UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
FACULDADE DE ENGENHARIA DA COMPUTAÇÃO

Realidade Virtual e Solução Numérica das Equações de Maxwell

Paulo Victor Mocbel dos Santos

BELÉM - PARÁ

2012

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
FACULDADE DE ENGENHARIA DA COMPUTAÇÃO

Paulo Victor Mocbel dos Santos

Realidade Virtual e Solução Numérica das Equações de Maxwell

Trabalho de Conclusão de Curso apresentado para obtenção do grau de Engenheiro em Engenharia da Computação, do Instituto de Tecnologia, da Faculdade de Engenharia da Computação.

BELÉM - PARÁ
2012

A adversidade leva alguns a serem vencidos e outros a baterem recordes. William Arthur Ward

Realidade Virtual e Solução Numérica das Equações de Maxwell

Este trabalho foi julgado adequado em _____ para a obtenção do Grau de Engenheiro da Computação, aprovado em sua forma pela banca examinadora que atribuiu o conceito _____.

_____

Prof. Dr. Rodrigo Melo e Silva de Oliveira

ORIENTADOR

_____

Washington César Braga de Sousa

COORIENTADOR

_____

Prof. Dr. Fabrício José Brito Barros

MEMBRO DA BANCA EXAMINADORA

_____

Prof. Dr. Josivaldo de Souza Araújo

MEMBRO DA BANCA EXAMINADORA

_____

Prof. Dr. Ádamo Lima de Santana

DIRETOR DA FACULDADE DE ENGENHARIA DA COMPUTAÇÃO

Dedico esse TCC a minha famlia e amigos, pelo apoio e incentivo, sem os quais, este trabalho no seria possvel.

Agradecimentos

Dedico esse trabalho a minha familia que sempre me apoio. Agradeço a todos os meus amigos e colegas de turma, por terem me ajudado nessa caminhada.

Paulo Victor Mocbel dos Santos

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

$\overline{E}$ Vetor intensidade de campo elétrico (V/m)

$\overline{H}$ Vetor intensidade do campo magnético (A/m)

ϵ Permissividade elétrica ($farad/m$)

μ Permeabilidade magnética ($henry/m$)

σ Condutividade elétrica ($siemen/m$)

2D Bidimensional

3D Tridimensional

FDTD Finite-difference time-domain

B-rep ou BREP Boundary Representation

RV Realidade Virtual

AV Ambiente Virtual

LANE-SAGS Synthesis and Analysis of Grounding Systems

Resumo

Neste trabalho, foi realizado o estudo de propagação de ondas eletromagnáticas em um ambiente *indoor*. Estudo esse efetuado através do desenvolvimento de uma interface que modela cenários 3D. Por meio da qual pode-se obeter uma malha compatível com o simulador que usa o método FDTD (LANE-SAGS). Com essa malha, foram feitas várias simulações, cada uma delas referente a um posicionamento diferente da antena nesse ambiente, sempre comparando as repostas simuladas com os valores esperimentais.

Capítulo 1

Introdução

Com o passar dos anos, a computação tem evoluído de forma exponencial [1], permitindo cada vez mais o ser humano: armazenar grande quantidades de dados, simular ambientes de grandes magnitudes, fazer previsão de eventos, se comunicar audiovisualmente, etc. Todo esse avanço propiciou sua expanção para quase todas as áreas da vida humana.

Mas especificamente na área de telecomunicações, pode-se dizer que esse crescimento vem possibilitando avanços nos estudos e aplicações relacionadas às simulações de ondas eletromagnéticas e antenas. O método das Diferenças Finitas no Domínio do Tempo (FDTD) [2] é um dos mais usados, e também mais antigos, nos estudos dessas propagações em ambientes *indoor* e *outdoor* [3].

Porém, muitas vezes, a representação virtual de determinados ambientes reais não é uma tarefa fácil. Dessa forma, surge a necessidade de criação de softwares que auxiliem nessa modelagem. Estes podendo ser de modelagem 2D ou 3D, que permitam a criação desde de estruturas básicas, como: triângulos, círculos, cubos, esferas , cones, pirâmides; até outras bem mais complexas(fractais e estruturas periódicas) e também gerem uma base que contenha as coordenadas de cada objetos desse cenário. Assim aproximando à área de telecomunicações ao ramo da computação gráfica, realidade virtual e engenharia de software.

1.1 Objetivos

Esse trabalho tem com finalidade a construção de um software que permita modelar ambientes tridimensionais (usando conceitos de realidade virtual) que se aproximem, da melhor forma possível, de um cenário real. A partir desse universo virtual, obter a malha compatível com o programa LANE-SAGS. Através dele, simular a propagação de ondas eletromagnéticas utilizando o método FDTD. Assim, tendo a possibilidade de analisar, por exemplo, tanto aterramento e descargas elétricas quanto a reposta desse cenário a propagação do sinal de uma antena.

Abaixo estão listados, de forma enumerada, os principais objetivos desse trabalho:

1. Representar estruturas tridimensionais por meio de alguns objetos básicos, como: cubo, esfera , cilindro, cone, etc.
2. Criar ambientes que representem virtualmente, de forma mais próxima possível, os do mundo real.
3. Gerar uma base de dados com as característica de cada estrutura. No caso específico de propagação de ondas eletromagnéticas, as características físicas dos objetos em questão são μ, σ e ϵ.
4. Gerar malha compatível com o simulador LANE-SAGS.

1.2 Organização do Trabalho

Este trabalho foi estruturado da seguinte forma:

- **Capítulo 1:** Trata de forma geral os aspectos desse projeto, como: relevância e objetivos.
- **Capítulo 2:** Esta ligado a base teórica necessária para realização do trabalho. Assim falando de computação gráfica, técnicas de modelagem de sólidos, teoria dos grafos de cena e do método FDTD.
- **Capítulo 3:** Tem como foco a interface desenvolvida nesse trabalho, suas aplicações, classes, aparência, objetos básicos, manipuladores de cena, região de análise e seus parâmetros, geração de malha, posicionamento de câmera, carregamento e salvamento da cena, atalhos de teclado, eventos de mouse, etc.

- **Capítulo 4:** Fala sobre uma aplicação usando o software com seus resultados. Seguindo todos os passos desde da criação de um ambiente ate sua simulação no LANE-SAGS.

Capítulo 2

Fundamentação Teórica

2.1 Computação Gráfica e Modelagem 3D

A computação gráfica esta presente em quase tudo na vida do homem, desde de pequenos jogos para celular até projetos grandiosos para viagens espaciais, tendo também um papel fundamental na medicina, onde possibilita a visualizações de órgãos internos ao corpo humano o que vem ajudando no tratamento de várias doenças [4] [5] [6]. Esta presente também em muitos outros segmentos, tais como os descritos na tabela 2.1.

Um dos pioneiros no quesito computação gráfica foi o aluno do MIT[1], Ivan Sutherland, que criou um programa de desenho chamado Sketchpad [7]. Através de uma caneta a laser, podia-se desenha na tela de um computador, salvar e depois recarregar o desenho feito.

Mais o primeiro computador com recursos gráficos de visualização de dados numéricos foi o **Whirlwind I**, também desenvolvido pelo **MIT**. As industrias automobilística e aero-espacial começaram a utilizá-lo , e assim esse, até então, novo campo da computação foi se desenvolvendo, abrangendo outros segmentos de atuação e com isso melhorando de forma exorbitante [8].

[1]MIT - Massachusetts Institute of Technology

Área	Aplicações
Medicina	Exames, diagnósticos, estudo, planejamento de procedimentos
Arquitetura	Perspectivas, projetos de interiores e paisagismo
Engenharia	Em todas as suas áreas (mecânica, civil, aeronáutica etc.)
Meteorologia	Previsão do tempo, reconhecimento de poluição
Segurança Pública	Definição de estratégias, treinamento, reconhecimento
Astronomia	Tratamento de imagens, modelagem de superfícies
Artes	Efeitos especiais, modelagens criativas, esculturas e pinturas

Tabela 2.1: Diferentes ramos da computação gráfica.

2.1.1 Sistemas de Coordenadas

Um sistema de coordenada é aquele que se utiliza para descrever objetos modelados em um universo. Através dele, consegue-se capturar, por exemplo, a posição e as dimensões dos objetos. Existem vários sistemas, a Figura 2.1 mostra três deles, que são: coordenadas polares, esféricas e cilíndricas. Onde o primeiro, através do raio e um ângulo podemos chegar a qualquer coordenada. Já para o segundo, obtemos as coordenadas pelo raio e dois ângulos. No terceiro, consegue-se um ponto qualquer na região de análise através do raio, um ângulo e um comprimento.

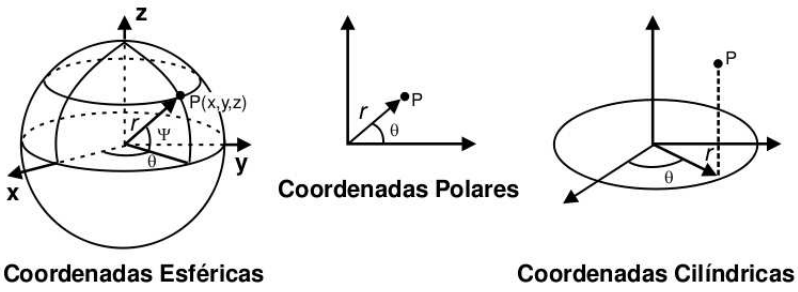


Figura 2.1: Sistemas de coordenadas.

2.1.2 Transformações

As transformações são umas das características mais importantes da computação gráfica. Elas representam um mapeamento de ponto(s) de uma determinada posição para outra [9]. As transformações mais usadas e conhecidas são: rotação, escala e translação.

Transladar significa mover o objeto. Essa operação é dada pela equação 2.1, onde adicionando quantias(T_x, T_y e T_z) a sua coordenada atual(x, y e z), obtém-se uma nova posição, transladada(x', y' e z'). A Figura 2.2 mostra essa transformação.

$$[x'\ y'\ z'] = [x\ y\ z] + [T_x\ T_y\ T_z] \tag{2.1}$$

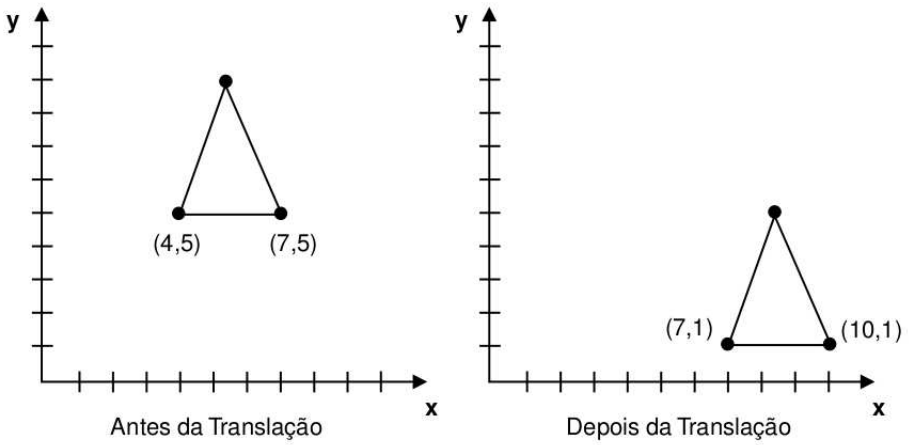


Figura 2.2: Operação de Translação 2D.

A operação de escala é aquela associada à mudança de dimensão dos objetos. Ela é representada pela equação matricial 2.2, que mostra que quando multiplicamos as dimensões atuais (x, y e z) por alguns fatores (S_x, S_y e S_z), obtemos um novo objeto escalado (x', y' e z'). A Figura 2.3 mostra bem o que ocorre quando escalonamos um objeto.

$$[x'\ y'\ z'] = [x\ y\ z] \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} = [xS_x\ yS_y\ zS_z] \tag{2.2}$$

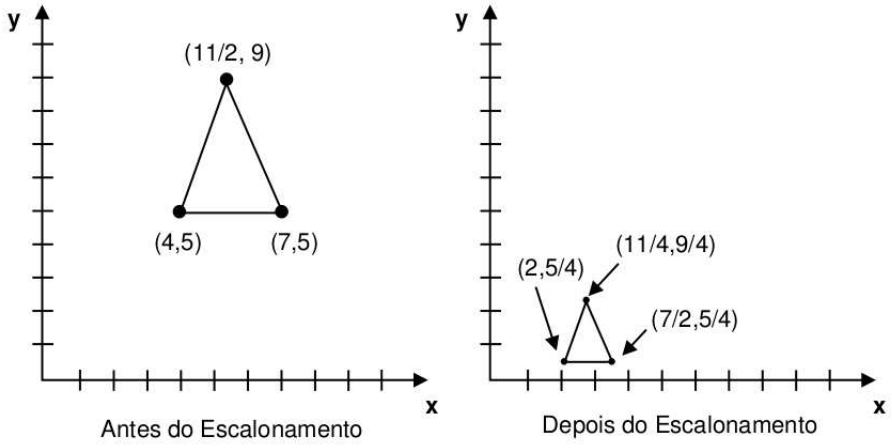


Figura 2.3: Operação de Escala 2D.

Por fim, quando desejamos girar um objeto ou um ponto no nosso sistema de coordenadas, usamos a transformação de rotação, a qual pode ser representada pelas formas matriciais 2.3, que mostram, respectivamente, as matrizes rotação para os eixos x, y e z. A Figura 2.4, ilustra essa operação.

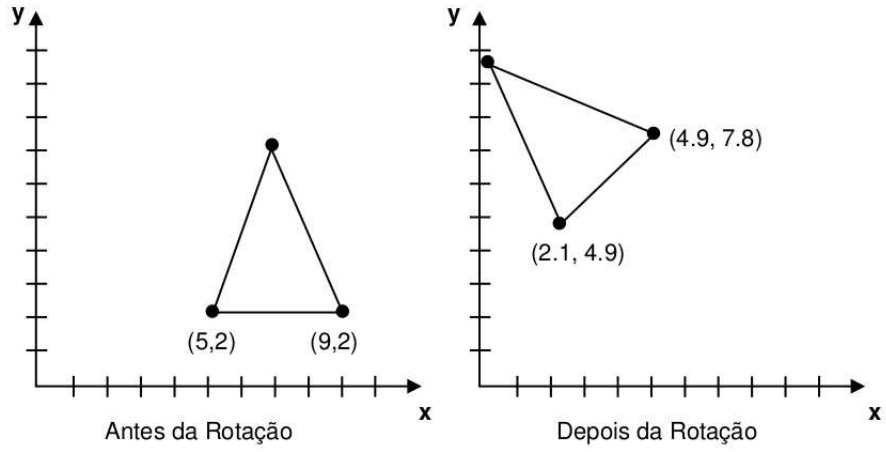


Figura 2.4: Operação de Rotação 2D.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}, R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}, R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

2.1.3 Representação de Sólidos

2.1.3.1 Representação Armada ou Wireframe

É a representação dada por um conjunto de arrestas que define as bordas do objeto [10]. Tem a vantagem de ser mais rápida quanto a renderização, porém tem um problema relacionado ao fato de dar margem para várias interpretações, além de dificultar cálculos como: volume e massa de sólidos. Por isso, ela geralmente não é considerada uma técnica de modelagem de sólidos. A Figura 2.5 esta ilustrando essa técnica.

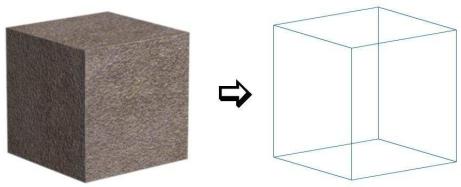


Figura 2.5: Representação Armada ou Wireframe.

2.1.3.2 Representação por Faces

Usa faces delimitantes que descrevem os contornos do sólido. É uma das formas mais usuais na modelagem de objetos tridimensionais nos dias de hoje. Ela também é conhecida com **Boundary Representation** ou **B-rep**, que consiste na descrição de objetos pelos seus contornos, ou seja, arestas e vértices. Para melhor entendimento, temos a Figura 2.6 como exemplo dessa representação.

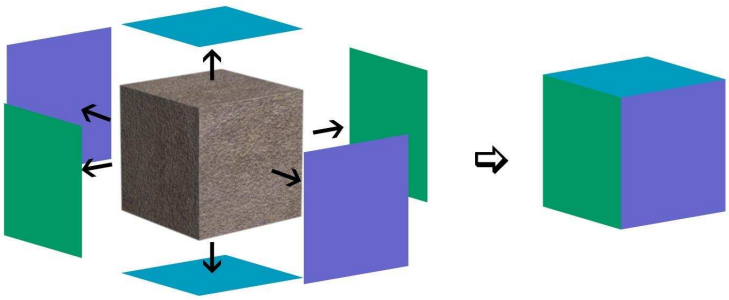


Figura 2.6: Representação por Faces.

2.1.3.3 Representação Poligonal

Polígono vem do grego *polys*, que significa muitos , e *gonos* que significa ângulos, assim esse representação é formada por figuras planas com muitos seguimentos de reta e muitos ângulos. Então podemos construir sólidos com o polígono mais simples, o triângulo, até um mais complexo, com um número elevado de lados(Figura 2.7). Essa técnica pode ser considerada uma especificação do caso apresentado na seção 2.1.3.2.

Tesselation é um das características dessa representação, significa preencher uma dada região através de várias repetições de um mesmo polígono até não haver mais espaços em ”branco”. A maioria das máquinas de jogos utilizam a representação por faces triangulares. Isso se deve ao fato de esta necessitar de menos processamento e também por possibilitar a representação de qualquer tipo de contorno [11].

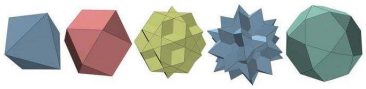


Figura 2.7: Representação Poligonal.

2.1.4 Técnicas de Modelagem Geométrica

Quanto ao âmbito da modelagem de objetos tridimensionais, tem-se basicamente três categorias: a manual, matemática e automática.

O método manual foi a base de toda a modelagem atual. É a técnica mais barata e mais antiga. Utiliza-se basicamente das medidas do modelo real e, é claro, da habilidade do

modelador. Já à matemática utiliza de algoritmos para gerar o objeto desejado. Esse método é muito utilizado para modelar, por exemplo, a proliferação de organismos microscópicos.

Modelagem automática é o método mais recente. Utiliza-se de scanners muitos sofisticados para obter o modelo tridimensional de qualquer ambiente ou sólido.

2.1.4.1 Combinação de Objetos

É um dos métodos mais intuitivos e antigos. É realizado através da combinação de um ou mais sólidos básico para obtenção de um outro [12], mostrado na Figura 2.8. O único detalhe que deve-se prestar bastante atenção nessa técnica, é a questão das operações booleanas(interseção, união e subtração) em alguns casos não gerarem como resultado um sólido, por exemplo, no caso da intercessão de dois cubos que não estão em contato, temos como resultado o vazio que não é um sólido válido.

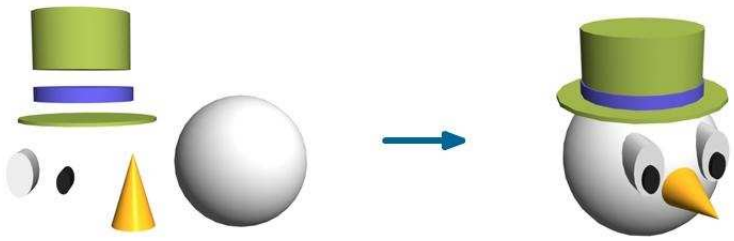


Figura 2.8: Modelagem por Combinação de Objetos.

2.1.4.2 Modelagem por Varredura(Sweep)

A modelagem por varredura é obtida pela movimentação de uma curva N_1 na trajetória de uma outra curva N_2 que descreverá um superfície que poderá ser usada como sólido. Sendo N_1 nomeada de *Curva de Contorno* e N_2 de *Caminho ou Diretriz* [13].

Ainda no questão da varredura, pode-se sitar dois casos particulares que são: varredura por *Extrusão*(Figura 2.9) ou *Translacional*(Figura 2.10) e a *Rotacional*. Para o a primeira é a translação de uma superfície na forma circular, por exemplo , ao longo de uma direção, gerando um sólido(no caso um cilindro). No segundo caso ocorre a rotação de uma superfície ao longo

de um eixo ou ponto de referência, resultando também em um sólido.

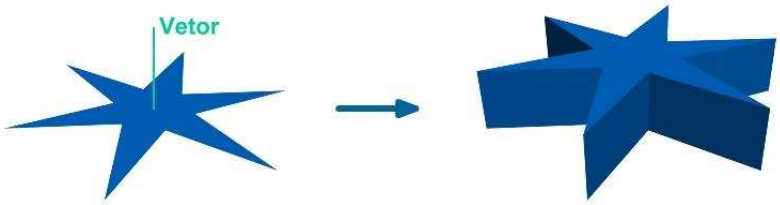


Figura 2.9: Modelagem por Extrusão.

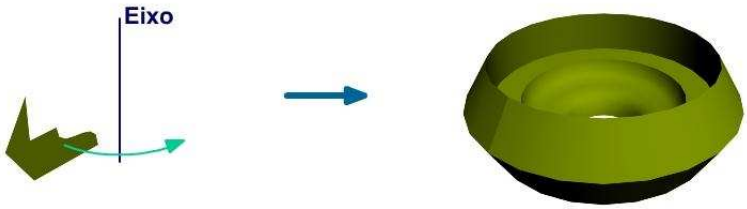


Figura 2.10: Modelagem Rotacional.

2.1.5 Realidade Virtual

O termo Realidade Virtual surgiu na década de 80 pelo artista e cientistas da computação, Jaron Lanier, que uniu os dois universos, que até então eram, muito distantes, o mundo real e o virtual [14]. Porém há registro de trabalhos anteriores a essa denominação, dentre eles esta o primeiro capacete que permitia imersão, e também pode-se falar do famoso SENSO-RAMA [15](que não obteve um grande sucesso comercial, pelo seu valor, mas com certeza foi um dos trabalhos que deram impulso para o futuro dos ambientes virtuais).

Mas o que é Realidade Virtual? É uma interface avançada que permite o usuário, navegar, modificar, interagir em tempo real com uma determinada aplicação. Existem basicamente dois tipos de realidade virtual quanto o fator imersão a imersiva e a não imersiva. A primeira é caraterizada por permitir ao usuário se sentir dentro do ambiente(através do uso de capacetes

especiais, óculos, luvas, roupas, caves [16], sensores, entre outros dispositivos), já a segunda isso não ocorre(onde usa-se mouse, teclado, monitores , etc) [17].

2.1.5.1 Grafo de Cena

É um dos conceitos mais importantes da teoria de jogos, e também da computação gráfica. Sua definição é a de ferramenta para representação de ambientes virtuais tridimensionais. Sendo esse ambiente descrito por diversos aspectos, dentre eles: descrição geométrica, câmera, transformação, aparência, comportamento e iluminação [18]. O cenário é formado por todos esses aspectos inseridos dentro do grafo de cena.

O grafo de cena é formado por nós, compondo arrestas que tem como resultado um grafo acíclico direcionado. Com cada nó tendo atributos ,sitados anteriormente, que poderão ou não influenciar nos outros nós que estão conectados a ele. Existe uma classificação para eles, que os divide em três categorias: nós raiz(é o primeiro nó, no qual todos os outros estão ligados de forma direta ou não) ,nós internos (geralmente contem informações de transformações 3D - rotação, escala e translação) e por fim estão os nós folha(que por padrão contem as dados de representação geométrica dos objetos da cena).

A propriedade fundamental dessa ferramenta é o que se chama de herança de estado. Que diz que cada nó deve herda as propriedades de estado do seus ancestrais até o nó raiz. Então se tivermos um grafo de cena formado por um *nó* raiz, casa, e dois *nós folha*, sendo o primeiro uma cadeira e o segundo mesa. Se rotacionarmos o *nó* casa, por consequência teremos a cadeira e a mesa rotacionadas também. Porém no caso da rotação de um dos *nós folha*, o *nó* casa permanecerá inalterado, haja vista que ele esta hierarquicamente no nível acima e a cadeira também não se moverá pelo fato de ter uma ligação direta com esse nó(esta no mesmo nível e não tem conexão direta). Na tabela 2.2 estão algumas vantagens proporcionadas pelo uso dessa técnica.

Vantagens	
Produtividade	Gerencia e reduz o numero de linhas que seriam necessárias para implementar a mesma funcionalidade utilizando uma interface de baixo nível, como a OpenGL.
Portabilidade	Encapsula todas as tarefas de baixo nível, reduzindo e até excluindo a parte de código que é especifica de uma plataforma.
Escalabilidade	Possibilita trabalhar tanto em máquinas com configurações básicas até supercomputadores.

Tabela 2.2: Vantagens proporcionadas pelo uso do grafo de cena.

2.2 Irrlicht

O Irrlicht é uma máquina de jogo de alta performasse escrita em linguagem C++ [19]. Contem todas as característica que pode-se encontrar em qualquer outra *engine*[2]. Existem muitos projetos desenvolvidos e em desenvolvimento usando-a, com uma boa comunidade, onde pode-se tirar dúvidas e publicar trabalhos e melhorias. E além de tudo isso, é completamente livre [20].

Ela tem como principais características:

1. Renderização em tempo real de alta performasse usando **Direct3D** e **OpenGL**.
2. Independe de plataforma.
3. Renderiza diretamente de arquivos, tais como : .zip, .pak, .pk3, etc.
4. Rápida e fácil detecção de colisão.
5. Possibilita o uso de vários linguagens de programação, como : Java, Delphi, entre outras.
6. Limpa, fácil de entender, e bem documentada.

[2]Máquina de jogo.

2.3 Método FDTD

O termo Método FDTD(Finite-Differene Time-Domain Method), primeiramente chamado de como Algorítimo de Yee, foi criado por Kane Yee em 1966 [21]. É uma técnica capaz de solucionar, de forma simples e elegante, numericamente as equações de Maxwell, de maneira direta. Tem como características principais: 1)distribuição geométrica discreta das componentes de campos Elétrico e Magnético em células, de maneira a satisfazer tanto a Lei de Faraday quanto a Lei de Ampère nas formas diferencial e integral e 2)aproximação das derivadas espaciais e temporais por diferenças finitas, de forma a se obterem equações explicitas para a atualização temporal de todas as componentes de campo [22].

Porém, por algum tempo, esse método foi deixado de lado devido ao fato de ter um custo computacional muito alto, aliado a isso, deve-se levar em consideração o contexto histórico em que ele se encontrava, onde os computadores ainda eram bem limitados. Mas, um tempo depois, ressurgiu através dos trabalhos de dois cientistas, chamados Allen e Brodwin, que aplicaram essa técnica para problemas tridimensionais relacionados à interação eletromagnética com meios materiais [23]. Assim, novas técnicas apareceram e melhoram a abordagem do método, acompanhado , é claro, da evolução dos computadores.

Essa técnica tem sido usada desde então, em uma grande quantidade de aplicações, entre elas estão: radares, antenas, fotônica, sistemas de telecomunicação, medicina, estruturas periódicas, sistemas de aterramento, entre várias outras.

O método FDTD aplicado com intuito de simular uma propagação eletromagnética, utiliza as equações de Maxwell na forma diferencial.

$$\overline{\nabla} \times \overline{E} = -\mu \frac{\partial \overline{H}}{\partial t}, \tag{2.4}$$

$$\overline{\nabla} \times \overline{H} = \epsilon \frac{\partial \overline{E}}{\partial t} + \overline{J}, \tag{2.5}$$

Onde:

$\overline{J} = \sigma \overline{E}$, vetor densidade de corrente elétrica de condução(ampere/m^2).

As equações (2.4) e (2.5) expandidas em coordenadas retangulares, geram as respectivas

equações escalares, mostradas abaixo.

$$\frac{\partial H_x}{\partial t} = \frac{1}{\mu}(\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y}), \tag{2.6}$$

$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu}(\frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z}), \tag{2.7}$$

$$\frac{\partial H_z}{\partial t} = \frac{1}{\mu}(\frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x}), \tag{2.8}$$

$$\frac{\partial E_x}{\partial t} = \frac{1}{\epsilon}(\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} - \sigma E_x), \tag{2.9}$$

$$\frac{\partial E_y}{\partial t} = \frac{1}{\epsilon}(\frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} - \sigma E_y), \tag{2.10}$$

$$\frac{\partial E_z}{\partial t} = \frac{1}{\epsilon}(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} - \sigma E_z) \tag{2.11}$$

sendo:

E_x, E_y, E_z e H_x, H_y, H_z representam as componentes dos campos elétrico $\overline{E}$ e magnético $\overline{H}$, respectivamente. Essas componentes são funções do tempo t e das três coordenadas cartesianas (x, y, z).

A lei de Faraday, equação (2.4), informa que quando há variação no tempo do vetor $\overline{B} = \mu\overline{H}$ (vetor densidade de fluxo magnético, em teslas), surgem componentes de campo elétrico circulando em torno da(s) componente(s) deste vetor. Já a lei de Ampère, equação (2.5), informa que quando há variação no tempo do vetor $\overline{D} = \epsilon\overline{E}$ (vetor densidade de fluxo elétrico, em) em certa direção e/ou a presença da fonte de corrente $\overline{J}$, esta causa circulação de campo magnético em torno da direção da(s) componente(s) deste vetor.

Kane Yee se baseou nessa duas observações para definir seu esquema de distribuição espacial e temporal das componentes de campo de seu algoritmo. Sua representação geométrica, chamada de célula de Yee, esta mostrada na Figura 2.11 [22].

Para que possam ser realizadas simulações de propagação de onda em FDTD, é criada uma malha 3D formada por múltiplas células de Yee que preenche toda região de análise, Figura2.12 [24], com a finalidade de mostrar as componentes de campo $\overline{E}$ e $\overline{H}$ atuantes em cada ponto da malha desta região.

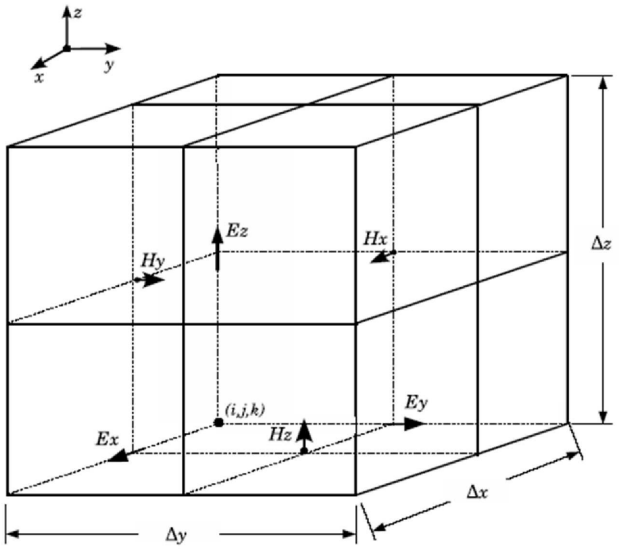


Figura 2.11: Célula Yee.

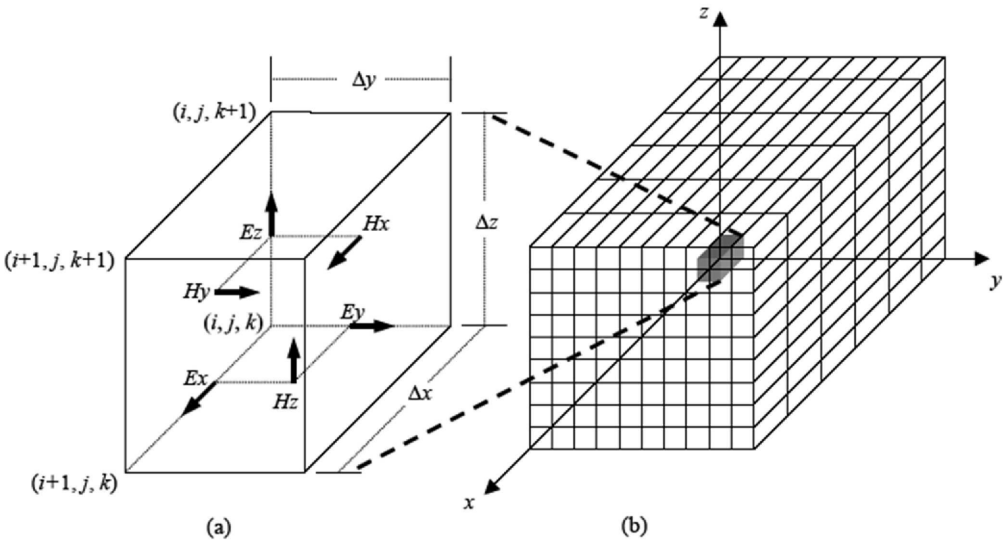


Figura 2.12: (a) Posição das componentes dos campos elétrico e magnético em uma célula de Yee;(b) Célula no interior de uma malha 3-D.

A referência a cada célula que compõe a malha, assim com as suas respectivas componentes de campo, é feita de forma discreta pelos índices i, j e k, de forma que uma determinada posição x, y, z (em metros) é referenciada no espaço discreto por i, j, k (número da célula correspondente). Estas referências são obtidas através das relações $x = i.\Delta_x, y = j.\Delta_y$ e $z = k.\Delta_z$, onde Δ_x, Δ_y e Δ_z são as dimensões das células de Yee(Figura2.11).

Capítulo 3

Desenvolvimento do Software

3.0.1 Tecnologias de Programação Usadas

As tecnologias de programação usadas foram: o motor gráfico de jogos irrlicht, a plataforma Qt-Creator e a linguagem C++. As informações sobre a engine irrlicht estão na seção 2.2. O Qt-Creator é um poderoso ambiente multi plataforma que permite a criação de diversas aplicações web e também desktop. O uso dele, neste trabalho, foi voltado para a criação de uma interface desktop, a qual pudesse ser usada em diferentes sistemas operacionais(Linux, Windows ou Macintoch) e com arquiteturas dissemelhante(32 ou 64 bits). Tanto essa plataforma quanto a linguagem foram escolhidas por serem muito usadas pelo comunidade desenvolvedora, por serem robustas e terem uma ótima documentação.

3.0.2 Conexão Qt-Irrlicht

O primeiro passo no desenvolvimento desse trabalho foi a união do universo irrlicht com a plataforma Qt-Creator. Para tanto foi necessário o estudo das classes e funcionalidades básicas desse dois mundos.

A ideia era colocar o motor gráfico irrlicht funcionando como uma janela dentro da interface gerenciada pelo qt. Assim criou-se a classe base de todo esse projeto, chamada IrrViewer. Ela contem os métodos base de uma classe Widget[1], que são : paintEvent(), resizeEvent() e paintEngine(); e também os necessários para a criação de um cenário 3D irrlicht, os

[1] Janela padrão do qt.

quais são : createIrrlichtDevice()(responsável pela criação do *device*[2], *smgr*[3] e *video-driver*[4]), cenaIrrlicht()(método que tem a finalidade de criar a cena), drawIrrlichtScene()(método de pintura).

Assim, com o Qt-Creator reconhecendo essa nova classe como uma janela(Widget) normal, pode-se passar para outra fase proposta, que era a criação de outras classes e métodos necessários para concretização do requisitos pré-estabelecidos para esse trabalho.

3.0.3 Classes

As classes criadas para este projeto foram: *IrrViewer, Cena, IrrNode, ReceiverEvent* e *MainWindow*. A primeira é a base de todo o desenvolvimento do software produzido nesse projeto, é a que conectar o irrlicht com o Qt. A classe *Cena* contem e controla tudo referente a cena,tal como: seleção, inserção, modificação e deleção de *nós*; Assim como também a geração de malha e eventos de mouse e teclado. A terceira classe é responsável pela criação de objetos: prisma, cubo, esfera, cone, haste, cilindro e antena.

Já a classe *ReceiverEvent* tem como finalidade receber os eventos de mouse do Qt e repassá-los para a engine irrlicht. Por fim, temos a classe MainWindow é responsável pela aparência do software, onde se encontram os botões com seus *signals and slots* [25].

3.1 Interface

A interface feita nesse projeto esta ilustrada na Figura 3.1. Ela foi baseada nos softwares de modelagem mais usados no mercado, que são : Blender, 3DStudio e Maya. A sua estrutura foi dividida em 4 áreas: barra superior, janela da cena, painéis laterais e barra inferior.

Na barra superior, Figura 3.2, encontramos primeiramente o botão *New*, o qual ativará uma nova cena irrlicht. Quando pressionado ativara uma janela (Figura 3.3) que requisitará os dados necessários para criação da região de análise, tais como dimensão x, y, z e o valor do $delta$ (dimensão da célula de Yee). Em seguida vem o botão *Open*, que carrega

[2]Responsável por todas as informações da cena

[3]Gerenciador de cena, ele que responsável por adicionar, remover, clonar e capturar objetos da cena.

[4]Responsável por tudo referente ao driver de video

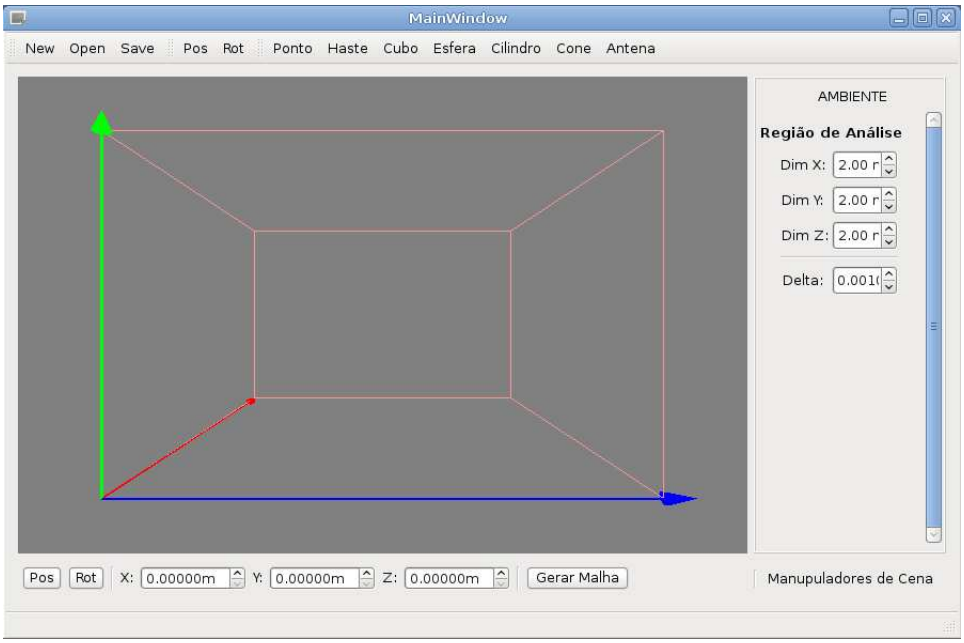


Figura 3.1: Layout interface.

uma cena anterior caso tenha já exista alguma salva. Depois dele temos o botão *Save*, que salva a cena atual em uma arquivo chamado *Map.in*, que contem todas as características dos objetos(dimensões, tipo e parâmetros físicos) assim como suas posições.

Figura 3.2: Barra Superior interface.

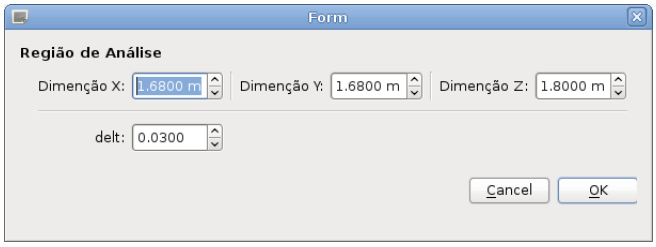


Figura 3.3: Janela com as característica da região de análise.

Mas em frente, encontram-se os botões *Pos* e *Rot*, os quais quando pressionados mostram o posicionamento e ângulos de rotação dos objetos selecionados respectivamente. Logo em seguida temos os criadores de objetos básico da interface, que são *Ponto*,*Haste*, *Cubo*, *Esfera*,

Cilindro, *Cone* e por fim *Antena*. Quando ativados criam o objeto desejado com os parâmetros especificados, assim como sua posição.

A janela da cena, Figura 3.4 é onde visualizamos nosso universo virtual. O painel lateral, Figura 3.5 é onde encontramos as características como dimensão e caraterísticas físicas do objeto selecionado ou criado.

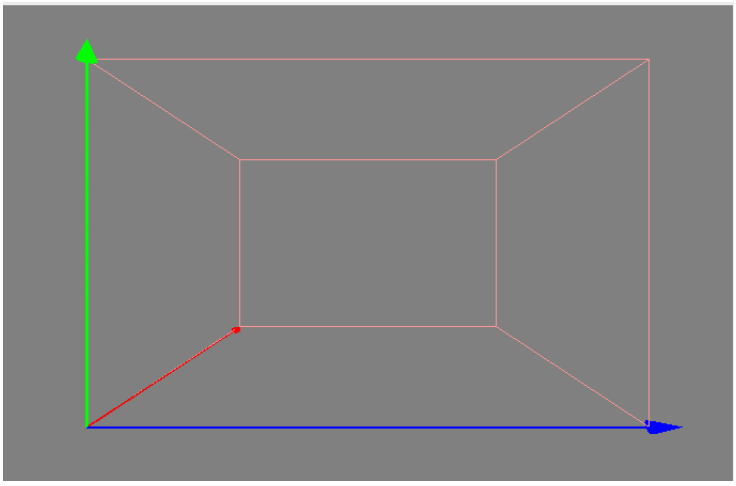


Figura 3.4: Janela da Cena.

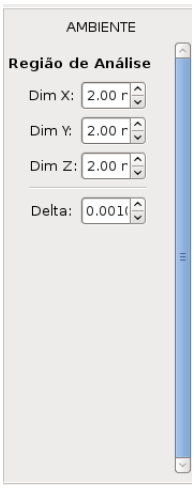


Figura 3.5: Painel lateral dos parâmetros da região de análise.

Na parte inferior da interface(Figura 3.6) encontram-se os botões *Pos* e *Rot*, que têm as mesmas funcionalidades dos já sitados da barra superior. Assim como a coordenada do objeto

selecionado(X, Y e Z). E por fim esta o botão *gerar Malha*, o qual tem a função de gerar a malha compatível com o simulador LANE-SAGS.

Figura 3.6: Barra inferior da interface.

3.2 Funcionalidades

3.2.1 Eventos de Mouse e Teclado

Tanto o irrlicht quanto o a biblioteca qt têm suporte à eventos de mouse e teclado. Todavia, o primeiro e realizado de forma distinta nesses dois ambientes. Então apareceu a necessidade de criação de um tipo de ”comunicador”que ”traduzisse”esse evento entres esses mundos diferentes.

Na interface esse ”comunicador”pegar as informações de mouse da janela do qt e envia de forma ”traduzida”para o motor gráfico irrlicht. São basicamente três métodos do qt que foram modificados para realizar esse tarefa: mouseMoveEvent(), mousePressEvent() e mouseReleaseEvent().

Já o evento de teclado não foi necessário o uso do ”comunicador”, pelo fato de não haver necessidade usarmos esse tipo de evento no irrlicht. Foram criados vários atalhos de teclado para facilitar a manipulação e navegação na cena, listados na tabela 3.1.

3.2.2 Colisão

O evento de colisão foi feito baseado na teoria de jogos. Existem vários tipos de colisão, porém a utilizada nesse trabalho foi por reta. Seu funcionamento é bem simples, cria-se uma reta imaginária na cena onde seu ponto de origem é a câmera e seu fim esta localizado em um ponto bastante distante ao inicial. O ponto final acompanha o movimento do mouse. Então quando clica-se em um dado ponto do universo criado na interface é disparado o evento de colisão o qual retorna *nó*[5] selecionado.

[5]Objeto selecionado

Atalho	Funcionalidade
Shift+O	Afastamento
Shift+P	Aproximação
W	Focaliza objeto selecionado
C	Clona objeto selecionado(duplica)
R	Remove objeto selecionado
1	Muda para câmera frente
2	Muda para câmera lateral esquerda
3	Muda para câmera lateral direita
4	Muda para câmera traseira
5	Muda para câmera topo
6	Muda para câmera base
M + X	Permite movimentação de objeto selecionado no eixo X com o mouse
M + Y	Permite movimentação de objeto selecionado no eixo Y com o mouse
M + Z	Permite movimentação de objeto selecionado no eixo Z com o mouse
Shift + A	Permite movimentação de objeto selecionado nos eixos X e Y com o mouse
Shift + B	Permite movimentação de objeto selecionado nos eixos X e Z com o mouse
Shift + C	Permite movimentação de objeto selecionado nos eixos Y e Z com o mouse
Shift + D	Permite movimentação de objeto selecionado nos eixos X, Y e Z com o mouse

Tabela 3.1: Tabela de Atalhos de teclado.

Mas isso só vale quando o há um objeto na coordenada selecionada sendo ele selecionável. No caso de se clicar em um dos gizmos[6] da região de análise, por exemplo, o evento de colisão retornará um objeto vazio, apesar de o gizmo ser um objeto normal ele foi configurado como não selecionável assim fazendo com que mesmo que a reta colida com ele o resultado será nulo. A Figura 3.7 mostra a ocorrência desse evento na interface, como pode-se observar o objeto selecionado é pintado na sua forma armada ou wireframe para diferenciar dos não selecionados.

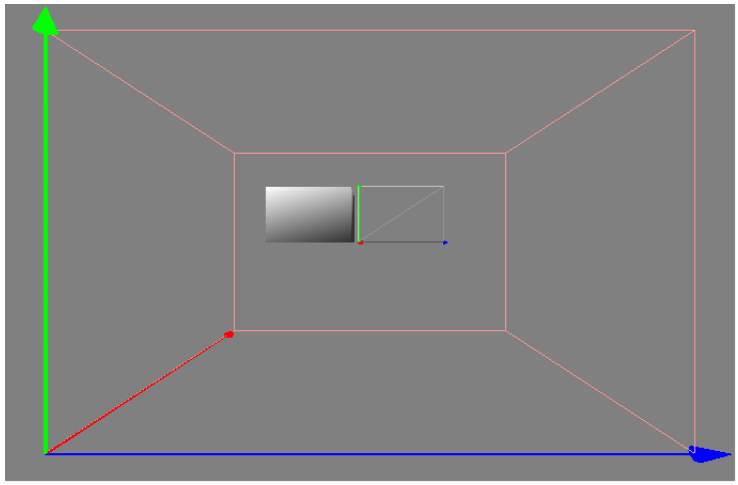


Figura 3.7: Ilustração do evento de colisão com o objeto cubo.

3.2.3 Visualizações

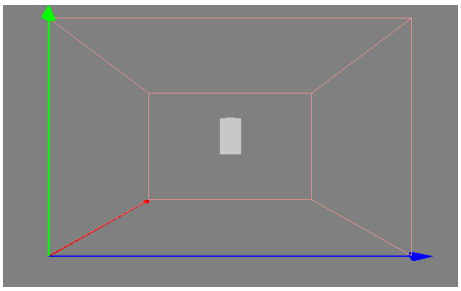
Ter vários ângulos de visualização é fundamental quando esta se modelando um ambiente 3D, principalmente quando vamos usá-lo para simulação de ondas eletromagnéticas pelo fato de qualquer erro deixado na estrutura poderá alterar no resultado final. Para sanar esse problema, o software confeccionado nesse projeto, permitir visualizar a região de análise por seis câmeras diferentes: frente(padrão), lateral esquerda, lateral direita, por trás, topo e base.

3.2.4 Aproximação e Afastamento

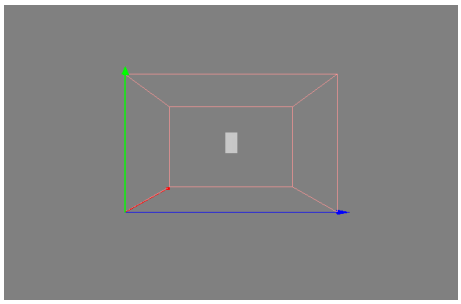
É uma característica importante da interface criada nesse projeto. Ela permite se aproximar(através do atalho de teclado $Shift + P$) e afastar(por meio de $Shift + O$) de uma objeto

[6]Orientadores de eixo, gizmo verde(representa o eixo y), gizmo azul(representa o eixo x) e gizmo vermelho(representa o eixo z)

que se encontra dentro da região de análise. Assim permitindo navegar pelo cenário virtual criado, podendo alterar, remover e duplicar objetos conforme necessário. Uma outra especificação dessa propriedade esta no fato de quando se tem um objeto selecionado. Se pressionarmos a tecla W a câmera irá focalizar nesse objeto. Para melhor entendimento a Figura 3.8(a) mostra a visualização de um objeto sem alteração de foco da câmera e distancia, já a Figura 3.8(b) mostra a visualização após um determinado afastamento e por fim as Figuras 3.9(a) e 3.9(b) mostram a aproximação normal e a por seleção.

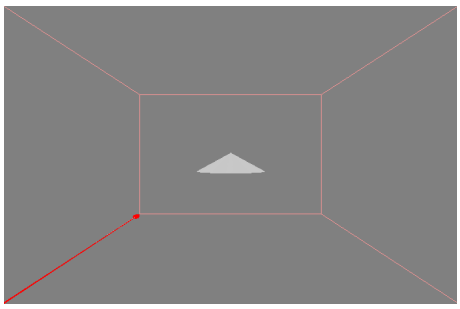


(a) Visualização de objeto cilindro sem aproximação, afastamento ou mudança de foco da câmera.

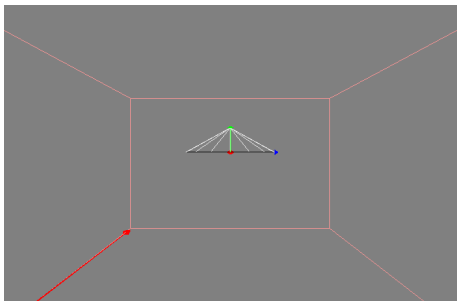


(b) Visualização de objeto cilindro com afastamento.

Figura 3.8: Visualização por afastamento.



(a) [Visualização de objeto por aproximação normal(sem mudança de foco).



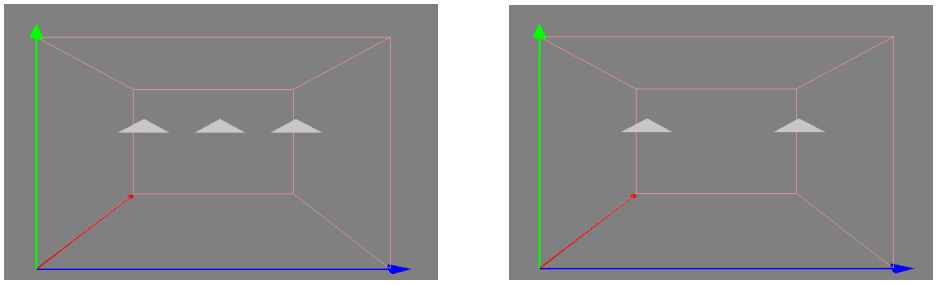
(b) Focalização de objeto selecionado através da tecla W.

Figura 3.9: Visualização por aproximação.

3.2.5 Remoção e Duplicação de Objetos

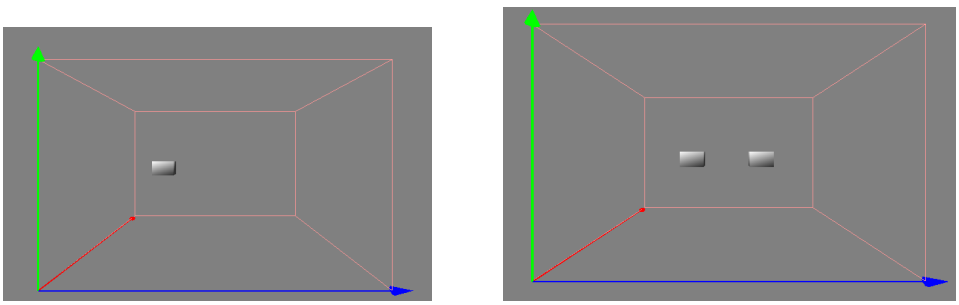
Nesse trabalho houve a necessidade criação de um mapa de objetos, pelo fato do irrlicht não permitir acesso direto ao seu array de objetos criados na cena. Dessa forma, qualquer alteração, remoção ou clonagem feita em um objeto na cena terá correspondência no mapa de *nós* gerenciado pela interface. Com isso, quando removemos um objeto criado na região de análise, ele também será removido do mapa. As Figuras 3.10(a) e 3.10(b) mostram a antes e depois de removermos um *nó* do nosso universo virtual.

O mesmo ocorre quando deseja-se duplicar um objeto, uma vez que isso ocorra o mapa receberá mais um *nó* com as mesmas característica do selecionado. Esse evento esta representado pelas Figuras 3.11(a) e 3.11(b).



(a) Antes da remoção. (b) Depois da remoção.

Figura 3.10: Operação de remoção.



(a) Antes da duplicação do objeto cone. (b) Depois da duplicação do objeto cone.

Figura 3.11: Operação de duplicação.

3.2.6 Conexão com o LANE-SAGS

Depois de implementar todas as funcionalidades do software, passamos para o último requisito proposto para este projeto que foi a conexão com o simulado LANE-SAGS. Isso foi realizado por meio de arquivos de saída (da interface) e entrada (no software LANE-SAGS).

Como a representação dos objetos e diferente no simulador, cada objeto da interface foi discretizado em um conjunto de prismas retangulares (com o menor prisma possível sendo a célula de Yee). Com cada prisma possuindo as características físicas ϵ (Permissividade Elétrica), σ (Condutividade Elétrica) e μ (Permeabilidade Magnética) do objeto.

A sequência de prismas representativos é gravada em um arquivo de saída chamado *bd.in*[7], compatível com o simulador LANE-SAGS. Cada prisma da sequência tem suas coordenadas e características físicas armazenados da seguinte forma: célula inicial X, célula final X, célula inicial Y, célula final Y, célula inicial Z, célula final Z, ϵ, σ e μ; Além deste arquivo, contendo a sequência de prismas representativo dos objetos, é gerado um segundo arquivo chamado *simconf.in*[8] contendo as informações referentes às: dimensões das células de Yee (em metros) e a dimensão em células do Espaço Delimitador (região de análise).

Com a conexão realizada, basta inserir alguns parâmetros, como: antena(s), planos de visualização, fonte e corrente; para então simular o ambiente. O diagrama mostrado na Figura 3.12, ilustra o processo de execução de uma simulação utilizando os dois softwares.

[7] bd - bloco dielétrico.

[8] simconf- configuração da simulação.

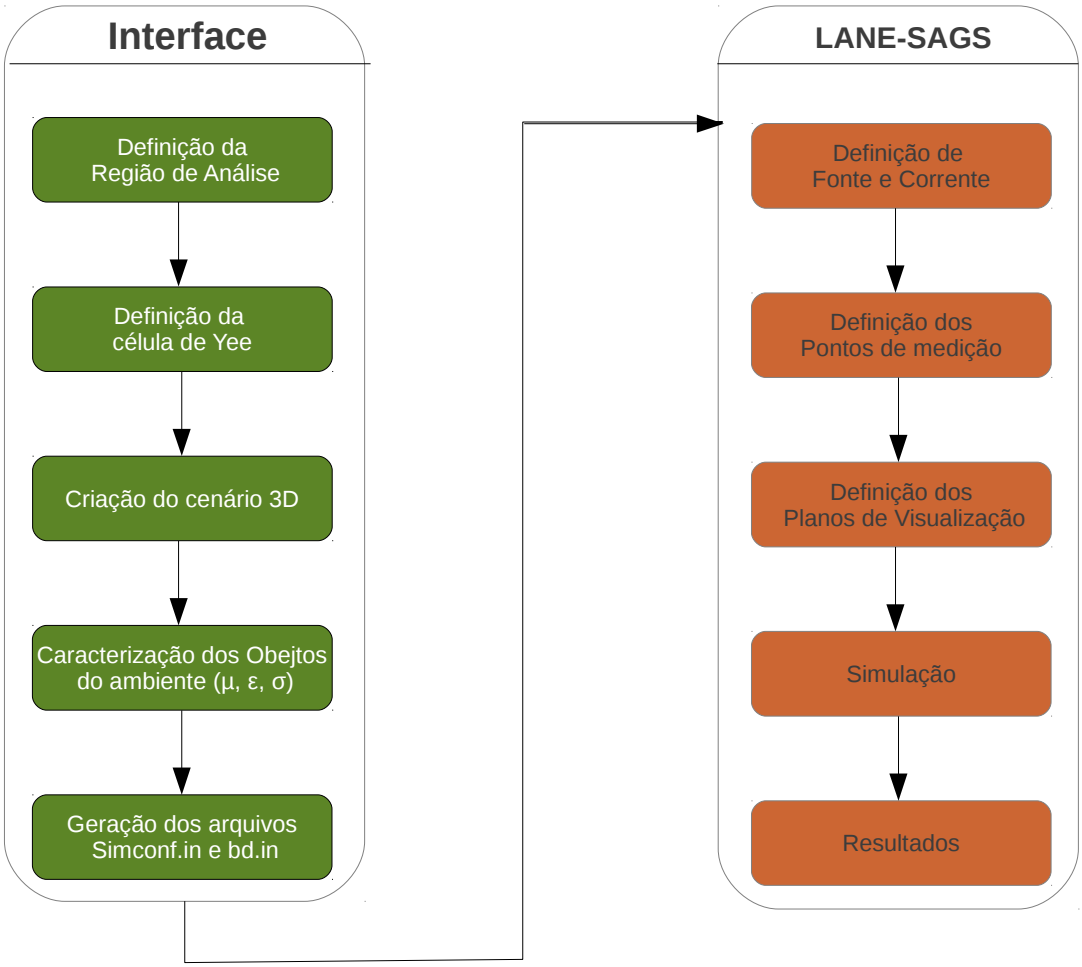


Figura 3.12: Diagrama de conexão com o LANE-SAGS.

Capítulo 4

Aplicação e Resultados

Com a finalidade de validar a interface, foi construído um AV apartir da caracteristicas de um ambiente real(escritório), com planta baixa ilustrada na Figura 4.1. Também foram modelados os objetos presentes nessa cenário real, como : computadores, monitores, bancadas, divisórias de vidro e porta.

Esperimentalmente, posde-se observar o comportamento da propragação do sinal de uma antena nesse escritório.

Alguns desse objetos estão dispostos em alturas diferentes em relação ao piso. As bancadas e o teto estão a uma altura de 0,75 metros e 2,73 metros, respectivamente.

Como resultado da modelagem usando a interface, obeteve-se a estrutura mostrada na Figura(colocar figura estrutura da interface). A disposição dos seus objetos seguiu a posicionamento real dos mesmos no momento das medições.

As especificações desse cenário, foram:

- *Delta*(Tamanho da célula de Yee) utilizada foi de $0,003m$.
- *Região de Análise* com dimensões: X = $15m$, Y = $3m$ e Z = $15m$.
- *A* lista das carateristicas físicas com seus respectivos matérias estam presentes na Tabela 4.1

Depois de ambiente ter sido montado, surguiu a necessidade de se introduzir a antena . Como a que foi usada na medição não tinha uma representação virtual, foi necessário criá-la. Com isso foram feitos alguns teste usando antenas dipolo, com o intuito de obeter a máxima

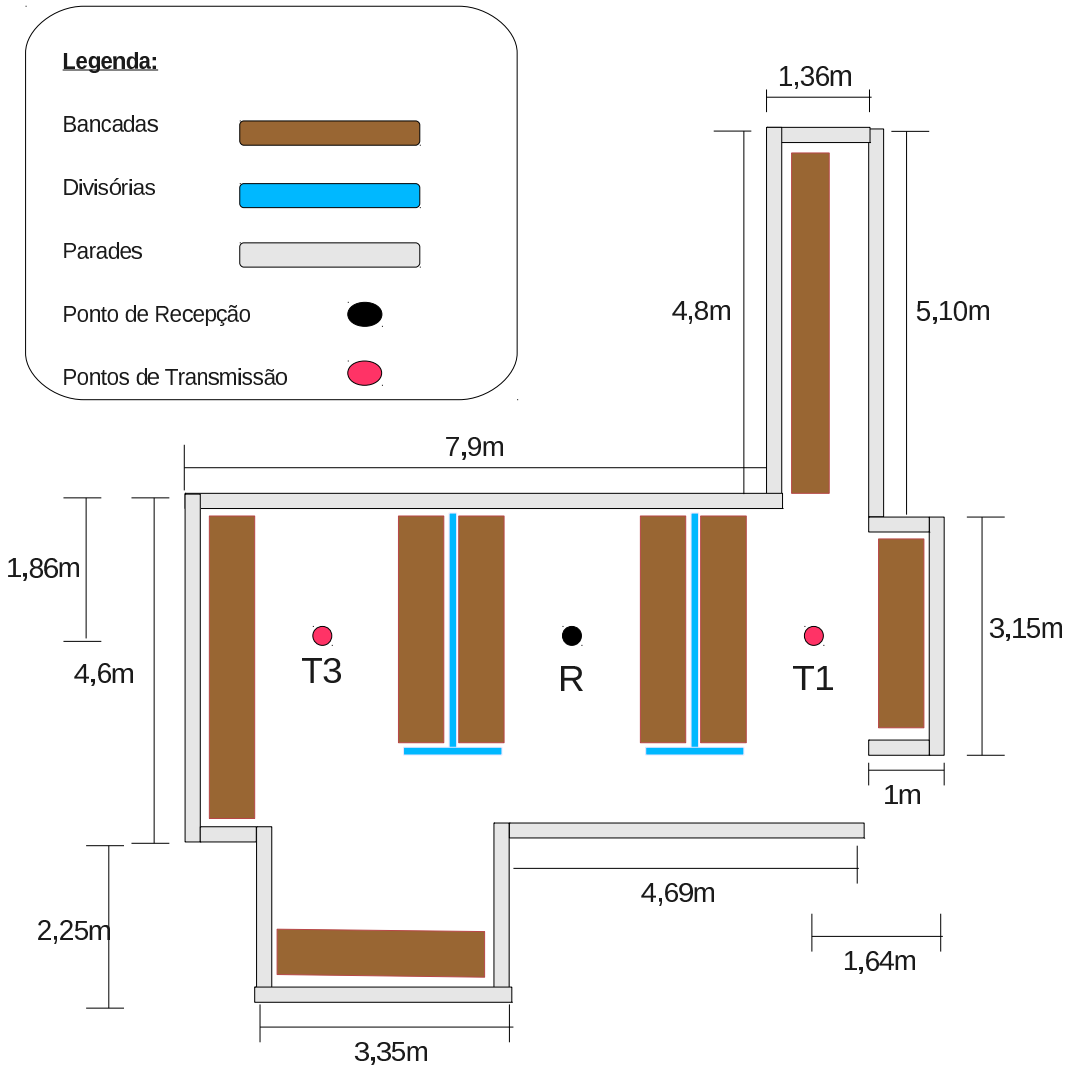


Figura 4.1: Planta baixa do escritório modelado usando a interface.

aproximação do comportamento da antena real.

Para todos os modelos de antena foi ultizada uma fonte de tensão de trêm de pulsos normalizado, Figura **??**, onde através da transforma de Fourier, pode-se observar se a mesma estava trabalhando na banda adequada. O resultado obtido foi o ilustrado na Figura **??**.

A primeira antena feita foi um dipolo mostrado na figura **??**

Visto que o sua perda de retorno não foi satisfatória, foram feitas modificações em sua dimensão e por fim foi adicionado um capacitor(Figura(colocar figura antena capacitor)). Com essas mudanças se obteve um antena com um comportamento parecido com o da real. Os

Elementos dos Ambiente	Material	ϵ	σ	μ
Parede	Tijolo	4	0,0135	1
Teto	Gesso	2,8	0,1533	1
Bancadas	Madeira	1,8	0,011	1
Monitores e Gabinetes	Metal	5	$valor$	1
Divisórias	Vidro	5	$valor$	1
Piso	Concreto	5	1	1

Tabela 4.1: Tabela de elemetos e seus materiais.

gráficos de perda dessas duas construções esta ilustrado na Figura(colocar figura da perda)

Capítulo 5

Considerações Finais

Referências Bibliográficas

[1] T. Back, “Evolutionary computation: comments on the history and current state,” pp. 3–17, 1997, iEEE Computational Intelligence Society.

[2] A. T. e Susan Hangness, “Computational electrodynamics: The finite-difference time-domain method,” 2005.

[3] R. M. e Silva de Oliveira, “Método fdtd aplicado na análise da propagação eletromagnética em ambientes indoor e outdoor,” Universidade Federal do Pará, Instituto de Tecnologia, 2002, trabalho de Conclusão de Curso.

[4] PUC-RS, “Origens da computação gráfica,” acessado em 06/2012. [Online]. Available: http://www.inf.pucrs.br/~pinho/CG/Aulas/Intro/intro.htm

[5] E. A. e A. Conc, *Computação Gráfica: Teoria e Prática*. Campus/Elsevier, 2003.

[6] C. K. e R. Tori, “Realidade virtual: Conceitos e tendências,” pp. 3–20, 2003, são Paulo: VII Symposium on Virtual Reality.

[7] I. Sutherland, “Sketchpad: The first interactive computer graphics,” Ph.D. dissertation, MIT - Massachusetts Institute of Technology, 1963.

[8] “Origens da computação gráfica,” acessado em 06/2012. [Online]. Available: http://hem.passagen.se/des/hocg/hocg_1960.htm

[9] A. Kamar, “Computer graphics - transformations, clipping and rasterization,” acessado em 06/2012. [Online]. Available: http://azharkamar.hubpages.com/hub/Computer-Graphics-Transformations-Clipping-Rasterization

[10] H. J. Speck, “Avaliação comparativa das metodologias utilizadas em programas de modelagem sólida,” 2001, dissertação de Mestrado, Universidade Federal de Santa Catarina/PPEP.

[11] A. T. e M. Oliveira, “Representação de objetos tridimensionais: Modelos poligonais,” 1995, departamento de Ciências de Computação e Estatística, Instituto de Ciências Matemáticas de São Carlos.

[12] D. Hearn and M. P. Baker, *Computer Graphics with OpenGL*. Pearson Prentice Hall, 2004.

[13] A. Gahan, *3ds Max Modeling for Games, Volume 1*. Focal Press, Inc, 2011.

[14] J. Lanier, “An insider’s view of the future of virtual reality,” 1992.

[15] H. Rheingold, “Virtual reality,” 1992.

[16] USP, “Caverna digital.” [Online]. Available: http://www.lsi.usp.br/interativos/nrv/caverna.html

[17] E. Communications and Technology, “Different kinds of virtual reality,” 2001.

[18] A. Ferreira, “Uma arquitetura para a visualização distribuída de ambientes virtuais,” 1999, dissertação de Mestrado, PUC/RJ.

[19] A. S. Kyan, *Irrlicht 1.7 Realtime 3D Engine*. Packt Publishing Ltd, 2011.

[20] N. Gebhardt, “Engine irrlicht,” 2003. [Online]. Available: http://irrlicht.sourceforge.net

[21] K. Yee, *Numerical solution of initial boundary value problems involving Maxwell’s equations in isotropic media*, 1966.

[22] R. M. e Silva de Oliveira, “Nova metodologia para análise e síntese de sistemas de aterramento complexos utilizando o método ln-fdtd, computação paralela automática e redes neurais artificiais,” Ph.D. dissertation, Instituto de Tecnologia da Universidade Federal do Pará, 2008.

[23] A. Taflove, *Numerical solution of steady-state electromagnetic scattering problems using the time-dependent Maxwell’s equations*, 1975.

[24] J. F. Almeida, “Análise fotônica em estrutura de microfita planar usando o método fdtd com processamento paralelo,” Ph.D. dissertation, Instituto de Tecnologia da Universidade Federal do Pará, 2004.

[25] Nokia, “Plataforma qt, signal and slot documentation.” [Online]. Available: http: //qt-project.org/doc/qt-4.8/signalsandslots.html