Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Curso de Ciência da Computação

VANESSA RODRIGUES COELHO LEITE

VISUALIZAÇÃO DE MODELOS URBANOS USANDO X3D E BANCO DE DADOS ESPACIAIS

São Luís
2009

VANESSA RODRIGUES COELHO LEITE

VISUALIZAÇÃO DE MODELOS URBANOS USANDO X3D E BANCO DE DADOS ESPACIAIS

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Maranhão, para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Anselmo Cardoso de Paiva.

São Luís

2009

Leite, Vanessa Rodrigues Coelho

Visualização de modelos urbanos usando X3D e banco de dados espaciais / Vanessa Rodrigues Coelho Leite. - São Luís, 2009

46f.

Impresso por computador (Fotocópia).

Orientador: Anselmo Cardoso de Paiva.

Monografia (Graduação) - Universidade Federal do Maranhão, Curso de Ciência da Computação, 2009.

1. Realidade Virtual 2. Modelo virtual urbano 3. Banco de dados geogrficos 4. X3D . I.Título.

CDU 004.946

VANESSA RODRIGUES COELHO LEITE

VISUALIZAÇÃO DE MODELOS URBANOS USANDO X3D E BANCO DE DADOS ESPACIAIS

Monografia apresentada ao Curso de Ciência da Computação da UFMA, como requisito para a obtenção parcial do grau de Bacharel em Ciência da Computação.

Aprovado em 05 de janeiro de 2010

BANCA EXAMINADORA

Anselmo Cardoso de Paiva

Doutor em Informática

Aristófanes Corrêa Silva

Doutor em Informática

Geraldo Braz Junior

Mestre em Engenharia de Eletricidade

À minha família. De sangue, de escolhas, de sonhos. Aos meus professores.

Agradecimentos

A Deus, por todas as coisas.

Aos meus pais, Andréa Coelho e Orlando Leite, pela educação que recebi.

À minha irmã, Brenda, pela companhia e pertubação para não me deixar dormir.

Aos amigos e primos, pelo apoio e por não insistirem em me convidar pra sair.

Aos professores que me orientaram em toda a minha jornada, em especial a Anselmo Paiva, por acreditar em mim e me orientar com destreza, a Aristófanes Silva, por todo o apoio e horas de conversa, a Carlos Salles, por me fazer querer ser melhor do que sou.

A todos que confiaram que eu podia ir além.

Muito obrigada.

"A vida é para quem topa qualquer parada. Não para quem pára em qualquer topada." *Bob Marley*

RESUMO

Modelos virtuais urbanos tem sido utilizados como formas de preservar o patrimônio arquitetônico ou para prever e simular situações de ambientes reais. Dessa forma utilizou-se a realidade virtual para fazer um passeio virtual pelo Centro Histórico de São Luís que é considerado patrimônio cultural da humanidade, utilizando um algoritmo de otimização da visualização baseados em Banco de Dados Espaciais. A otimização consiste no carregamento apenas dos objetos que estão sendo visualizados na cena, aumentando assim a quantidade de quadros por segundo, o que dá mais realidade à cena. A modelagem tridimensional da cidade utilizou o X3D, um padrão aberto para distribuição de conteúdo tridimensional. O X3D combina tanto características de geometria quanto descrições de comportamento, sendo descrito pelo Consórcio Web3D.

Palavras-chave: Realidade virtual. Modelo Virtual Urbano. Banco de Dados Geográficos. X3D.

Abstract

Urban virtual model have been used as ways to preserve architectural heritage or to predict and simulate situations of real environments. Thus we used virtual reality to make a virtual tour through Historic Center of São Luís that is considered world cultural heritage, using an optimization algorithm based on Spatial Database to allow the visualization of urban models. The three-dimensional modeling of the city used the X3D, an open standard for content distribution in three dimensions. The X3D combines features of both geometry and descriptions of behavior, being described by the Web3D Consortium.

Key-words: Virtual Reality, Urban Virtual Model, Geographical Databases, X3D.

Lista de Figuras

Lista de Siglas

API - Application Programming Interface (Interface para Programação de Aplicações)

AVC - Ambientes Virtuais Colaborativos

CAD - Computer-Aided Design (Projeto Auxiliado por Computador)

LOD - Level Of Details (Nível de Detalhes)

RAM - Random Access Memory (Memória de Acesso Aleatório)

RV - Realidade Virtual

SAI - Scene Access Interface (Interface de Acesso à Cena)

SDOAPI - Oracle Spatial Java Library (Biblioteca Java para Oracle Spatial)

SGBD - Sistema Gerenciador de Banco de Dados

SIG-3D - Sistema de Informações Geográficas

SQL - Structured Query Language (Linguagem de Consulta Estruturada)

URL - Uniform Resource Locator (Localizador Padrão de Recursos)

VRML - Virtual Reality Modeling Language (Linguagem para Modelagem de Realidade Virtual)

X3D - eXtensible 3D (3D Extensível)

XML - eXtensible Markup Language (Linguagem de Marcação Extensível)

Sumário

1 INTRODUÇÃO

A Realidade Virtual (RV) possui diversas definições. Por ser bastante abrangente cada definição pode envolver conceitos tecnológicos ou aspectos gerais diferentes, mas comumente representa uma participação num ambiente tridimensional gerado por computador em tempo real, sendo basicamente uma interface homem-máquina (Tori et al., 2006). A RV pemite que o usuário explore e até mesmo modifique o ambiente virtual através de características como interação, imersão e navegação (Vince, 2004).

Atualmente, é possível observar o uso de RV na maioria das áreas do conhecimento e nas mais diversas aplicações, seja educacional, médica, comercial ou científica. Componentes como banco de dados, páginas Web, conteúdo multimídia e até dispositivos móveis, possibilitam o uso da RV como um repositório de informações, sendo útil em ambientes virtuais nas áreas de turismo, arte e cultura, planejamento urbanístico e estrutural, impactos ambientais, sistemas colaborativos, entre outros.

Devido à sua capacidade de retratar o mundo real de maneira bastante realista, os modelos virtuais urbanos têm sido amplamente empregados como interfaces para sistemas de informação, capazes de retratar lugares existentes no mundo, simular situações da natureza e projetar modelos futuros a fim de prever e evitar problemas. O uso de RV com modelos de cidades possibilita uma conservação do patrimônio arquitetônico que não pode ser guardado em museus, como pinturas e esculturas, além de auxiliar serviços turísticos, de emergência e de planejamento urbano.

Graças ao avanço tecnológico-computacional, os computadores pessoais têm evoluído tanto em *hardware* e *software* quanto em dispositivos de entrada e saída (E/S) especiais, o que tem tornado a RV cada vez mais acessível. A RV *desktop*, ou seja, aquela feita para simulações em computadores pessoais, tem sido bastante utilizada pela facilidade de simulação através de monitores comuns utilizando navegadores Web de visualização 3D e *plugins*. Existem várias linguagens de descrição de mundos 3D para *desktop*, a maioria é gratuita e está disponível para qualquer usuário, como VRML e o próprio X3D. Neste trabalho utilizou-se o X3D, apresentado em 1999 pelo Consórcio

Web3D (Web3D Consortium, 2004b).

O objetivo deste trabalho é facilitar a visualização de modelos urbanos na Web, utilizando uma técnica de otimização da visualização juntamente com X3D e Banco de Dados Espaciais para o suporte ao armazenamento e recuperação dos dados, de forma a possibilitar o carregamento sob demanda dos objetos que compõem o modelo.

Como exemplo de implementação desenvolveu-se uma aplicação desta técnica baseada em um padrão aberto de distribuição de conteúdo 3D (X3D). Esta aplicação foi desenvolvida em Java, e utilizou a API (*Application Programming Interface*) SAI (*Scene Access Interface*) para fazer a comunicação com o conteúdo X3D. O SGBD utilizado foi o Oracle Spatial, que suporta as funcionalidades de um banco de dados espacial.

O restante deste trabalho está organizado nos seguintes capítulos:

O Capítulo 2, no qual é explicado a fundamentação teórica necessária para o entendimento deste trabalho. E são abordados realidade virtual, X3D, grafo de cena, banco de dados espacial, além de outros detalhes da tecnologia utilizada.

O Capítulo 3 que apresenta o algoritmo utilizado para a otimização da visualização de modelos urbanos, a arquitetura trabalhada e uma aplicação do modelo proposto por meio de um protótipo baseado no Centro Histórico de São Luís, bem como comparações entre o modelo sem otimização e com a estratégia de otimização proposta, além dos resultados obtidos.

O Capítulo 4 apresenta a conclusão do trabalho, além de sugestões para trabalhos futuros.

1.1 Trabalhos relacionados

A construção virtual de modelos urbanos tem sido bastante empregada, seja para reconstrução de cidades que não existem mais, que foram destruídas parcialmente, ou simplesmente para mostrar como determinada cidade foi um dia, por representarem um ambiente histórico que não pode ser visitado e apreciado pessoalmente. Geralmente, para reconstruir um ambiente utilizando-se realidade virtual é necessário um grande poder de processamento ou ainda equipamentos muito específicos para sua visualização. Assim, torna-se necessário o desenvolvimento de técnicas capazes de otimizar a visualização desses

modelos.

Em (Porto et al., 2004) foi desenvolvido um algoritmo de oclusão específico para o sistema desenvolvido que trata o problema da recuperação de modelos tridimensionais de objetos em SGBD (Sistema Gerenciador de Banco de Dados) Objeto Relacional. O trabalho desenvolvido faz parte de um sistema maior de apoio à recuperação e exibição de objetos 3D em aplicações de AVC (Ambientes Virtuais Colaborativos) e SIG-3D (Sistema de Informações Geográficas). Porém, não foram utilizados dados geográficos reais, não foi feito o tratamento da movimentação na cena e nem desenvolvido um módulo de visualização. O sistema cria arquivos de saída em AutoLISP (Almeida, 1996) para visualização em AutoCAD (Autodesk, 2009).

Já em (Hamill and O'Sullivan, 2003) foi construído um sistema de navegação urbana em primeira pessoa da cidade de Dublin que permite ao usuário ir aonde desejar. Nesse trabalho utilizaram quase a totalidade das construções pertencentes à cidade com aplicação de texturas. Dessa forma, para otimizar a visualização do sistema proposto, eles utilizaram técnicas como LOD (*Level of Details*), Descarte por Oclusão e um algoritmo de gerenciamento de texturas. O sistema foi desenvolvido a partir de um modelo CAD já existente com as bibliotecas OpenGL (OpenGL, 2006) e OpenAL (OpenAL, 2006). Os modelos utilizados foram feitos no 3D Studio Max, e possuem em média 500 polígonos e 1MB de textura cada, sendo que o maior possui 13.500 polígonos e 6MB de textura, que representa o Trinity College, mostrado na Figura 1.1. Este sistema apresenta bons resultados com uma taxa de quadros que varia entre 25 e 60 fps (quadros por segundo), entretanto ele não utiliza banco de dados geográficos, não aproveitando as vantagens da utilização de uma base de dados espaciais.

Figura 1.1: Vista do Trinity College em Dublin. Fonte: (Hamill and O'Sullivan, 2003)

Em (Serrão, 2008) foi proposto o algoritmo de otimização utilizado

nesse trabalho, bem como feita a modelagem do Centro Histórico de São Luís, através de um protótipo denominado Revvir. A modelagem foi feita utilizando o 3dStudioMax (AutoDesk, 2008), baseada em um modelo CAD, constituindo-se de 115 calçadas, 962 prédios e praças, e 205 segmentos de rua, totalizando 1282 objetos com 42228 primitivas e 94236 vértices. O algoritmo é baseado na densidade dos modelos urbanos e foi implementado utilizando a linguagem C++ e o OpenSceneGraph (Osfield and Burns, 2006). A otimização encontra-se no fato de exibir apenas os objetos que estão sendo visualizados, alterando constantemente o grafo de cena, inserindo os objetos que não constam no grafo e retirando aqueles que não são mais necessários. O protótipo desenvolvido atingiu bons resultados com uma média de 68 fps.

2 CONCEITOS E TECNOLOGIAS

Neste capítulo apresentam-se alguns conceitos básicos necessários para o entendimento do processo de reconstrução virtual do modelo urbano, bem como da otimização da visualização da cena. São eles: realidade virtual, X3D e Oracle Spatial.

2.1 Realidade Virtual

Nesta seção apresentam-se alguns conceitos básicos de Realidade Virtual (RV). Como foi dito anteriormente, um sistema de RV deve permitir ao usuário interagir, navegar e sentir-se imerso no sistema. Dependendo do tipo de sistema, essas características podem ocorrer em maior ou menor grau.

Atualmente é possível criar ambientes com um elevado grau de interatividade graças ao avanço da RV, que possibilita aos usuários em vez de simplesmente atuar numa aplicação com o clique de um botão, utilizar ações pertencentes ao mundo tridimensional como abrir portas, ligar interruptores, caminhar por avenidas, e etc.

A **interação** representa a capacidade do computador em captar os movimentos do usuário e transformar o mundo instantaneamente para representar a ação executada. Quanto mais imediata e mais representativa for a transformação do mundo, maior é a facilidade de cativar as pessoas. Os videogames utilizam-se desta habilidade (capacidade reativa) para se tornarem populares.

A **imersão** representa o sentimento de se estar dentro do ambiente. Um sistema imersivo é facilmente obtido com o uso de capacete de visualização, ou salas com projeções das visões nas paredes, teto, e até mesmo piso. Além da visão, a utilização da percepção gerada pelos outros sentidos também são importantes para o sentido de imersão, por isso dispositivos que trabalham com som, posicionamento da pessoa, movimentos e controle de reações tornam-se muito importantes.

A visualização tridimensional através de monitor é considerada não imersiva. Entretanto, os outros sentidos acabam dando algum grau de imersão à realidade virtual com o uso de monitores, mantendo sua caracterização e importância, e tornando-a mais

acessível a qualquer usuário (Kirner and Pinho, 1997).

A **navegação** é representada pela capacidade do usuário de movimentar-se pelo mundo. Ela pode ocorrer tanto de uma maneira pré-estabelecida, quando o usuário não tem escolha de rotas, quanto associar-se à interação e permitir que o mesmo modifique o mundo escolhendo por onde andar.

2.2 X3D - *Extensible 3D*

X3D (Web3D Consortium, 2004b) é um padrão aberto para distribuição de conteúdos 3D, que não é uma API, nem um formato de arquivo para descrever geometria. Ele contempla tanto descrições de geometria quanto de comportamentos instantâneos, num simples arquivo em vários formatos disponíveis para isso: Extensible Markup Language (XML) (W3C, 1996), Virtual Reality Modeling Language (VRML) e binário.

Foi desenvolvido a partir de uma revisão da especificação ISO VRML97 (ISO - International Organization for Standardization, 1997), e incorpora os avanços dos recursos disponíveis nos últimos dispositivos gráficos comerciais juntamente com melhorias na sua arquitetura. O núcleo da especificação do X3D está continuamente a ser desenvolvido pelo X3D Specification Working Group. Ele possui quatro níveis de funcionalidades através das várias definições de perfis (*profile*), como mostra a Figura 2.1. Cada perfil suporta todas as funcionalidades dos perfis por ele englobados. O perfil "Intercâmbio" é o perfil básico para a comunicação entre as aplicações. É o apoio da geometria, texturização, iluminação básica e animação. Não existe um modelo de tempo de execução para a renderização, o que torna muito fácil de usar e integrar em qualquer aplicação. O "Interativo" permite a interação básica com um ambiente 3D, adicionando nós sensores para a navegação e interação do usuário (por exemplo, PlanseSensor, TouchSensor etc), além de iluminação adicional (Spotlight, PointLight). Já o perfil "Imersivo" permite total interação e gráficos 3D, incluindo suporte de áudio, colisão, nevoeiro, e scripts. E o "Completo" inclui todos os nós definidos, incluindo NURBS (modelo matemático para gerar e representar curvas e superfícies), animação de humanóides e componentes geoespacial.

O padrão X3D é independente de plataforma e permite a criação de ambientes virtuais por meio dos quais o usuário pode interagir, navegar e visualizar objetos em ângulos diferentes. Ele utiliza o modelo XML visandos prover uma integração entre

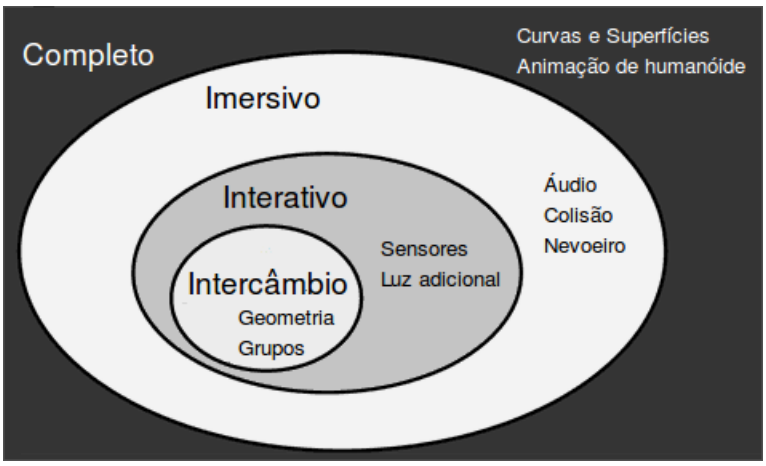

Figura 2.1: Perfis do X3D. Fonte: (Web3D Consortium, 2004a)

conteúdos 3D e a Web. Por ser baseado no VRML, o X3D herdou seus principais componentes, tais como: grafos de cena, arquitetura de eventos, sensores, prototipagem, além de adicionar componentes novos como o *export* e o *import*. Possui um sistema de coordenadas baseado na regra da mão direita, no qual os eixos x, y e z representam, respectivamente, norte, para cima e leste, conforme demonstra a Figura 2.2.

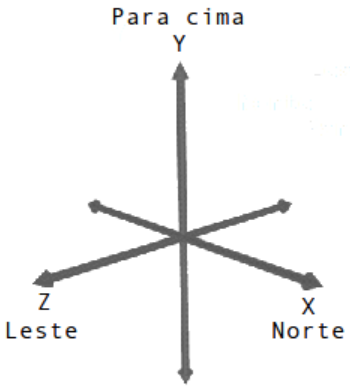

Figura 2.2: Sistema de Coordenadas

Um ambiente virtual é composto por diversas características, tanto do mundo real quanto do abstrato. Uma cena é formada por objetos que pertencem à esse ambiente que em geral são compostos por descrições geométricas, aparência e comportamento (Ferreira, 1999). Grafos são estruturas de dados que são compostas de um conjunto de nós (ou vértices) e de arcos (ou arestas) (Tanenbaum et al., 1995). Com base nessas definições é possível conceituar grafo de cena como um grafo discreto, acíclico e dirigido, ou seja, que não possui nenhum ciclo e que não permite caminhar através de

nós predecessores guardando informações diversas. Os grafos de cena, ao contrário dos grafos matemáticos, possuem nós heterogêneos, isto é, de tipos diferentes.

Computacionalmente, grafo de cena é uma estrutura que permite organizar os objetos da cena hierarquicamente. Ele possui uma raiz que agrupa todos os objetos da cena. Os nós seguintes armazenam e agrupam informações e nós relacionados, e os nós folhas armazenam a geometria dos objetos (Haro et al., 2005), como mostra a Figura 2.3.

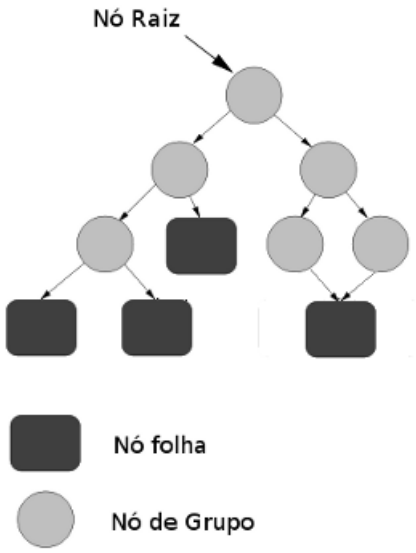

Figura 2.3: Estrutura do Grafo de Cena. Fonte: (Reiners, 2009)

O X3D possui formas básicas para o desenvolvimento de mundos virtuais como *Box*, *Sphere*, *Cone*, *Cylinder* e *Text*. Estes nós podem ser utilizados individualmente para criar formas simples, ou em conjunto dando origem a formas mais complexas. As formas simples estão representadas na Figura 2.4.

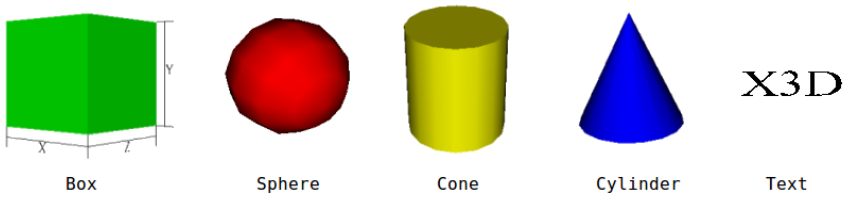

Figura 2.4: Formas básicas do X3D

Adicionalmente a esses nós de geometria, existe no X3D nós de comportamento como, por exemplo, *Group*, *Transform*, *Shape* e *Appearance*. Os nós *Group* e *Transform* são nós que servem para agrupar outros nós, entretanto este último é utilizado também para

aplicar transformações de rotação, translação e escala. Todos os nós que estão agrupados pelo nó *Transform* sofrem as mesmas transformações. O nó *Appearance* é utilizado para descrever as características de aparência dos nós como textura, transparência e cor. O nó *Shape* é utilizado para agrupar nós de aparência e de geometria.

Para a visualização do mundo virtual, o X3D disponibiliza um nó denominado *Viewpoint*. Este nó permite a configuração da posição e orientação da observação do mundo. Existe ainda o nó *NavigationInfo* que define características como o limite de visibilidade do avatar, o tipo de navegação utilizada, e as dimensões do avatar. A Figura 2.5 representa um exemplo de um arquivo X3D no qual é possível identificar o cabeçalho (1) e a descrição da cena 3D (2).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D version="3.0" profile="Immersive" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
xsd:noNamespaceSchemaLocation="http://www.web3d.org/specifications/x3d-3.0.xsd">
<head>
    <meta name="filename" content="lot1 12 3.x3d" />
    <meta name="generator" content="Blender 249" />
    <meta name="translator" content="X3D exporter v1.55 (2006/01/17)" />
</head>

<Scene>
<NavigationInfo headlight="FALSE" visibilityLimit="0.0" type='"EXAMINE","ANY"' avatarSize="0.25, 1.75, 0.75" />
<Background groundColor="0.057 0.221 0.4" skyColor="0.057 0.221 0.4" />

<Transform DEF="Exemplo" translation="-76.125854 0.000006 130.293442" scale="0.100000 0.100000 0.100000"
rotation="-0.987470 -0.111585 -0.111585 1.583405">
<Shape>
    <Appearance>
        <Material DEF="MA__01_-_Default" diffuseColor="0.761 0.137 0.224" specularColor="0.401 0.401 0.401" emissiveColor="0.0 0.0 0.0"
            ambientIntensity="0.167" shininess="0.098" transparency="0.0" />
    </Appearance>
    <IndexedFaceSet solid="true" coordIndex="0 1 2 -1, 3 4 5 -1, 6 7 8 -1, 9 10 11 -1, 12 13 14 -1, 15 16 17 -1, ">
        <Coordinate DEF="coord_Layer:PARE_01_-_Defau"
                point="563.263062 2943.400146 5.447569, 566.432434 2936.868896 5.447571, 571.152588 2944.135254 7.400038, 566.432434 2936.868896
                5.447571, 582.436096 2945.774170 5.447571, 571.152588 2944.135254 7.400038, 571.152588 2944.135254 7.400038, 582.436096 2945.774170
                5.447571, 575.383789 2946.391357 7.400628, 582.436096 2945.774170 5.447571, 579.266663 2952.305664 5.447570, 575.383789 2946.391357
                7.400628, 579.266663 2952.305664 5.447570, 563.263062 2943.400146 5.447569, 575.383789 2946.391357 7.400628, 575.383789 2946.391357
                7.400628, 563.263062 2943.400146 5.447569, 571.152588 2944.135254 7.400038, " />
    </IndexedFaceSet>
</Shape>
</Transform>
</Scene>
</X3D>
```

1

2

Figura 2.5: Exemplo de arquivo X3D

A identificação de eventos é feita com auxílio de nós sensores como por exemplo, o *TouchSensor*, *ProximitySensor* etc. Ele é capaz de detectar ações de dispositivos como o *mouse* e possui a forma do objeto que estiver agrupado com ele. As rotas são criadas através de uma declaração *Route* que define o nó de origem e o nó destino.

A Figura 2.6 representa o modelo conceitual de execução de eventos.

Apesar de suas características, o X3D não é uma linguagem de programação de propósito geral. Para permitir a criação de aplicações com requisitos não suportados pelo X3D, foi definida uma API, SAI (*Scene Access Interface*), que permite integrar cenas X3D com aplicações em outras linguagens de programação. Com a SAI uma aplicação externa pode interagir com uma cena X3D alterando características dos nós, notificando eventos, incluindo novos nós ou até mesmo excluindo os já existentes.

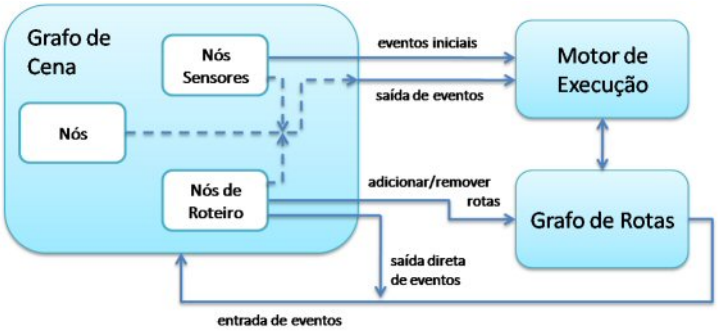

Figura 2.6: Modelo Conceitual de Execução. Fonte: (ISO - International Organization for Standardization, 2008)

2.2.1 SAI / Xj3D

Na segunda parte da especificação ISO/IEC 19975 (ISO - International Organization for Standardization, 2009) é tratada a SAI. A SAI fornece um conjunto padrão de serviços que devem ser disponibilizados por um *browser* para que o autor possa acessar o grafo de cena, suportando interações e modificações na cena. A versão utilizada neste trabalho foi a 2.0.

Conceitualmente ela permite cinco tipos de acessos para a cena 3D:

- Acessar funcionalidades do *browser*;
- Receber notificações do *browser*, como por exemplo URLs mal formadas;
- Enviar eventos para dentro da cena;
- Ler o último valor passado para campos de saída dentro da cena, e
- Receber notificações de eventos de campos dos nós que pertencem à cena.

O tratamento de eventos é também é descrito na SAI, e eles podem ter início dentro do grafo de cena como também da aplicação (*browser*). Os eventos são considerados transitórios e são gerados no momento em que ocorre a ação específica. Um evento gerado por um nó pode se propagar para outros nós através de um mecanismo de rotas. Por exemplo, a partir de alguma ação do usuário (movimento do mouse, teclado etc) um evento será gerado por um nó sensor e se este nó possuir rotas para outros nós o evento será propagado para todos eles.

Cada linguagem de programação tem a sua própria API. O Xj3D é uma implementação que segue as especificações da SAI para Java. Ele é um projeto *open*

source do Consórcio Web3D (Web3D Consortium, 2009), publicado sob a licença LGPL (Lesser General Public License). A API do Xj3D permite tanto o acesso ao grafo de cena através de um *browser* próprio quanto o desenvolvimento de um *browser* totalmente personalizado.

O *browser* é definido na SAI como o mecanismo básico para encapsular o grafo de cena e prover um conjunto mínimo de recursos para manipulá-lo dinamicamente.

Seguindo as especificações da SAI, o Xj3D permite vários modos de navegação:

WALK: O modo *WALK* é um modo obrigatório segundo a SAI. Esse modo permite ao usuário navegar pelo mundo como se andasse pelo terreno. Possui suporte à detecção de colisão e até efeitos de gravidade.

FLY: Também é um modo obrigatório. Esse modo permite ao usuário navegar no mundo como se estivesse voando. Ele possui suporte à detecção de colisão, mas não apresenta os efeitos de gravidade.

PAN: É um modo opcional no qual é possível ver o mundo em qualquer direção, sendo possível arrastar a cena. Neste modo não existe efeitos de colisão nem de gravidade.

ROLL: Modo também opcional no qual é possivel ver o mundo em qualquer direção e utilizar os movimentos da câmera nos seis graus de liberdade (rotação para os dois lados em cada eixo: X, Y e Z).

Examine: Nesse modo é possível mudar o campo de visão fazendo uma rotação no centro de rotação da camera.

LookAt: Com o LookAt é possível dar um zoom na cena, dando ênfase a determinado objeto desejado. Ao utilizar o LookAt o centro de rotação da camera é alterado.

Através dele também é possível depurar o grafo de cena, uma vez que é disponibilizado funções e métodos que permitem essa interação, bem como o tratamento de eventos tanto da cena para o *browser* quanto do inverso.

2.3 Oracle Spatial

O SGBD (Sistema Gerenciador de Banco de Dados) Oracle (Oracle, 2009) possui uma extensão desenvolvida sobre o modelo objeto-relacional denominada Spatial (Oracle, 2005), que foi utilizada neste trabalho devido à sua robustez, por ter uma extensa documentação e por ser gratuita, quando utilizada em aplicações não comerciais. Esta extensão habilita aos seus usuários um conjunto de procedimentos e funcionalidades que permitem acessar, modificar, e consultar dados espaciais em um banco de dados Oracle. Ela possui os seguintes componentes:

- Um esquema SQL (MDSYS) que prescreve o armazenamento, a sintaxe e a semântica dos tipos de suporte geométrico de dados;
- Um sistema de indexação espacial;
- Mecanismos de gerenciamento: operadores e funções para consultas, junções, e outras operações espaciais;
- Mecanismos de administração.

O Oracle Spatial possui também um modelo de dados hierarquizado composto de elementos, camadas e geometrias. Uma camada é composta por um conjunto de geometrias que, por sua vez, contém um ou mais elementos, como mostra a Figura 2.7.

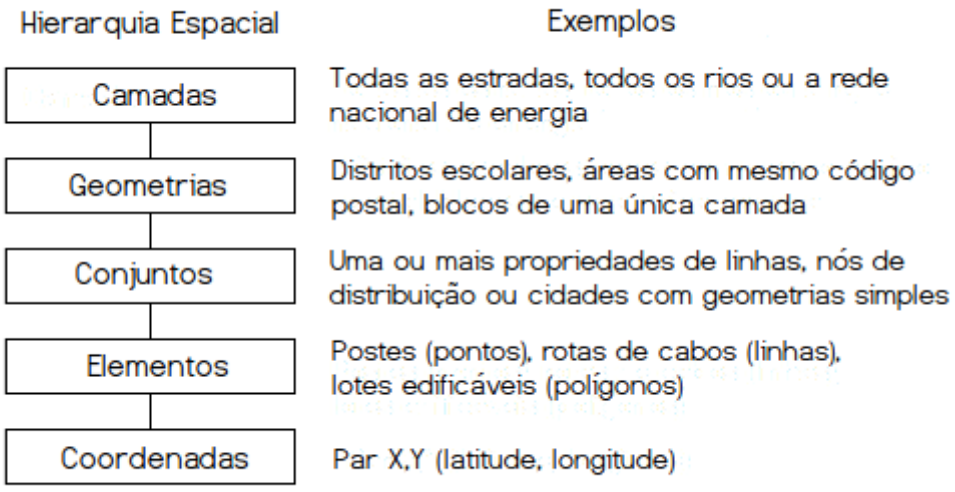

Figura 2.7: Hierarquia do Modelo Conceitual. Fonte: (Sharma, 2001)

Cada elemento é constituído por um tipo espacial primitivo, tais como ponto, linha ou polígono. Uma geometria pode ser constituída por um elemento simples ou por um conjunto de elementos. Elemento é o tipo primitivo de dado que é suportado pelo Spatial, sendo o componente básico da geometria. Ele é construído utilizando uma ou mais coordenadas, dependendo do elemento (Silva, 2002).

O Oracle Spatial suporta o modelo de dados padrão do OpenGeoespatial (OGC - Open GeoSpatial Consortium, 2008):

- ponto: elementos que possuem duas coordenadas que representam longitude (X) e latitude (Y);
- linha: elemento composto por dois ou mais pontos, que definem segmentos de linha.
- polígono: elemento composto por segmentos de linhas conectados que formam um anel fechado em seu interior.

Além de uma coleção formada por esses tipos, dando origem a arcos circulares, círculos, linhas e polígonos compostos, representados na Figura 2.8.

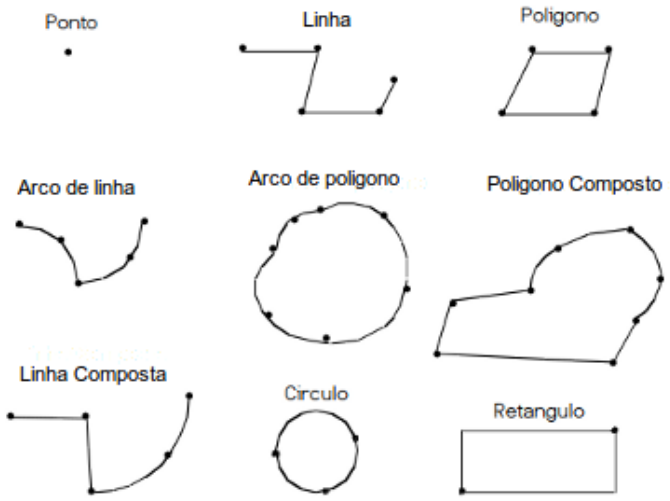

Figura 2.8: Tipos de Dados espaciais do Oracle Spatial. Fonte: (Oracle, 2005)

Para a manipulação de dados espaciais, o Oracle Spatial criou um tipo de objeto denominado SDO_GEOMETRY. É nele que são armazenadas a geometria, coordenadas e informações sobre o tipo e projeção dos objetos espaciais. Este objeto é armazenado em uma tabela espacial, onde as informações alfanuméricas estão em colunas

de tipos triviais (VARCHAR, NUM, DATE, etc), e a referente à geometria em uma coluna específica da tabela do tipo MDSYS.SDO_GEOMETRY. A tabela armazena cada instância de um objeto em uma linha, e a junção de todas essas instâncias forma uma camada.

O Oracle Spatial possui um conjunto de operadores e funções espaciais já implementadas que, juntamente com a linguagem SQL, oferecem suporte a consultas espaciais. Operadores e funções oferecem tratamento dos elementos, entretanto, possuem algumas diferenças, por exemplo, funções não utilizam índices nas tabelas espaciais, o que é obrigatório para trabalhar com os operadores. O uso de índices faz com que as consultas que utilizam operadores sejam muito mais eficientes. Outra diferença é que os operadores só podem ser utilizados em cláusulas WHERE, enquanto as funções podem ser utilizadas em cláusulas WHERE e SELECT. As operações e funções são apresentadas nas Tabelas 2.1 e 2.2 com uma descrição de suas funcionalidades.

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

Operadores Espaciais	**Descrição**
SDO_NN	Determina os vizinhos mais próximos à 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 interagem de algum modo
SDO_WITHIN_DISTANCE	Determina se uma geometria está a uma dada distância da outra

Com o Oracle Spatial é possível fazer consultas topológicas entre duas geometrias. Para isso, pode-se utilizar o operador SDO_RELATE ou a função SDO_GEOM.RELATE. Ambos utilizam o Modelo de 9-interseções definido em (Egenhofer and Herring, 1991). Este modelo considera as interseções entre os interiores, fronteiras ou exteriores de duas geometrias como vazio (0) ou não vazio (1). Tanto o operador quanto a função recebem como parâmetro o tipo de relação topológica que deve ser consultada. Os parâmetros e as descrições dos possíveis valores topológicos constam abaixo:

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

Funções Espaciais	**Descrição**
SDO_GEOM.RELATE	Determina como duas geometrias interagem
SDO_GEOM.SDO_AREA	Calcula a área de um polígono de duas dimensões
SDO_GEOM.SDO_BUFFER	Gera um *buffer*[1] ao redor de uma geometria
SDO_GEOM.SDO_DIFFERENCE	Retorna a geometria correspondente à diferença topologica entre duas geometrias
SDO_GEOM_DISTANCE	Calcula a distância 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 o perímetro de uma dada geometria
SDO_GEOM.SDO_UNION	Retorna a geometria correspondente à união topológica de 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 válida
SDO_GEOM.WITHIN_DISTANCE	Determina se uma geometria está a uma distância específica (distância Euclidiana) de outra

- EQUAL: dois objetos são iguais quando possuem a mesma fronteira e o mesmo interior
- DISJOINT: dois objetos são disjuntos quando nem a fronteira nem o interior de ambos se interceptam, ou seja, quando eles não se relacionam
- TOUCH: dois objetos se tocam quando suas fronteiras se interceptam, mas seu interior não.
- INSIDE: ocorre quando o primeiro objeto está totalmente dentro do segundo e suas fronteiras não se tocam

[1]O *buffer* gerado ao redor de uma geometria é tratado como outra geometria

- OVERLAPBDYINTERSECT: ocorre quando a fronteira e o interior dos dois objetos se interceptam (*Overlap Boundary Intersect*). Aplicado a dois polígonos.
- OVERLAPBDYDISJOINT: ocorre quando o interior de um objeto intercepta a fronteira e o interior do outro, mas as duas fronteiras não se interceptam. Aplicado a objetos do tipo linha e polígono.
- ANYINTERACT: dois objetos tem algum tipo de interação quando não são disjuntos.
- CONTAINS: ocorre quando o segundo objeto encontra-se 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).
- 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.

Para um maior entendimento segue a Figura 2.9 mostrando as relações citadas:

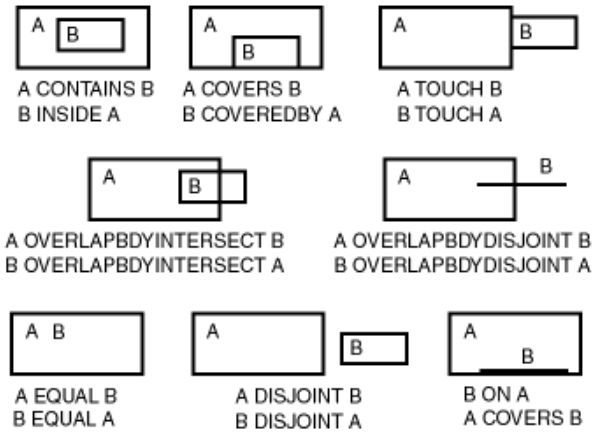

Figura 2.9: Relações Topológicas implementadas no Oracle Spatial. Fonte: (Oracle, 2005)

Para utilizar os operadores espaciais é necessário que a geometria armazenada no banco esteja indexada. O Oracle Spatial suporta essa criação de índice de duas formas: *R-tree* e *Quadtree*.

É possível criar estes índices com sintaxes SQL, e cada um deles pode ser mais ou menos apropriado dependendo da situação. Eles podem também ser usados simultaneamente para indexar uma mesma coluna com geometria.

R-trees são estruturas de dados semelhantes à *B-trees* (Cormen et al., 2001), utilizadas para indexação de informação multidimensional. São bastante utilizadas em indexação de coordenadas de dados geográficos. Uma aplicação real seria como: "encontre as lojas mais próximas de onde estou em um raio de 2 km" (Guttman, 1984). *Quadtrees* também são utilizadas em indexação de dados espaciais, diferenciando-se das *R-trees* por cada nó ter até quatro nós filhos. Elas são bastante utilizadas no particionamento de espaços bidimensionais. O índice *R-tree* pode ser usado em lugar do *Quadtree*, ou em conjunto com ele (Sharma, 2001).

As *R-trees* estão implementadas como árvores em nível lógico, mas internamente estão implementadas como tabelas do banco de dados. Para chegar à raiz dos dados são feitas buscas que envolvem SQL recursivos (Kothuri et al., 2002). Dessa maneira, as buscas tornam-se mais eficientes, uma vez que existe uma preservação de aproximações espaciais. Entretanto, a criação e atualização de índices pode se tornar um processo lento (Oliveira et al., 2007). Para cada camada de geometria a *R-tree* mantém um índice hierárquico no mínimo retângulo envolvente da geometria, conforme mostra a Figura 2.10.

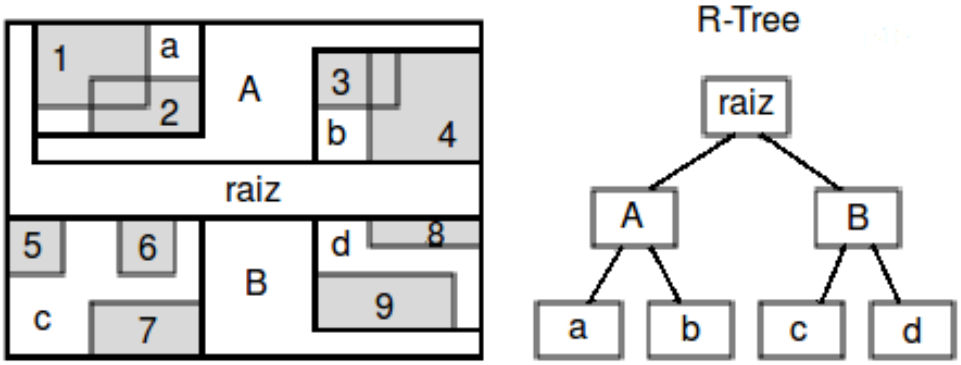

Figura 2.10: Índice Hierárquico R-tree. Fonte: (Oracle, 2005)

As buscas espaciais e operações de manipulação de dados são feitas pela *Quadtree* utilizando índices e *B-trees*. Dessa forma, a criação dos índices é simplificada e a atualização é rápida, além de prover um controle de concorrência (Kothuri et al., 2002).

Ela realiza aproximações da geometria através da aproximação de quadrantes, resultantes da divisão recursiva do espaço, que pode ser visto na Figura 2.11.

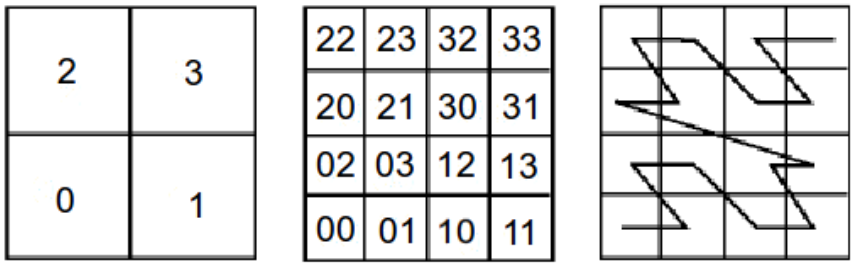

Figura 2.11: Decomposição Quadtree. Fonte: (Murray, 2003)

3 VISUALIZADOR DE MODELOS URBANOS

Este capítulo descreve o desenvolvimento do visualizador de modelos urbanos proposto por este trabalho. Em especial, serão discutidos: o algoritmo de otimização utilizado com o banco de dados espacial.

3.1 Servidor de Dados Espaciais

Neste trabalho utilizou-se um algoritmo proposto em (Serrão, 2008) que insere na cena apenas os objetos que estão a uma determinada distância da posição do observador, diminuindo assim a quantidade de recursos necessários para a visualização do modelo. É fácil observar que quanto mais objetos existirem no modelo mais eficiente essa otimização se apresenta.

O algoritmo proposto utiliza uma estratégia baseada em um *buffer* em volta da posição de cada câmera de observação, que será denominada apenas de câmera no decorrer do texto. Esse *buffer* armazena os objetos que devem ser carregados na cena em um determinado momento. Ele é uma geometria oferecida pelo banco de dados espacial, e pode ser de duas formas: triangular ou circular.

A formação triangular possui um vértice localizado na posição da câmera e os outros dois à frente, enquanto no circular a câmera é posicionada no centro, como mostra a Figura 3.1. Utilizamos então o *buffer* circular de forma a permitir uma maior movimentação do observador sem a necessidade de novas inserções na cena. Dessa forma, objetos que não estão no campo de visão do observador estão disponíveis para o caso de uma rotação "mais rápida" da câmera. Este *buffer* possui um raio que é passado como parâmetro que define a área total do mesmo.

A obtenção dos objetos que serão inseridos na cena é feita através da interseção da geometria do *buffer* com as geometrias armazenadas nas tabelas espaciais. Essa interseção é calculada por meio da utilização dos operadores espaciais adequados somados

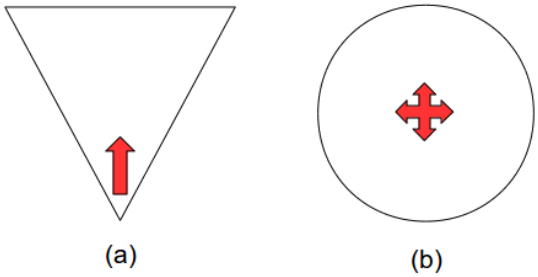

Figura 3.1: Formas de *buffers*: (a) triangular e (b) circular

às funções espaciais necessárias. Para tornar o acesso aos objetos mais eficiente, foram criados índices espaciais para cada uma das tabelas que possuem dados geográficos.

Como uma foma de diminuir ainda mais o consumo de *hardware* no tratamento com o grafo de cena, utilizou-se uma tolerância para o deslocamento do usuário na cena. Dessa forma, antes de ser feito o tratamento do grafo de cena, é feita uma checagem da posição da câmera. Se ela estiver dentro da área de tolerância, o usuário pode movimentar-se sem ser necessário inserir objetos no grafo, nem verificar se os objetos já estão na cena. A Figura 3.2 mostra a área de tolerância no deslocamento da câmera.

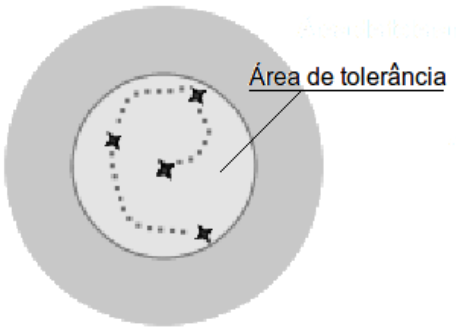

Figura 3.2: Área de Tolerância no deslocamento da câmera

3.2 Arquitetura

A arquitetura cliente-servidor é uma abordagem que separa os processos em plataformas independentes, compartilhando recursos e obtendo o maior benefício possível em cada dispositivo diferente (Vaskevitch, 1995). Seguindo esta abordagem, este trabalho propõe uma arquitetura dividida em dois módulos: um servidor de dados espaciais e uma aplicação cliente responsável por controlar a visualização da cena dinamicamente,

requisitando ao servidor de dados novos objetos 3D sempre que necessário.

O visualizador de modelos urbanos é o responsável por comunicar-se com o banco de dados espacial, apresentar a cena ao usuário e manipular os objetos baseado nos eventos gerados pelo usuário. A visualização de modelos urbanos define um ou mais pontos de observação. Cada ponto de observação define uma área circular cujos objetos devem ser carregados.

Esta proposta baseia-se no fato de que a maioria dos modelos urbanos são densos com relação à quantidades de objetos a ele pertencentes, e que, quando visualizados do solo, os prédios e casas mais próximos obstruem a visualização dos demais. Assim, em todas as direções possíveis de visualização os espaços são preenchidos pelos prédios e casas, sem ser necessário carregar um grande número de objetos. Outros modelos de visualização requerem outros algoritmos de otimização.

A arquitetura de visualização está dividida em três camadas: apresentação, aplicação e dados, demonstrada na Figura 3.3.

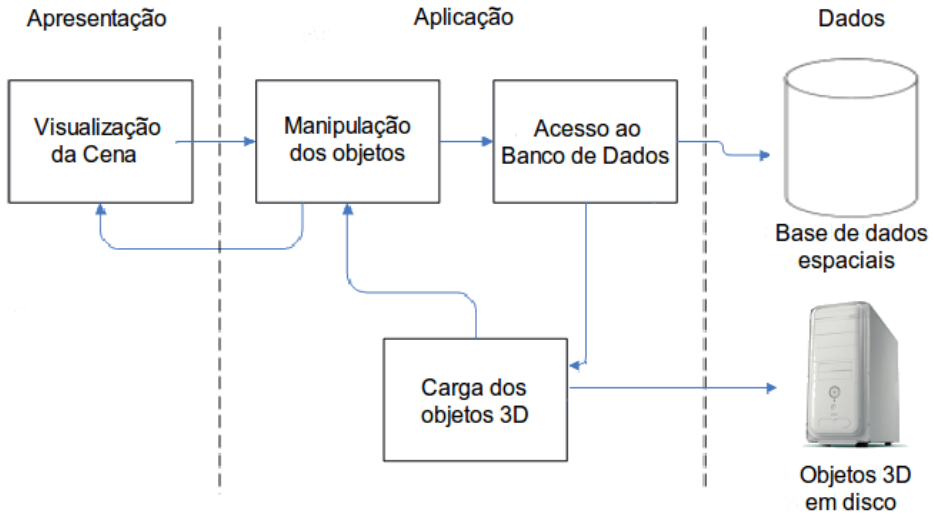

Figura 3.3: Arquitetura de visualização

A camada de apresentação é responsável pela visualização da cena, ou seja, é a interface de interação entre o usuário e o modelo de mundo desenvolvido. É nessa camada que existe o tratamento contínuo dos objetos que geram a cena enquanto a aplicação está sendo executada. É a partir dela que saem as informações do caminho que o usuário está percorrendo e que são passadas para a manipulação dos objetos, que recebe como entrada as informações da observação do usuário, e que atualiza os objetos que cercam a mesma.

Nessa camada encontra-se o módulo de visualização da cena, responsável

pela exibição do ambiente virtual gerado. A interface entre o ambiente virtual e o usuário ocorre através do *mouse* que, ao ser arrastado com o botão esquerdo pressionado movimenta a câmera passeando pela cena. Inicialmente, são carregados um conjunto de objetos pré-determinados, mas, a partir da movimentação do usuário, a inserção de objetos passa a ser dinâmica.

Já a camada de aplicação é responsável pela conexão com a base de dados para a geração das consultas, pelo processamento dos dados recebidos e obtenção dos objetos mais próximos, baseado na estratégia de *cache*. A partir de então é feita a inserção dos objetos no modelo e o tratamento dos modelos 3D que estão em disco. É nesta camada que são tratados os eventos gerados pelo usuário e é realizada a checagem dos objetos que são retornados pelo banco para identificar aqueles que já estão na cena. Compondo esta camada temos três módulos: módulo de acesso ao banco, módulo de manipulação dos objetos e módulo de carga dos objetos.

O módulo de acesso ao banco de dados é a interface responsável por conectar a aplicação ao banco de dados, através de uma referência à instância do objeto de conexão. Para isso, criou-se uma classe genérica responsável pela conexão e pelos métodos necessários à interação da aplicação com a base de dados espaciais.

O módulo de manipulação dos objetos constrói e atualiza o modelo urbano baseado nas informações recebidas pela camada de dados e trata os eventos enviados pela camada de apresentação. Por meio da estratégia de *cache*, seleciona-se os objetos que estão a determinada distância do observador, que serão adicionados à cena.

Já no módulo de carga dos objetos, realiza-se a inserção dos objetos 3D verificando sempre se os objetos armazenados pelo *buffer* já estão carregados, evitando assim redundâncias de objetos na cena, além de não sobrecarregar a renderização do ambiente virtual.

A camada de dados encapsula o acesso aos dados, tanto daqueles armazenados no banco quanto dos armazenados em disco. Ela é representada por uma instância de um banco de dados espacial, bem como pelo conjunto de arquivos armazenados em disco dos objetos 3D modelados. A base de dados é composta por um conjunto de tabelas específicas para a representação dos dados geográficos no SGBD. O esquema utilizado leva em consideração o relacionamento das entidades espaciais e não-espaciais que representam o modelo urbano.

3.2.1 Modelo Histórico de São Luís

Para a demonstração do algoritmo de otimização desenvolveu-se uma aplicação que carrega os objetos na cena de acordo com a posição do usuário. Esta aplicação é composta de um visualizador (*browser*) capaz de exibir os objetos 3D e permitir a navegação no ambiente criado. O modo de navegação utilizado é o *WALK*. Para a construção do mundo utilizou-se como modelo o Centro Histórico de São Luís[1], capital do Maranhão, localizada no Nordeste do Brasil, considerada patrimônio da humanidade pela UNESCO (UNESCO, 1997).

Para criar o *browser* utilizou-se o Xj3D, que recebe os objetos 3D, câmera e objetos que compõem o mundo. Esta aplicação possui apenas um ponto de observação. Ao redor do modelo urbano desenvolvido existe um nó de proximidade que serve para mapear os movimentos do usuário chamando as funções necessárias para a inserção dos demais objetos. A Figura 3.4 representa o *browser* criado com a visualização dos primeiros objetos carregados.

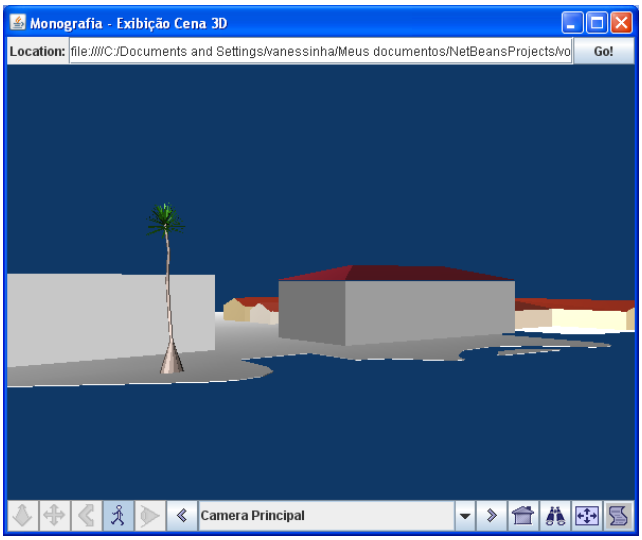

Figura 3.4: *Browser* e visualização inicial

O grafo de cena gerado é estruturado conforme pode ser visto na Figura 3.5. O grafo gerado possui um nó raiz que é composto do nó *Proximity Sensor*, responsável por mapear toda a cena, verificando continuamente se o usuário está se movimentando pelo ambiente. O nó raiz possui como filhos dois nós de agrupamentos (*Lot* e *Squares*)

[1] A modelagem tridimensional do Centro Histórico utilizada neste trabalho foi desenvolvida em (Serrão, 2008), realizando-se aqui apenas uma conversão para X3D

que contém os nós que armazenam as geometrias e demais características dos objetos 3D.

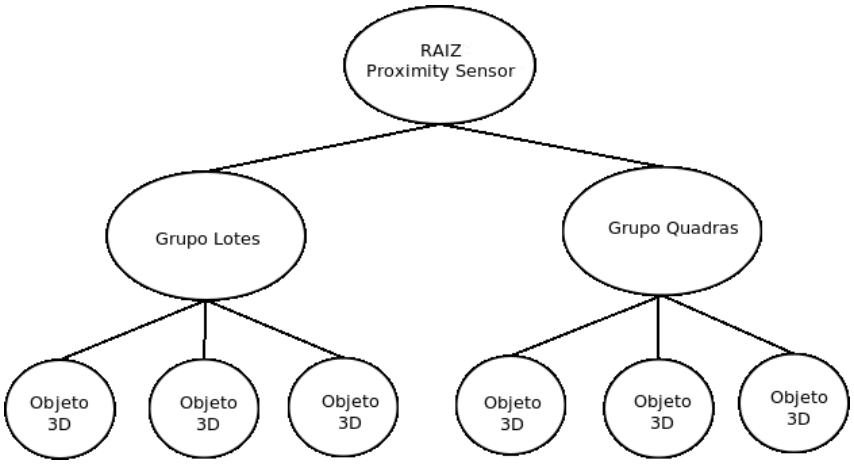

Figura 3.5: Grafo de Cena do Modelo Desenvolvido

Os objetos utilizados neste trabalho foram construídos com estruturas simples, sem aplicação de texturas. A Figura 3.6 mostra a visualização de dois pontos diferentes do modelo.

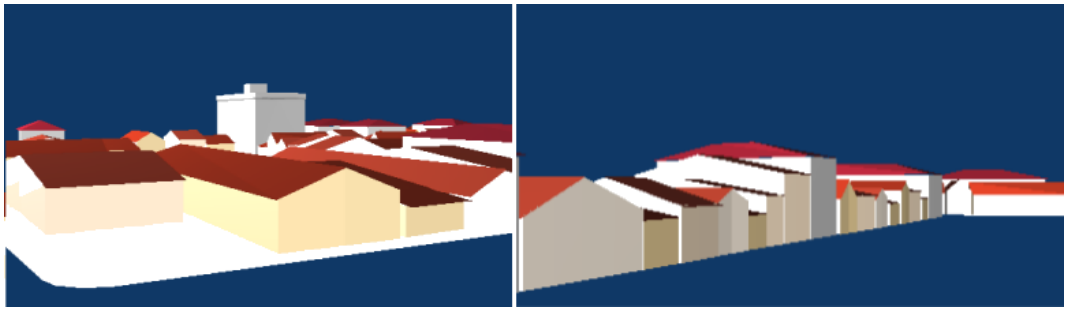

Figura 3.6: Visualização de dois pontos do modelo

Para o tratamento dos objetos que se encontravam dentro do *buffer*, utilizou-se a extensão Spacial do Oracle por meio da biblioteca SDOAPI(Oracle Spatial Java Library) (Oracle, 2004) que define os métodos para acesso aos dados espaciais deste SGBD.

No banco utilizado estão armazenadas as informações que compõem o mundo construído. Tabelas que representam estruturas tridimensionais como (*Square* e *Lot*) possuem campos para o armazenamento de informações espaciais que são específicos denominado *Geometry* e *Path*. O campo *Geometry* armazena a geometria bidimensional de cada entidade espacial enquanto o *Path* armazena as referências aos objetos 3D em disco. As tabelas não espaciais, *Street* e *Segment_Street*, também são definidas no banco.

Em *Street* são armazenados os códigos e nomes de cada rua, enquanto a *Segment_Street* materializa o relacionamento N-N entre as tabelas *Segment* e *Square*. A Figura 3.7 mostra o diagrama do esquema utilizado neste trabalho.

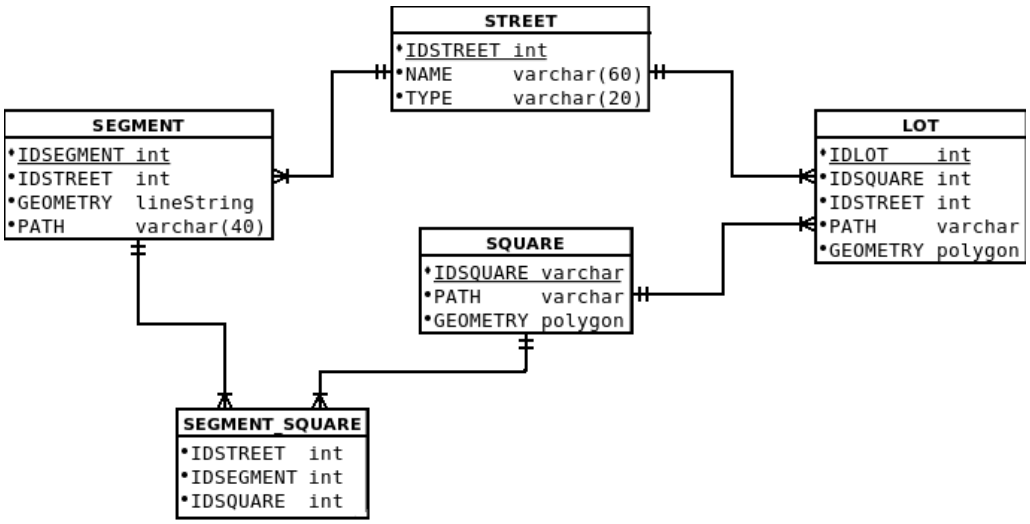

Figura 3.7: Modelagem do banco de dados

A aplicação inicia-se carregando um conjunto de objetos pré-determinados, mas, a partir da movimentação do usuário, é executada a seguinte sequência de atividades, como mostra a Figura 3.8:

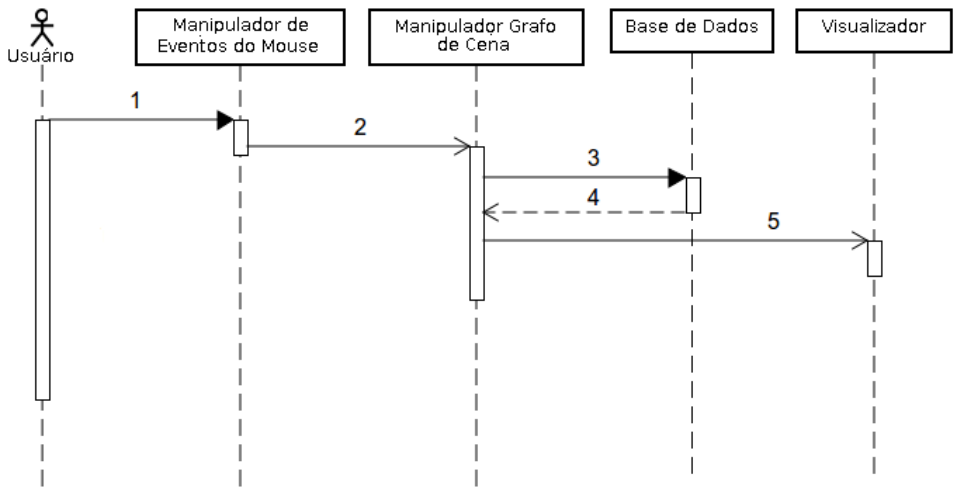

Figura 3.8: Diagrama de Sequência resumido

1. Usuário movimenta o *mouse*;

2. Tratamento do *mouse*, passando a nova posição do usuário;

3. Busca na base de dados os objetos que devem ser carregados;

4. Verifica, entre os objetos retornados, quais já estão carregados na cena;

5. Carrega na cena os novos objetos.

3.3 Resultados

Para apresentar as condições de teste do algoritmo proposto foram feitas comparações entre o modelo urbano carregado dinâmicamente e o mesmo modelo carregado totalmente no inicio do processo. Os testes foram feitos utilizando o SGBD Oracle 10g instalado em rede local e *browser* Xj3D versão 2.0. Sem utilizar nenhum tipo de otimização a taxa de quadros por segundo (fps) ficava em torno de 22 fps, pois todos os arquivos eram carregados de uma só vez. Dessa forma, eram consumidos muitos recursos de *hardware* e o pré-processamento da cena levava um tempo muito maior. A configuração de *hardware* utilizada nos testes foi: processador AMD Athlon 64 3000+ de 1.8 GHz de clock, 2048 MB de memória RAM, placa aceleradora de vídeo NVidia GeForce FX 5500 de 128 MB, rodando o sistema Operacional Windows XP.

Após a inclusão da estratégia utilizada obteve-se uma taxa média de 38 fps, no mesmo *hardware*, reduzindo consideravelmente o consumo de recursos da máquina.

Nos testes executados foi definido um percurso, do qual retirou-se a taxa de quadros por segundo em alguns pontos. A trajetória e os pontos podem ser conferidos na Figura 3.9.

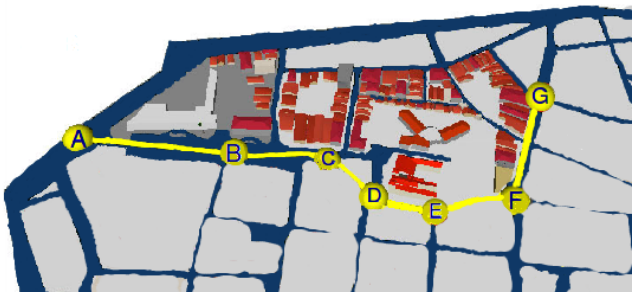

Figura 3.9: Percurso definido e pontos de amostragem

Observou-se que em alguns pontos a taxa de quadros por segundo sofreu uma

diminuição devido ao aumento do número de objetos na cena, como nos pontos C e G. A comparação entre as taxas no percurso com e sem otimização pode ser conferida no gráfico da Figura 3.10. Observa-se que sem a otimização a taxa de quadros inicia-se baixa devido ao carregamento dos objetos na cena, mas em seguida mantém-se constante por não haver mais nenhuma inserção no grafo de cena.

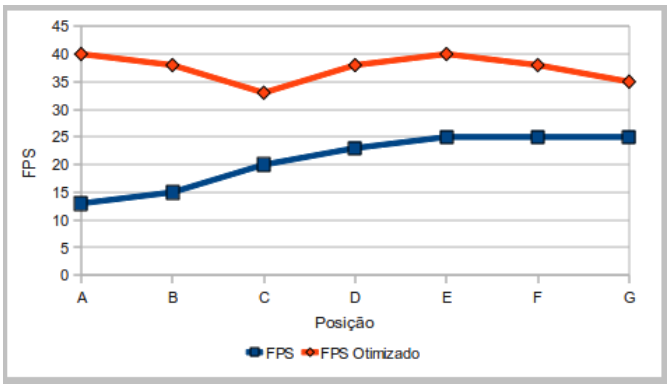

Figura 3.10: Taxa de quadros por segundo com e sem a otimização

O tamanho do raio do *buffer* foi definido com base em testes de forma a melhorar o desempenho. Foram testados *buffers* de tamanhos 100, 150, 200, 250 e 300. Para os *buffers* de 100 e 150 a taxa de fps era muito alta, entretanto o modelo urbano visualizado não era satisfatório. Já no *buffer* de raio 300 o modelo urbano era amplo, entretanto a taxa de fps caía consideravelmente. Assim, foi escolhido o raio de tamanho 200, por apresentar a melhor relação entre visualização de modelo urbano e quadros por segundo. O gráfico da Figura 3.11 representa a relação entre a taxa de quadros por segundo e o tamanho dos *buffers*.

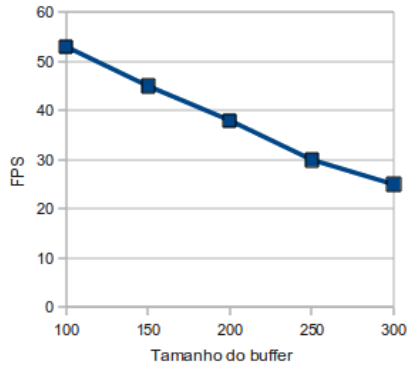

Figura 3.11: Relação entre fps x Tamanho do raio do *buffer*

4 CONCLUSÃO

Modelos virtuais urbanos têm sido utilizados com grande frequência e para os mais diversos fins. A Realidade Virtual possibilita passeios turísticos por ambientes virtuais, planejamento urbanísticos de cidades, sistemas de navegação, localização etc. Essa gama de novas possibilidades gerada pela RV tem sido motivação para inúmeras pesquisas e sistemas de desenvolvimento de grandes modelos urbanos.

No decorrer do levantamento bibliográfico deste trabalho, pôde-se notar diversas formas de construções de ambientes virtuais, utilizando-se diversas técnicas para otimizar os modelos e suas visualizações. A otimização é necessária para grandes modelos pois o custo de visualização dos mesmos ainda é muito alto, necessitando de *hardware* com alto poder de processamento e armazenamento e às vezes até mesmo de *softwares* específicos.

Os trabalhos já existentes comumente utilizam banco de dados espaciais, mas concentram suas otimizações na renderização do ambiente através de técnicas como LOD (*Level Of Details*), sem otimizar o carregamento dos objetos que compõem a cena, carregando-se apenas aqueles que o usuário pode visualizar em um determinado momento, juntamente com a utilização de banco de dados espaciais para o suporte ao armazenamento e indexação dos dados.

Dessa forma, o principal objetivo deste trabalho é propor uma técnica de otimização da visualização de grandes modelos urbanos. Esta técnica visa possibilitar que tais modelos possam ser visualizados em *hardwares* sem alto poder de processamento, de tal forma que mesmo assim a sensação de imersão no mundo não seja comprometida. Assim, é possível utilizar a arquitetura aqui desenvolvida como uma forma de aproximação da RV ao usuário comum, possibilitando a esse usuário interagir com vários novos tipos de informações, principalmente no que se refere ao ensino da preservação de patrimônios históricos.

O algoritmo desenvolvido apresentou bom desempenho no ambiente em que foi testado, mantendo um bom nível de quadros por segundo durante a movimentação do usuário. Todos os testes foram realizados com o SGBD instalado em uma rede local.

Como trabalhos futuros pretende-se estender a técnica de otimização de forma a aumentar a eficiência do sistema retirando da cena os objetos que não estão sendo visualizados e incluíndo características de LOD. Empregar a utilização de modelos digitais de terreno para a visualização de relevo, tornando o ambiente mais realista, bem como inserir textura na cena e desenvolver uma técnica de otimização para outros modos de navegação, como o de visão panorâmica. Além de incluir um *buffer* adaptativo, baseado em moldes de densidade do modelo, de forma a ser maior em áreas esparsas e menor em áreas mais densas.

Referências Bibliográficas

Almeida R. Lisp para AutoCad. Visual Books Editora, Florianópolis, SC, 1996.

AutoDesk. AutoDesk. Disponível em <http://usa.autodesk.com>, 2008.

Autodesk. AutoCAD 2010. Disponível em <http://usa.autodesk.com/adsk/servlet/pc/index?id=137 2009.

Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. 2nd Revised edition. The MIT Press, 2001.

Egenhofer MJ, Herring JR. Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases. 1991.

Ferreira AG. Uma Arquitetura para a Visualização Distribuída de Ambientes Virtuais. Master's thesis, Departamento de Informática. Pontifícia Universidade Católica - Rio de Janeiro, 1999.

Guttman A. R-Tress: A Dynamic Index Structure for Spatial Searching. In SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, June 18-21, 1984. pp. 47–57.

Hamill J, O'Sullivan C. Virtual Dublin - a Framework for Real-Time Urban Simulation. Journal of WSCG 2003; 11: 221–225.

Haro JS, Neto AV, Cateriano PSH. Processo de *rendering* para sistema 3D interativo utilizando grafos de cena. In REIC - Revista Eletrônica de Iniciação Científica, volume 5. SBC, 2005; .

ISO - International Organization for Standardization. The Virtual Reality Modeling Language. Disponível em <http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/>, 1997.

ISO - International Organization for Standardization. Information technology — Computer graphics and image processing — Extensible 3D (X3D) — Part 1: Architecture and base components. Disponível em

<http://www.web3d.org/x3d/specifications/
ISO-IEC-19775-1.2-X3D-AbstractSpecification>, 2008. Acesso em 3/10/2009.

ISO - International Organization for Standardization. Information technology — Computer graphics and image processing — Extensible 3D (X3D) — Part 2: Scene access interface (SAI). Disponível em <http://www.web3d.org/x3d/specifications/
ISO-IEC-19775-2-X3D-SceneAccessInterface/>, 2009. Acesso em 27/10/2009.

Kirner C, Pinho MS. Introdução à Realidade Virtual. Livro do Mini-curso, 1º Workshop de Realidade Virtual, São Paulo, SP, 1997.

Kothuri RKV, Ravada S, Abugov D. Quadtree and R-tree Indexes in Oracle Spatial: A Comparison using GIS Data. In Proceedings of the 2002 ACM SIGMOD international conference on Management of data. pp. 546 – 557.

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

OGC - Open GeoSpatial Consortium. OpenGis Specification. Disponível em <http://www.opengeospatial.org/standards/>, 2008.

Oliveira TCA, Campos SRS, Pietro LAE, Lasmar EBC. Indexação dos Dados Espaciais do Banco de Dados do Inventário de Minas Gerais. In Anais XIII Simpósio Brasileiro de Sensoriamento Remoto. p. 5973–5981.

OpenAL. Openal Cross-Platform 3D Audio. Disponível em <http://www.openal.org>, 2006.

OpenGL. OpenGL the Industry's Foundation for High Performance Graphics. Disponível em <http://www.opengl.org>, 2006.

Oracle. Oracle Technology Network - Spatial Software. Disponível em <http://www.oracle.com/technology/software/products/spatial/index.html>, 2004.

Oracle. Oracle Spatial User's Guide and Reference 10g Release 2. Disponível em <http://youngcow.net/doc/oracle10g/appdev.102/b14255/toc.htm>, 2005. Acesso em 27/06/2009.

Oracle. Oracle Database. Disponível em <http://www.oracle.com/us/products/database>, 2009.

Osfield R, Burns D. OpenSceneGraph. Disponível em <http://www.openscenegraph.org>, 2006.

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

Reiners D. Scene Graph Rendering. Disponível em <http://oldsite.vrjuggler.org/pub/scenegraph-rendering.ieeevr2002.pdf>, 2009. Acesso em 23/11/2009.

Serrão RS. Algoritmo de Otimização para Visualização de Modelos Urbanos com Banco de Dados Geográficos. Master's thesis, Curso de Pós-Graduação em Engenharia de Eletricidade. Universidade Federal do Maranhão, 2008.

Sharma J. Oracle Spatial: an Oracle Technical White Paper. Disponível em <http://www.oracle.com/>, 2001.

Silva R. Bancos de dados geográficos: uma análise das arquiteturas dual (Spring) e integrada (Oracle Spatial). Master's thesis, Engenharia de Transportes. Escola Politécnica da Universidade de São Paulo, 2002.

Tanenbaum AM, Langsam Y, Augenstein MJ. Estruturas de Dados Usando C. MAKRON Books, 1995.

Tori R, Kirner C, Siscoutto R. Fundamentos e Tecnologia de Realidade Virtual e Aumentada. Pré Simpósio VIII Symposium on Virtual Reality, 2006.

UNESCO. O Patrimônio Mundial no Brasil. Disponível em <http://www.brasilia.unesco.org/areas/cultura/areastematicas/patrimoniomundial/patrimonio-mundial-brasil>, 1997.

Vaskevitch D. Estratégia Cliente/Servidor. Berkeley, 1995.

Vince J. Introduction to virtual reality. Springer-Verlag, 2004.

W3C. XML Technology - W3C. Disponível em <http://www.w3.org/standards/xml/>, 1996.

Web3D Consortium. What is X3D? Disponível em <http://www.web3d.org/about/overview/>, 2004a. Acesso em 20/08/2009.

Web3D Consortium. X3D international specification standards. Disponível em <http://www.web3d.org/x3d/specifications/x3d_specification.html>, 2004b. Acesso em 20/08/2009.

Web3D Consortium. Web3D Consortium. Disponível em <http://www.web3d.org/>, 2009. Acesso em 25/03/2009.