framework for Real-Time Urban Simulation. *Journal of WSCG* **11**, 221–225.

Havemann, S. and D. Fellne (2005). A Versatile 3D Model Representation for Cultural Reconstruction. Disponível em http://citeseer.ist.psu.edu/543956.html.

Kothuri, R., S. Ravada and D. Abugov. (2002). Quadtree and R-tree Indexes in Oracle Spatial: A Comparison using GIS Data. *ACM SIGMOD 2002* pp. 546–557.

Murray, C. (2002). Oracle Spatial User's Guide and Reference Release 9.2. Disponível em http://download-uk.oracle.com/docs/cd/B10501_01/appdev.920/a96630.pdf.

Murray, C. (2003). Oracle Spatial User's Guide and Reference 10g Release 1 (10.1). Disponível em http://www.oracle.com/technology/products/spatial/pdf/qt.pdf.

MySQL (2006). Manual de Referência do MySQL 4.1.. Disponível em http://dev.mysql.com/doc/refman/4.1/pt/index.html.

Netto, A.V., J.G. Denipote and P. S. H. Cateriano (2005). Interface 3D para Manipulação de Dados em Redes de Distribuição de Energia Elétrica. *Revista Infocomp* **4**(4), 73–81.

OGC (2007). Open Geospatial Consortium. Disponível em http://www.opengeospatial.org.

Oliveira, Thomaz C. A., Samuel R. S. Campos, Luca A. E. Pietro and Elias B. C. Lasmar (2007). Indexação dos Dados Espaciais do Banco de Dados do Inventário de Minas Gerais. *Anais XIII Simpósio Brasileiro de Sensoriamento Remoto* pp. 5973–5981.

OpenAL (2006). Openal Cross-Platform 3D Audio. Disponível em http://www.openal.org.

OpenGL (2006). Opengl the Industry's Foundation for High Performance Graphics. Disponível em http://www.opengl.org.

Oracle (2005). Oracle Spatial User's Guide and Reference 10g Release 2. Disponível em http://youngcow.net/doc/oracle10g/appdev.102/b14255/toc.htm.

Oracle (2007). Oracle c++ call interface. Disponível em http://www.oracle.com/technology/tech/oci/occi/index.html.

Osfield, R. and D. Burns (2006). Openscenegraph. Disponível em http://www.openscenegraph.org.

Porto, F.A.M., J.C. Oliveira and E.S. Coutinho (2004). Algoritmo de Oclusão para Tratamento de Objetos em um SIG 3D. Technical report. Instituto Militar de Engenharia - Departamento de Engenharia de Sistemas.

PostGIS (2007). Postgis manual. Disponível em http://postgis.refractions.net.

Reiners, Dirk (2002). Scene Graph Rendering. Disponível em http://oldsite.vrjuggler.org/pub/scenegraph-rendering.ieeevr2002.pdf.

Schilling, Arne and Alexander Zipf (2003). Generation of VRML City Models for Focus Based Tour Animations. pp. 347–354.

Schneider, M. (1997). Spatial Data Types for Database Systems. Technical report. Springer-Verlag. Berlin Hidelberg.

Sharma, J. (2001). Oracle Spatial: an Oracle Technical white paper. Disponível em http://www.oracle.com.

Silva, Rosângela (2002). Bancos de dados geográficos: Uma Análise das Arquiteturas Dual (Spring) e Integrada (Oracle Spatial). Master's thesis. Escola Politécnica da Universidade de São Paulo.

Sola, J., P. H. Cateriano and A. V. Netto (2005). Processo de Rendering para Sistema 3D Interativo Utilizando Grafos de Cena. *REIC ANO V*.

Tenenbaum, A. M., Y. Langsam and Moshe J. Augenstein (1995). *Estruturas de Dados Usando C*. MAKRON Books.

Tortelli, D. M. and M. Walter (2007). Implementação da Técnica de Displacement Mapping em Hardware Gráfico. *SBGames 2007 - VI Brazilian Symposium on Computer Games and Digital Entertainment*.

USDI (2007). U. S. Geological Survey. U.S. Department of the Interior. Disponível em http://erg.usgs.gov/isb/pubs/gis_poster/index.html.

Valente, L. (2004). Representação de Cenas Tridimensionais: Grafo de Cenas. Technical report. Universidade Federal Fluminense. Niterói, Brasil.

VRML (2005). VRML Virtual Reality Modeling Languagem. Disponível em http://www.w3.org/MarkUp/VRML/.

Watson, B., N. Walker, L. Hodges and A. Worden (1996). Effectiveness of Peripheral Level of Detail Degradation when used with Head-mounted Displays. Technical report. Graphics, Visualization and Usability (GVU) Center, Georgia Institute of Technology.

Weinberger, J. (2002). The Spatial RDBMS in the Enterprise. *Directions Magazine*. Disponível em http://www.directionsmag.com/article.php?article_id=259&trv=1.

Wilmott, J., L.I. Wright, D.B. Arnold and A.M. Day (2001). Rendering of Large and Complex Urban Environments for Real-time Heritage Reconstructions. *ACM Press* pp. 111–120.

Zeev, Avi Bar (2003). Scenegraphs: Past, Present and Future. Disponível em http://www.realityprime.com/scenegraph.php.

Apêndice A

Artigo Científico Publicado

Architecture Based on Virtual Reality Techniques and Geographic Data Base for Storage and Visualization of Urban Virtual Models

R.S. Serrão
Universidade Federal do Maranhão, São Luís, MA, Brasil
A.C. Paiva
Universidade Federal do Maranhão, São Luís, MA, Brasil

ABSTRACT: Virtual Reality systems have been used in diverse areas in recent years presenting numerous benefits. One class of such systems deal with urban virtual models manipulation and visualization, that can be used for urban planning, architectural visualization and games. This paper presents conception and development of a software architecture based on geographical database that supports storage, recovery and real time visualization of urban virtual models. The proposed architecture makes intensive use of geographical databases technology to develop algorithms for these operations performance optimization. The cache strategy lead to a good performance of navigation in virtual urban models. We built an urban virtual model based on the historical center of São Luís to test the proposed architecture.

1 INTRODUCTION

One of areas that has been widely used Virtual Reality (VR) technology is the

urban environment modeling, that makes possible the realistic reconstruction of places in the world beyond the insertion of tools for its interactive visualization and simulation of diverse possible situations to happen in the real world. Moreover, the previous construction of virtual urban environment, simulating real urban environments, also it has been carried out in order to foresee and to prevent problems.

The urban virtual models have been widely used as interfaces for information systems. The VR makes possible the construction of sufficiently interactive models and the association of other components as data base, Web pages and multimedia components. Furthermore, it favors intuitive use as information repository, being useful in virtual environments on tourism, art and culture, urban planning, navigation systems, mechanical and structural tests, environmental and visual impact, meteorology, among much others. For example, streets, quarters and, even though, cities can be constructed and to divulge by Web for the virtual tourism.

Another important aspect that can be explored by urban virtual models construction is the preservation area of historic patrimony. Thus, we can use these urban models as tool for the scientific documentation of urban areas of historical interest, and also for the virtual reconstruction of monuments or not preserved or partially preserved regions. Some initiatives, aiming at to the scientific documentation of urban patrimony in digital way, have been developed around of the world. The Conference of the VSMM (Virtual Systems and Multimedia Society) opened, from 1998, a thematic session about virtual reality techniques applied to historic patrimony, as well as the Seminary of the SIGraDi (Latin American Digital Graphical society), from 1999, it started to include a session about "Digital Patrimony/ Digital Heritage". In 1996, deriving of VSMM Society, the Virtual Heritage Network was created (net for studies dissemination about the virtual patrimony). In 2002, UNESCO promoted a series of conferences to celebrate 30o anniversary of Convention for Protection of the Humanity Patrimony. The conference, occurred in Alexandria, exactly dealt with systems of geographic and multimedia information applied to historic patrimony (Paraizo et al. 2003). Consequently, we can observe the growth of virtual models use of historic patrimony areas with intention to catalogue and to preserve data about

these historical monuments.

This work considers to the conception and development of a software architecture based on geographic data base that gives support to the storage, recovery and visualization of urban virtual models in real time, prioritizing the implementation of algorithms of optimization and strategies of cache that makes possible the navigation in urban models of wide scale. It was used a urban model constructed ad hoc from a map of São Luís historical center, tumbled as humanity's historical patrimony.

2 RELATED WORKS

The use of Virtual Reality in construction of great urban models in set with information systems can serve as an important vehicle of information. Thus, appearing plus a media alternative that contributes for the preservation and the spreading of the historic patrimony. Therefore, there is necessity of a visualization program of urban models that be efficient and be associated with a geographic data base. In this way, the visualization transforms into an excellent interface metaphorfor analysis of data on database.

In (Wilmott et al. 2001) is described a package that brings a set of rendering and optimization techniques for great and complex urban environment visualization with rates of interactive pictures for second. The package was constructed based on a structure of proprietor Scene Graph developed for CHARISMATIC project (Havemann et al. 2005). The article presents the combination of Real-time Optionally Adaptation Meshes (ROAM), View Frustum Culling, Occlusion Culling and Levels of Detail (LOD) for the constructions in an single application with objective to obtain a walk in real-time in virtual model. As result, a comparison of pictures averages for second in walk of composed model of 544.002 polygons is presented, being that, without the use of optimization techniques it has obtained a rate of 1,6 frame per seconds (fps), whereas with the use of them it was gotten a rate of 35,54 fps.

A system of urban simulation that has as objective to create a large-scale imersive simulation of Dublin city is described in (Hamill et al. 2005). The navigation through environment is made in first person giving freedom for the user to go where to desire. As practically all constructions of the city had been enclosed, it was used LOD, Discarding for Occlusion and an algorithm of textures

management to optimize the system. The system was developed from a CAD model with OpenGL and OpenAL libraries (OpenGL et al. 2006), the last one for sound. The models used to simulate the constructions had been made with 3D Studio Max (AutoDesk et al. 2006), have, on average, 500 polygons, use approximately 1 MB of texture each, being that the greater has 13500 polygons and 6 MB of textures. The load and position of the models in the environment are controlled by a simple archive text. Ahead of this, the frame rate is between 25 and 60 fps, depending on the complexity of the current scene.

In (Netto et al. 2005) is presented the development of a 3D graphical interface using the virtual environments technology with the objective of assisting the decision taking in a computational system for reduction of losses in nets of distribution of electric energy. This 3D interface was constructed for the visualization of a great amount of data in a distribution system, with great carriage nets, beyond facilitating to interpretation (evaluation) of proposals solutions for the system and to allow an easy modification of the system of distribution with objective of planning these nets. In this work, it was created a SIG capable of contemplating the biggest number of information, linked to georeferencement , where it had the necessity of innumerable data storage and information about the net load of electric energy distribution, such as pole tension, substation and the feeders of energy capacity; substation and poles localization and positioning data in relation to city; information about different poles, handles and keys of found opening and closing types on electric net, etc. To give support to this great volume of information was necessary to use a spatial data base. The model construction was based on a 2D map in CAD format and aerial images of São Carlos city (SP). From the 2D map was generated the three-dimensional map. In this article the use of optimization techniquesof rendering and visualization process had not been cited.

The work (Porto et al. 2004) approaches the recovery problem of three-dimensional object models in DBMS Relational-Object, searching to optimize it with use of an occlusion algorithm developed specifically for this system. The work inserts it in a bigger development context of a support complete system to recovery and exhibition of 3D objects in AVC (Ambientes Virtuais Colaborativos) and SIG-3D applications. However, real geographic data were not used and nor it

was constructed a visualization module, being generated exit archives in AutoLIPS for visualization in AutoCAD.

In the cited examples it can be observed that although the good results presented in relation to the performance, in the two first works, the models do not incorporate a geographic data base on its architecture and therefore it does not present a practical way to add useful information regarding the constructions that compose the model all. Already the third article implements SIG but it does not approach optimization techniques in relation to rendering and visualization. The fouth article presents optimization techniques in DBMS Object-Relational, but do not use data of an extensive area for tests.

3 INVOLVED TECHNOLOGIES

In this section, we present the technologies involved in development of the model and construction of the archetype. All the work was developed in C++ in set with OpenSceneGraph (OSG) graphical library (Osfield & Burns et al. 2006), using DBMS Oracle for the storage of the database (Oracle et al. 2006).

In the development of this work, it was decided to use the OSG because it offer a complete package with great part of what it can wait of a scene graph. It brings implemented some important optimizations for a program that works with great models. OSG is a graphical library with opened code, independent of platform, writing in the programming language C++ on the graphical library OpenGL.

The OSG possess various constructed optimizations, among them it can be cited the following ones: view-frustum culling, occlusion culling, techniques of reduction of exchanges of states of the OpenGL (lazy-state-change), occlusion plan, discarding of small objects, support in detail levels, support to diverse types of archives.

As DBMS, we used the Oracle Spatial that is a spatial extension developed on the Relational-object model of DBMS Oracle. This extension is based on specifications of OpenGIS and contains a set of functionalities and procedures that allows to store, to have access, to modify and to consult spatial data of vectorial representation. The Oracle Spatial is formed by the following components (Casanova, M. et al. 2005):

- A proper model of data called MDSYS that defines storage form, syntax and semantics of supported spatial types;

  - Mechanism of spatial indexation;
  - A set of operators and functions to represent consultations, spatial junction and other operations of spatais analysis;

- Applications administrative.

The data model of Oracle Spatial consists of a elements hierarchic structure, geometries and plans of information (layers). Each plan is formed by a set of geometries, that in turn are formed by a set of elements. Each element is associated with a primitive spatial type, as point, line or polygon (with or without islands).

The bidimensional spatial types are composed by points formed for two commanded X and Y, frequently corresponding to longitude and latitude. The extension also supports storage and indexation of three-dimensional and tetradimensionais types, but the functions and operators just function for the bidimensional types. A geometry can be formed by an only element, or by a homogeneous set (multipoints, multilines or multipolygons) or heterogeneous (collection) of elements. An information plan is formed by a geometries collection that possess the same set of attributes.

Based on Relational-object model, the Spatial defines a type of object, to represent spatial data to be manipulated, call SDO_GEOMETRY. This object contains geometry in itself, coordinates, e information about its type and projection. In a spatial table, the geometry alphanumeric attributes are defined as columns of basic types (VARCHAR2, NUMBER, IT DATES, amongst others), e geometry as a column of SDO_GEOMETRY type. In spatial table, each instance of the spatial data is stored in a line, and the set of all instances of this table forms an information plan.

For the development of this work, we opted to the spatial extension of the DBMS Relational-Object Oracle for having gratuitous license for not commercial use; for the robustness of the tool and extensive documentation.

4 ARCHITECTURE PROPOSAL

The architecture (Fig. 1) proposal is based on three layers, where we have the data presentation layer, the application layer, the it lodges the system logic and finally the data layer.

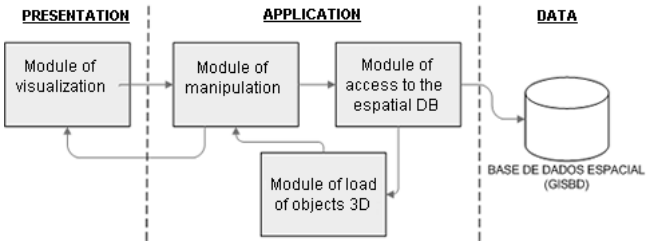


Figura A.1: General view of the architecture

The data layer is represented by an instance of a special data base, called of data source. Our architecture can deal with different sources of data, for example, Oracle, Postgresql and IBM DB2. This layer is responsible for independence of data bases in manipulation of geographic data.

The database is composed by a specific set of tables for representation of geographic data in the DBMS. The data project was constructed taking in consideration the relationship of the spatial and not-spatial entities that represent the shaped urban space. All the tables with spatial attributes (SEGMENT, SQUARE and LOT) have the fields called Geometry and Path, this used to keep the path in disk of 3D objects whereas that one stores the geometry of the spatial entity. It was also necessary the definition of not-spatial tables STREET and SEGMENT_STREET, being the first one used to store the codes and names of each street, and second to materialize the relationship between SEGMENTS and SQUARE tables. In Figure 2, the diagram of the data project is presented. It must to stand out that the 3D objects, which are visualized on application, are not stored in the bank, e yes in record with standardized name that is composed by the object class's name concatenated with the values that composes the keys of each table plus the archive extension, for example, lot1_66_3.3DS, that it is a lot of square of code 1, as code of 66 lot and code 3 that identifies the respective street.

The application layer is responsible for the logic business. It receives and treats the repassed events through the interface, it carries out the spatial transformations on the camera, from it are made spatial consultations and scene graph manipulation of the model (Fig. 3), formatting it for to be rendered and visualized in the

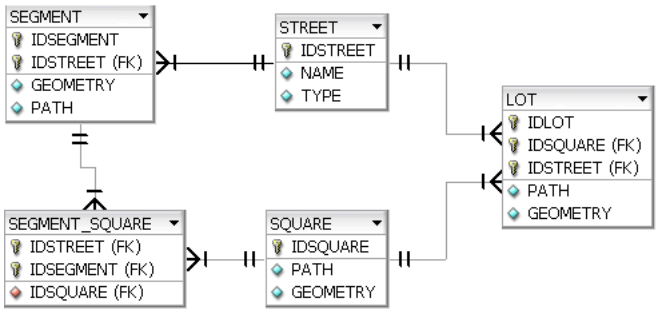


Figura A.2: Data model

presentation layer. This layer is composed for the following modules: of access to the spatial data base, of manipulation and the structure management module of the scene graph.

The access module to the spatial data base is the responsible interface for connecting the application to data base, allowing that geographic data being recognized and used in the same one.

The manipulation module is responsible for dealing with the events sent by presentation layer, to carry out the geometric transformations (translation and rotation) on the camera and to implement a strategy of pre-cache based on a buffer involving the camera, that allows to select objects to a certain distance of observer that will be added to the graph. A circular buffer involving the camera, that has parameterized ray, was adopted, opting it to this form instead of one triangular forms in order to minimize the number of insertions and object removals in the graph, not being necessary to modify the structure when the camera will be turned quickly.

To get the object list that is loaded or discarded of the scene graph is made the intersection of the area contained in the buffer with the geometries contained in spatial tables. From the first displacement of camera is made the difference between the current buffer and the previous buffer in order to get the area that makes intersection with new objects to be loaded, e also the area that contains the objects that will have to be unloaded. With this, it is not necessary making the intersection of spatial tables with allarea of the buffer, but only with the resultant areas of the differences. This procedure is clearly shown at Figure 3, where the resultant areas of this operation are represented the displacement of

the camera of the point A to point B is shown, being that area in gray involves all objects to be discarded, already the objects that are in the intersection area are kept and those that are inside of the clear yellow area will be enclosed in the graph.

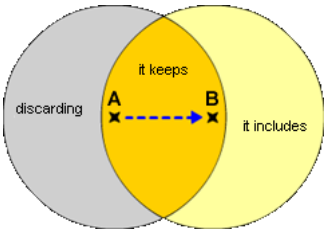


Figura A.3: Areas covered for buffers

Moreover, the management module of scene graph that carries out the insertion or discarding of objects of scene graph, it verifies to each insertion if the objects that are in the current buffer are not in the graph, hindering the redundancy in the object load. These operations with sptial tables are carried out through the operators implemented in space DBMS.

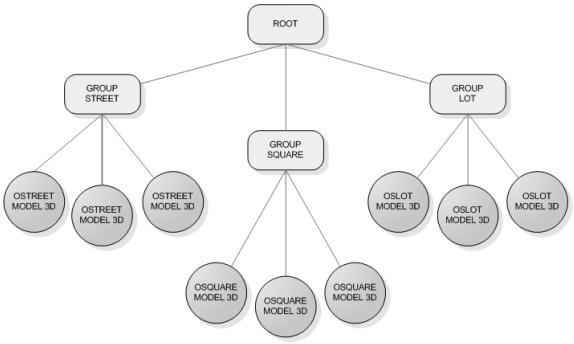


Figura A.4: Graph scene of model

In the presentation layer, we meet the visualization module that is responsible for exhibition of scene generated from the graph. It also shows, in a window, some informations about each graph node as mouse's cursor goes by them. The interface between user and application only give it for mouse, whereas when dragging pressuring the right button it applies to the camera a horizontal

displacement in all the directions.
Initially, the objects of a predetermined localization are loaded and from the first movement of camera is executed the tasks flow illustrated in Figure 5.

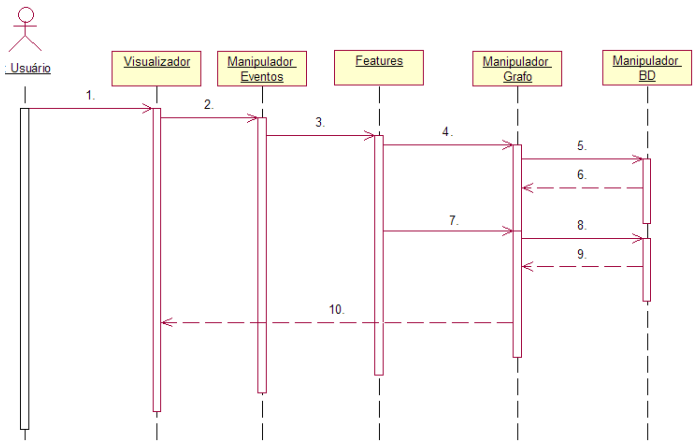


Figura A.5: Imagem com as informações do objeto 3D sendo exibidas

1. User moves the mouse;
2. Treatment of mouse's event, in this case, pressuring and dragging the right button to carry out a translation;
3. Call of method that calculates the new position of the camera;
4. Executed the method of removal of objects;
5. The objects, to be removed, are consulted in the bank according with the buffer;
6. Removed objects return;
7. Executed the load method of objects;
8. The objects, to be added in the graph, are consulted in the bank according with the buffer;
9. Added objects return;
10. Brought up to date graph and renderized scene.

5 ARCHETYPE (RevVir)

The RevVir is an archetype developed to make possible the use of proposal architecture. It consists of a viewer that beyond showing 3D objects in virtual environment also makes possible to show a set of information of each object. Also methods of drawing from the geometries stored in the bank had been implemented.

5.1 Acquisition of Data

As reference for the modeling of urban environment, the historical center of São Luís city, composed for 115 squares, 53 streets that had been divided in 205 segments of street and 962 lots. From a CAD model it was made to acquisition of the spatial data used to populate the bank with the data referring to the historical center. After the construction of the database it was implemented the responsible application for to communicate with the data base, to access the spatial data base, to make spatial research and, with its results, to generate the visualization of the georeferenciated area and the 3D models. Figure 4 shows to a 2D view of the region.

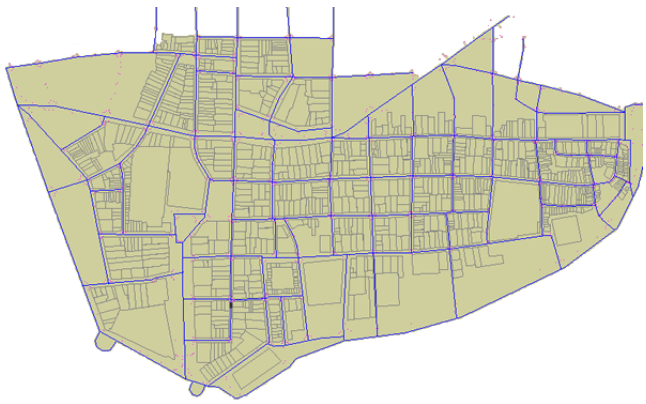


Figura A.6: Visualization 2D from the application

5.2 Object modeling 3D

Priority was given to the 3D objects modeling that represent squares and constructions loaded in the urban model. Amongst these 115 sidewalks of square, 400 buildings and the squares had been shaped . Initially tests had been made only with simplified objects to represent these volumes. For the accomplishment of 3D objects modeling is being used the tool 3D Studio Max.

The following Figures 7-8 show the current period of visualization implemented in the application.

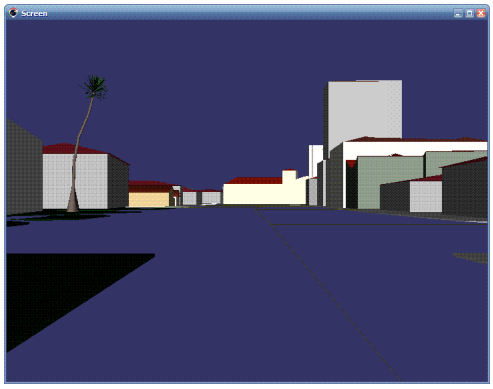


Figura A.7: Objects 3D (a)

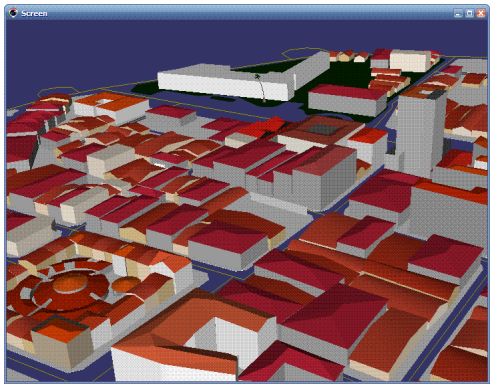


Figura A.8: Objects 3D (b)

5.3 Results

In archetype development, some performance tests to each stage of the work had been carried through. Initially, without the implementation of visualization cache, the rate of pictures per seconds was around 19 the 30, because it have none management politic of scene graph , being necessary to load all objects before the visualization starting, what it consumed reasonable hardware resources. After the inclusion of cache technique, that limited the objects number in scene graph,

the rate of pictures per seconds kept it between 30 and 60, reducing considerably the objects number in graph and the consumption of machine resources. In Figure 9, we can observe the objects grouping that are inside of a ray of 200 units around the position of the camera.

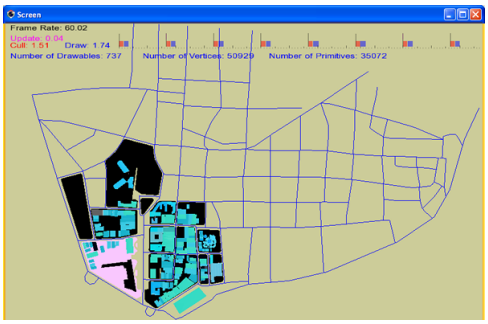


Figura A.9: Loaded objects in cache

6 CONCLUSION

In this work an architecture of software was presented based on geographic data base that propose to give support to the storage, recovery and visualization of urban virtual models in real time, using an performance optimization algorithm based on operations carried out in the spatial data base and a more efficient management of scene graph.

The proposal architecture aims at to make possible the visualization of great urban models in systems with restrictions of hardware without sacrificing the sensation of continuity on displacement through the virtual environment. Moreover, a group of information contained in a database is incorporated to virtual environment, propitiating an interesting interface man machine.

As future works, we intended to extend the architecture still more, incorporating others optimization techniques, as multiresolution and multitrheads, in order to increase the efficiency and usability of system. Moreover, it is intended to extend the data sources to make possible the construction of access modules to other spatial DBMSs.

7 REFERENCES

AutoDesk, http://usa.autodesk.com, January 2006.

Casanova, M., Câmara, G., Davis, C., Vinhas, L. & Queiroz, G.R. 2005.

Bancos de Dados Geográficos. Curitiba: MundoGEO.

Hamill, J. & OŚullivan, C. 2003. Virtual Dublin A Framework for Real-Time Urban Simulation. Journal of WSCG 11: 221-225.

Havemann, S. & Fellner, D.W. 2001. A versatile 3D Model Representation for Cultural Reconstruction. ACM Press: 205-212.

Netto, A.V., Denipote, J.G. & Cateriano, P. S.H. 2005. Interface 3D para manipulação de dados em redes de distribuição de energia elétrica. Infocomp. 4: 73-81.

OpenGL The Industry's Foundation for High Performance Graphics, http://www.opengl.org, September 2006.

Oracle Spatial, Oracle, Document Reference Manual, http://www.oracle.com/technology/products/spatial, March 2006.

Osfield, R. & Burns, D. OpenSceneGraph, http://www.openscenegraph.org, March 2006.

Paraizo, R.C. 2003. A representação do Patrimônio Urbano em hiperdocumentos: um estudo sobre o Palácio Monroe. Rio de Janeiro: Universidade Federal do Rio de Janeiro.

Porto, F.A.M., Oliveira, J.C. & Coutinho, E.S. 2004. Algoritmo de oclusão para tratamento de objetos em um SIG 3D. Rio de Janeiro: Instituto Militar de Engenharia.

Wilmott, J., Wright, L.I., Arnold, D.B. & Day, A.M. 2001. Rendering of Large and Complex Urban Environments for Real time Heritage Reconstructions. ACM Press. 111-120.