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

Criação de Ambientes *indoor* através de Realidade Virtual para 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

Criação de Ambientes *indoor* através de Realidade Virtual para 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

Criação de Ambientes *indoor* através de Realidade Virtual para 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 final pela banca examinadora que atribuiu o conceito _____.

_____

Prof. Dr. Rodrigo Melo e Silva de Oliveira

ORIENTADOR

(UFPA)

_____

Washington César Braga de Sousa

COORIENTADOR

(UFPA)

_____

Prof. Dr. Fabrício José Brito Barros

MEMBRO DA BANCA EXAMINADORA

(UFPA/Campus de Tucuruí)

_____

Prof. Dr. Josivaldo de Souza Araújo

MEMBRO DA BANCA EXAMINADORA

(UFPA)

_____

Prof. Dr. Ádamo Lima de Santana

DIRETOR DA FACULDADE DE ENGENHARIA DA COMPUTAÇÃO

Dedico este TCC à minha mãe, pai, irmãos, a Laryssa Rosendo e ao meus amigos, pelo apoio e incentivo, sem os quais este trabalho não seria possível.

Agradecimentos

Primeiramente agradeço a oportunidade dada pelo meu orientador, Rodrigo Melo, e co-orientador, Washington, sem os quais não seria possível a realização desse projeto. Sou grato também pela forma com que eles gerenciaram esse trabalho.

Agradeço também a minha mãe, por toda a sua dedicação e sacrifícios para me proporcionar o máximo de conforto em outra cidade. Ao meu pai, pela torcida fiel que sempre me acompanhou e respeitou as minhas decisões. A minha irmã, por todas as tarefas domésticas realizadas e o meu irmão, pelo seu senso crítico e visão de mercado, que me da sempre conselhos visando ajudar no meu futuro profissional.

Queria agradecer também a minha namorada e companheira, Laryssa Rosendo, que sempre me apoiou mesmo nos momentos mais difíceis durante o curso, por acreditar quando até eu mesmo duvidada da minha capacidade e pelo amor dedicado a mim. Agradeço também a família dela, que me acolheu e me tratou e trata como um membro deles.

Por fim agradeço a todos os meus amigos e colegas de faculdade pelo companheirismo durante todo o percusso para a obtenção da graduação.

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

MIT Massachusetts Institute of Technology

Resumo

Neste trabalho, foi realizado o estudo de propagação de ondas eletromagnéticas em um ambiente *indoor*. Este estudo foi efetuado através do desenvolvimento de um *software* que modela cenários 3D, por meio do qual gera-se uma malha computacional compatível com o simulador numérico baseado no método FDTD (LANE-SAGS). Utilizou-se desta ferramenta para modelar parte de um prédio real, localizado no CETUC (PUC-Rio), além das antenas dipolo. Dessa forma o modelo computacional (baseado em realidade virtual), aqui desenvolvido, representa graficamente os parâmetros dos diferentes meio que compõem o modelo numérico baseado no método FDTD. Foi calculada a resposta impulsiva do canal de rádio e perfil de potência e retardo. Resultados compatíveis com os publicados na literatura, para esta prédio, foram obtidos.

Palavras-chave: Computação Gráfica, Modelagem de Ambientes, Ambientes *indoor*, FDTD, Equações de Maxwell e Perfil de Potência e Retardo do Canal.

Capítulo 1

Introdução

Com o passar dos anos, a computação tem evoluído de forma exponencial [1], permitindo cada vez mais ao ser humano: armazenar grande quantidades de dados, simular ambientes de grandes magnitudes, fazer previsão de eventos, se comunicar audiovisualmente, etc. Todo esse avanço vem refletindo de forma crescente na vida do homem.

Mais 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 antigos) nos estudos relativos à propagações de ondas eletromagnéticas em ambientes *indoor* e *outdoor* [3].

Porém, muitas vezes, a representação virtual de desses ambientes não é uma tarefa fácil. Dessa forma, surge a necessidade de criação de *softwares* que auxiliem nessa construção. Eles podem ser de modelagem 2D ou 3D desde de que permitam a criação de estruturas básicas, como: triângulos, círculos, cubos, esferas e pirâmides; como também outras bem mais complexas(fractais, estruturas periódicas e prédios), gerando sempre a base que contenha as coordenadas de cada objeto desse cenário.

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, um cenário real. A partir desse universo virtual, objetiva-se obter a malha compatível com o programa LANE-SAGS. Através dele, simular a propagação de ondas ele-

tromagnéticas utilizando o método FDTD. Assim possibilitando analisar tanto aterramento e descargas elétricas quanto a resposta desse cenário a propagação do sinal de uma antena.

Dessa forma, os principais objetivos desse trabalho são:

1. Representar estruturas tridimensionais por meio de alguns objetos básicos, como: cubo, esfera, cilindro e cone.

2. Criar AV com base em ambientes reais.

3. Gerar uma base de dados com as característica de cada estrutura, sua posição, dimensão e parâmetros físicos(μ, σ e ϵ).

4. Gerar malha compatível com o simulador LANE-SAGS, efetuar a simulação e comparar os resultados com experimentos reais relativos à propagação eletromagnética em um ambiente *indoor*.

1.2 Organização do Trabalho

Este trabalho foi estruturado da seguinte forma:

- **Capítulo 2:** Trata da base teórica necessária para realização do trabalho, falando de computação gráfica, técnicas de modelagem de sólidos, teoria dos grafos de cena e o método FDTD.

- **Capítulo 3:** Fala sobre 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, atalhos de teclado, eventos de mouse, etc.

- **Capítulo 4:** Mostra uma aplicação usando o *software* com seus resultados (comparados com os obtidos via medição), seguindo todos os passos desde da criação de um ambiente até sua simulação no LANE-SAGS.

Capítulo 2

Fundamentação Teórica

2.1 Computação Gráfica e Modelagem 3D

Computação gráfica é a representação e manipulação de dados utilizando um computador, com ajuda de *software* e *hardware* especializados. O seu desenvolvimento facilitou a iteração humano-computador, melhorando o entendimento e a interpretação de uma variedade de tipos de dados [4].

Os desenvolvimentos na área de computação gráfica têm causado impactos em vários tipos de mídia, assim revolucionando as industrias de: animação, cinema e jogos. Ela também tem afetado outros seguimentos da vida do ser humano, tais como os exemplos descritos na Tabela 2.1.

Um dos importantes conceitos da computação gráfica é o da modelagem 3D, que é o processo matemático para representação de objetos tridimensionais via software. Estes modelos podem ser feitos manualmente ou automaticamente; o processo manual tem semelhança com o método artístico de criação de esculturas, visto que o modelador constrói um objeto utilizando somente as ferramentais básicas disponíveis(tendo a criatividade como a ferramente principal). Já no processo automático o modelador utiliza-se de super *scanners*, que capturam as característica de um objeto ou ambiente 3D deixando somente alguns ajustes por parte do modelador.

2.1.1 Transformações

As transformações geométrica são ferramentas importantes em computação gráfica, pois permitem realizar ações do mundo real em um mundo virtual criado computacionalmente. Elas

Tabela 2.1: Diferentes ramos da computação gráfica.

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

representam o mapeamento de um ou mais pontos de acordo com suas mudanças de posição [5]. As transformações mais usadas e conhecidas são: rotação, escala e translação.

2.1.1.1 Translação

A transformação geométrica mais básica é a de translação. É definida como a movimentação de todos os ponto, pertencente a um objeto, por uma mesma distância em uma mesma direção. Esta operação é dada pela equação (2.1), mostrando que quando certas quantidades (T_x, T_y e T_z) são adicionadas às coordenadas (x, y e z) (de todos os ponto de um dado objeto), obtêm-se uma nova posição transladada (x', y' e z') para o ponto (x, y, z). A Figura 2.1 ilustra essa transformação.

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

2.1.1.2 Escala

A operação de escala está associada à mudança de dimensão (tamanho) dos objetos. Ela é representada pela equação matricial (2.2), a qual expressa que a multiplicação das dimensões atuais (x, y e z) pelos fatores de escala (S_x, S_y e S_z) tem como resultado um novo objeto escalado (x', y' e z'). A Figura 2.2 mostra o que ocorre quando um objeto é escalonado.

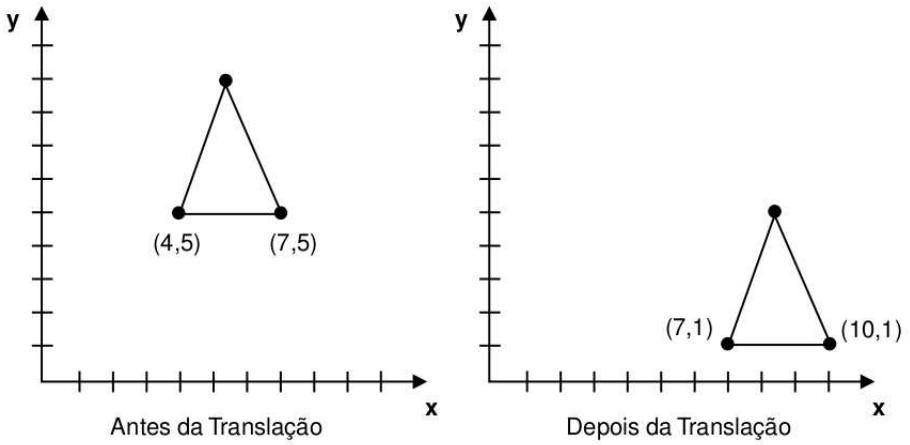


Figura 2.1: Operação de Translação 2D.

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

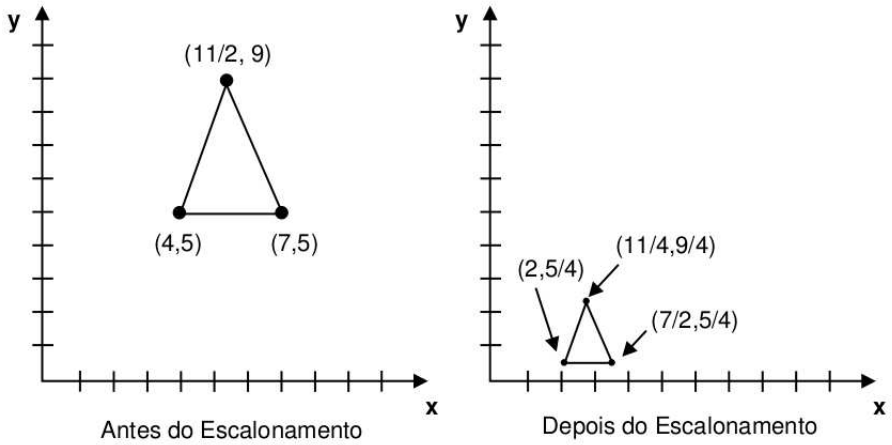


Figura 2.2: Operação de Escala 2D.

2.1.1.3 Rotação

Rotação é a transformação que realiza o movimento giratório de um dado objeto entorno de um ponto fixo conhecido como centro de rotação. Esta transformação pode ser representada pelas formas matriciais (2.3), que mostram, respectivamente, as matrizes de rotação referentes

aos eixos x, y e z. A Figura 2.3 ilustra esta operação.

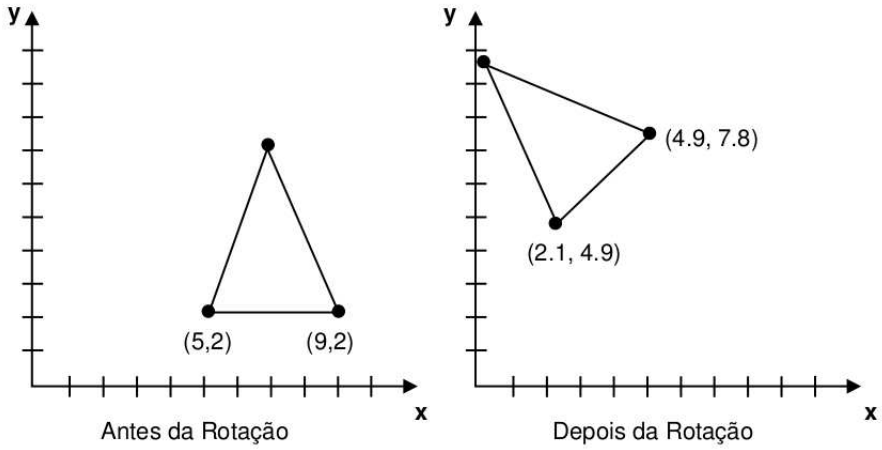


Figura 2.3: Operação de Rotação 2D.

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

2.1.2 Representação de Sólidos

A representação de sólidos é um tópico de fundamental importância no aprendizado de modelagem 3D. Pois, dependendo da representação escolhida, pode-se tanto manipular quanto visualizar determinadas características de um objeto (sólido).

A teoria de representação de sólidos é bastante ampla e possui um número muito grande de técnicas de representação. As técnicas utilizadas no desenvolvimento deste projeto são: Representação Aramada, Representação por Face, Representação por Polígono e Representação por Enumeração da Ocupação Espacial.

2.1.2.1 Representação Armada ou *Wireframe*

Nessa representação, visualiza-se somente as arestas e vértices que formam o objeto (Figura 2.4) [6]. Sendo esta, a técnica mais simples de representação de objetos 3D.

Suas vantagens e desvantagens estão associadas à visualização simplificada que esta proporcionada. Como vantagem, destaca-se a questão do custo computacional, visto que não há a necessidade de pintura das faces dos objetos (já que elas não existem neste modelo). A desvantagem é referente a ambiguidade na visualização dos objetos, como pode ser visto no exemplo apresentado na Figura 2.5.

Neste trabalho, a representação armada foi utilizada no evento de colisão, seção 3.5.2, para diferenciar a visualização de um objeto quando está selecionado.

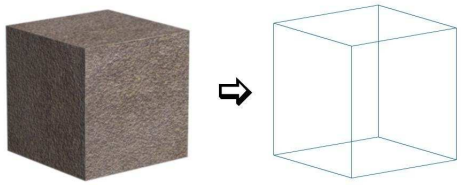


Figura 2.4: Representação Armada ou *Wireframe*.

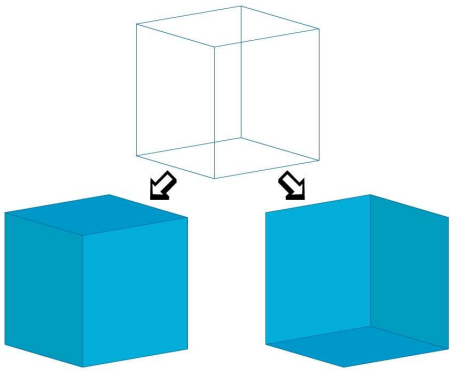


Figura 2.5: Ambiguidade de representação de sólidos.

2.1.2.2 Representação por Faces

Também conhecida com *Boundary Representation* ou simplesmente *B-rep*, essa técnica representa um modelo sólido por meio de suas fronteiras espaciais (Fig. 2.6), tendo em geral uma superfície externa e uma convenção que define qual lado da superfície está o material sólido. Desta forma, um objeto sólido pode ser representado por faces, delimitadas por arestas, definidas por vértices [7].

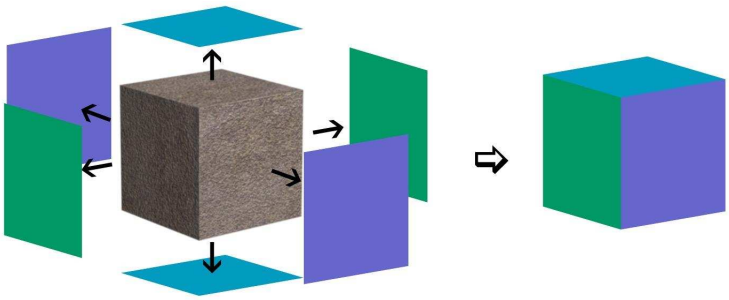


Figura 2.6: Representação por Faces.

A primeira geração desta técnica, permitia representar sólidos somente por faces planas, o que limitava o número de formas geométricas que poderiam ser modeladas. Já a segunda geração, incluiu objetos primitivos com superfícies analíticas, possibilitando, assim, a criação de modelos um pouco mais complexos como cilindros, esferas, cones, etc.

Foram realizados outros estudos, como intuito de que a técnica ***B-Rep*** conseguisse modelar um maior número de formas geométricas, resultando em uma maior utilização deste método na modelagem de sólidos [7].

Sua relevância nesse projeto, está associada ao entendimento da diferença entre o universo físico e o representativo (modelado), visto que, quando utiliza-se essa técnica, as informações referentes ao material sólido do objeto modelado não tem relevância.

2.1.2.3 Representação Poligonal

Polígono vem do grego *polys*, que significa muitos e *gonos* que significa ângulos. Assim, esta representação é formada por figuras planas com muitos segmentos de reta e muitos ângulos. Então, pode-se construir sólidos com o polígono mais simples, o triângulo, ou mesmo com um mais complexo, com um número elevado de lados(Fig. 2.7). Essa técnica pode ser considerada uma especificação do caso apresentado na seção 2.1.2.2.

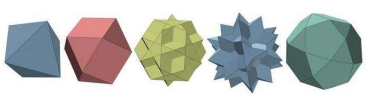


Figura 2.7: Representação Poligonal.

Tessellation 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 utiliza a representação por faces triangulares. Isso se deve ao fato de esta necessitar de menos processamento e também por possibilitar a representação de grande parte dos tipos de contorno [8].

A *engine Irrlicht* se utiliza dessa técnica de representação para criar as malhas de seus objetos. A Figura 2.8, ilustra o método de criação da malha cônica utilizando a técnica de representação poligonal.

Parâmetros:

radius	Raio do cone.
length	Comprimento do cone.
tessellation	Número de quadriláteros em torno da circunferência do cone.
colorTop	A cor do topo do cone.
colorBottom	A cor da base do cone.
oblique	Inclinação do cone.

Figura 2.8: Método de criação da malha de um cone que usa a técnica de *tessellation* [9].

2.1.2.4 Representação por Enumeração da Ocupação Espacial

Esta técnica consiste na decomposição do espaço ocupado por um objeto em uma grade regular, formada por cubos(também chamados de *voxels*) de dimensões iguais [10] , como está ilustrado pela imagem 2.9.

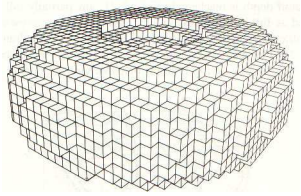


Figura 2.9: Exemplo de objeto representado por enumeração da ocupação espacial

Quando um objeto é representado usando enumeração por ocupação espacial, controla-se somente a ausência ou presença da célula (cubo) em cada posição da grade. Desta forma, o objeto pode ser armazenado em uma lista de cubos que pertencem ao espaço por ele ocupado.

Este modelo de representação permitiu a conexão do *software* desenvolvido neste projeto com o simulador que utiliza o método FDTD (LANE-SAGS) . Esta união será melhor abordada na seção 3.5.5.

2.1.3 Realidade Virtual

O termo Realidade Virtual surgiu na década de 1980 pelo artista e cientista da computação Jaron Lanier, que uniu dois universos, que até então eram muito distantes: o mundo real e o virtual. Porém, há registros de trabalhos anteriores a essa denominação. Um deles é o primeiro capacete que permitia imersão. Outro que não pode ser esquecido é o SENSORAMA, que foi construído na década de 1960 e tinha a capacidade de reproduzir imagens coloridas 3D, som, além de pode simular cheiros, ventania e vibrações durante os filmes.

Mas o que é Realidade Virtual? É uma interface avançada que permite ao usuário navegar, modificar, interagir em tempo real com uma determinada aplicação que contenha objetos virtuais [11]. Existem basicamente dois tipos de realidade virtual quanto o fator imersão: a imersiva e a não imersiva. A primeira é caracterizada por permitir ao usuário se sentir dentro do ambiente(através do uso de capacetes especiais, óculos, luvas, roupas, caves [12], sensores, dentre outros dispositivos). Já na segunda, isso não ocorre, utiliza-se mouse, teclado, monitores, etc [13].

2.1.3.1 Grafo de Cena

O grafo de cena é um conceito muito usado em computação gráfica, realidade virtual e teoria de jogos. Trata-se de uma estrutura de dados hierárquica que serve para representar as características e relações dos objetos que compõem um ambiente virtual tridimensional nas aplicações de computação gráfica.

Um ambiente virtual é uma representação de diversos aspectos do mundo abstrato ou virtual. Dentre estes aspectos estão a descrição geométrica, câmera, transformação, aparência, comportamento e iluminação, que são os mais relevantes em uma aplicação gráfica [14]. Sendo assim, a representação de um cenário virtual é dada pela inserção de todos estes aspectos dentro do grafo de cena. Esta estrutura é formada por nós que compõem arestas, criado um conjunto em forma de árvore (grafo acíclico direcionado), Figura 2.10. Em um grafo de cena cada nó tem atributos que podem, ou não, influenciar seus nós conectados (nós filhos). Existe uma hierarquia na organização dos nós, que corresponde de forma semântica e espacial com o universo

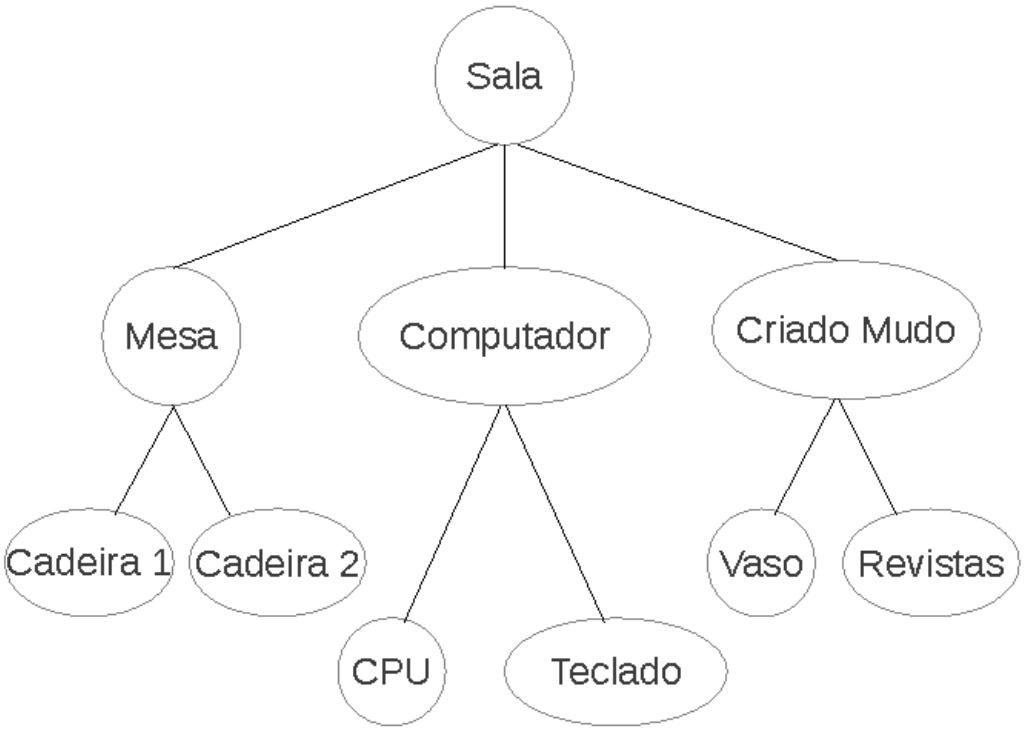


Figura 2.10: Grafo de cena de uma sala.

modelado.

Com base nas características hierárquicas principais, os nós podem ser classificados em três categorias: nó raiz(é o primeiro nó, no qual todos os outros estão ligados de forma direta ou não), nó interno (geralmente contém informações de transformações 3D - rotação, escala e translação) e, por fim, há o nó folha (que por padrão contém as dados de representação geométrica dos objetos da cena).

A propriedade fundamental dessa ferramenta é o que se chama de herança de estado. Ela determina que cada nó deve herdar as propriedades de todos os seus ancestrais provenientes do nó raiz, ou seja, se a posição do objeto principal for alterada, todos os objetos vinculados a ele irão modificar-se da mesma forma, mantendo suas posições relativas ao principal. Visualiza-se, então, que esta propriedade é bastante útil na organização de cenas 3D.

A maioria dos motores de jogos, incluindo o escolhido para este projeto, utilizam as teorias de realidade virtual e grafo de cena. A justificativa está nas vantagens proporcionadas pelo uso desses conceitos. As principais vantagens podem ser vistas na Tabela2.2.

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

Vantagens	
Produtividade	Gerencia e reduz o número de linhas de código 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 específica de uma plataforma.
Escalabilidade	Possibilita trabalhar tanto em máquinas com configurações básicas até supercomputadores

2.2 Motor Gráfico

Um motor de jogos, dentro do conceito de engenharia de *software*, é a parte do projeto que executa certas funcionalidades para um programa (*software*). No desenvolvimento de jogos, o motor se encarregará de interagir com o hardware gráfico, afim de dar o suporte necessário para visualização dos modelos, cuidará da entrada de dados do jogador, tratará todos os processos matemáticos envolvidos e gerenciar outros aspectos relacionados ao desenvolvimento de baixo nível, facilitando o desenvolvimento de um jogo ou sistema gráfico [15].

Existem inúmeras definições para uma *game engine*, porém todas elas convergem para as seguintes características principais:

- Possibilitar o reaproveitamento de código.
- Promover uma fácil integração entre código e modelagem 3D.
- Permitir o desenvolvimento de vários jogos, usando o mesmo motor. Todavia, é natural encontrar restrições em relação ao tipo de jogo que pode ser desenvolvido.

A Tabela 2.3 mostra os principais motores gráficos utilizados atualmente.

A *game engine* utilizada neste trabalho, foi o *Irrlicht*. Esse motor de jogos foi escolhido, pelo conhecimento prévio obtido por meio do uso desta ferramenta em outros trabalhos e, principalmente, por ser *open source* (código aberto).

Tabela 2.3: Principais motores de jogos.

Motores Tradicionais	3D GameStudio, BGE(Blender), C4 Engine, **Irrlicht**, NeoAxis Engine, Panda3D, Source, Torque Game Engine, Unreal Engine
Motores Proprietários	3D GameStudio, C4 Engine, DX Studio, EGO, Havok, NeoAxis Engine, Source, Unity, Truevision3D
Motores de Código Aberto	BGE (Blender), Crystal Space, **Irrlicht**, jME, ODE, OGRE, OpenSceneGraph, Panda3D, RPG Toolkit

2.3 Método FDTD

Kane Yee criou o método FDTD (Finite-Differene Time-Domain Method), primeiramente chamado como Algorítimo de Yee, em 1966. Ésta técnica possibilita solucionar numericamente as equações de Maxwell de maneira simples e direta [16]. 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.

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.

Porém, por algum tempo, esse método foi deixado de lado devido ao 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, algum 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 [17]. Assim, novas técnicas apareceram e melhoram a abordagem do método, acompanhado 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 [18].

O método FDTD aplicado com intuito de simular 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 equações escalares mostradas abaixo.

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

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

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

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

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

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

Onde:

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.

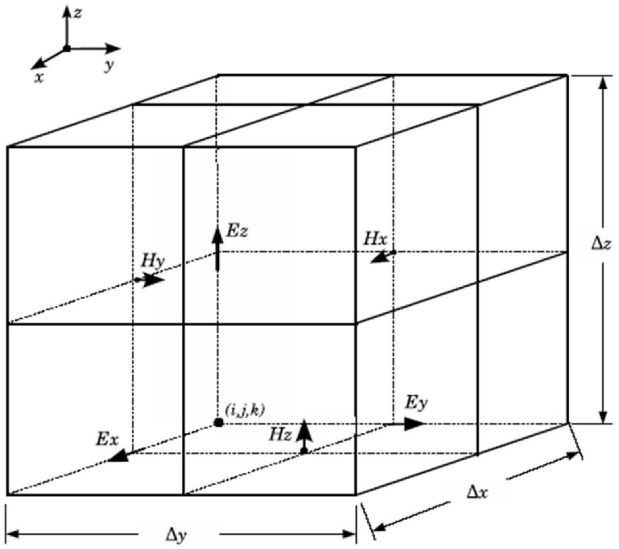


Figura 2.11: Célula Yee.

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, está mostrada na Figura 2.11 [16].

Para que possam ser realizadas simulações de propagação de onda através do método FDTD, é criada uma malha 3D formada por múltiplas células de Yee que preenchem toda região de análise. A Figura 2.12 [19], tem a finalidade de mostrar as componentes de campo de $\overline{E}$ e $\overline{H}$ distribuídas espacialmente na célula de Yee e a representação discreta de uma região do espaço através de múltiplas células.

Como modo de referenciar uma célula nesta malha, utiliza-se os índices i, j, k (discretos) para determinar a coordenada x, y, z (em metros), 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)..

2.3.1 Obtenção das Equações FDTD para $\overline{E}$ e $\overline{H}$

A análise das equações (2.6)-(2.11) permite notar que suas derivadas podem ser aproximadas por derivadas centradas, tanto em relação ao tempo, quanto em relação ao espaço. A equação (2.12), é obtida a partir do trucamento de segunda ordem da série de Taylor, definindo o conceito de derivada centrada unidimensional.

$$\frac{\partial f(x)}{\partial x} \approx \frac{f(x+\Delta_x) - f(x-\Delta_x)}{2\Delta_x} \tag{2.12}$$

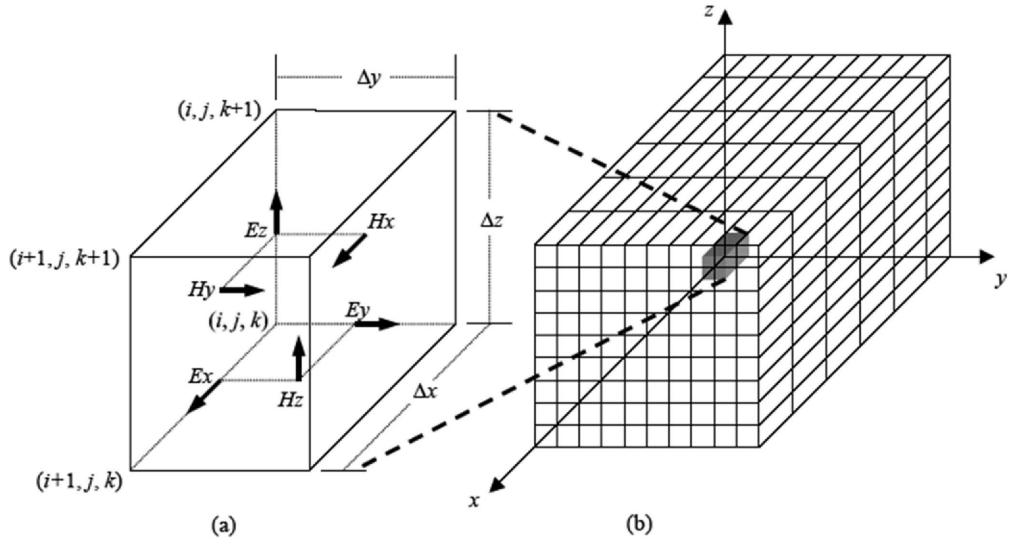


Figura 2.12: (a) Posição das componentes dos campos elétrico e magnético em uma célula de Yee;(b) Célula no interior de uma malha 3-D.

Observando a equação (2.12) nota-se que a derivada no ponto x em relação a x pode ser aproximada por meio dos valores da função $f(x)$ nas coordenadas $x + \Delta_x$ e $x - \Delta_x$. Esta aproximação melhora com a diminuição do valor do Δ_x. Então, uma função F que dependa do tempo t e das coordenadas espaciais (x, y, z) pode ser aproximada por uma função F_a discreta, como representado matematicamente pela eq.(2.13).

$$F(t, x, y, z) \approx F_a^n(i, j, k) \tag{2.13}$$

Utilizando a técnica de derivada centrada nas Equações (2.6)-(2.11), obtêm-se as equações discretas para $\overline{H}$ e $\overline{E}$:

$$Hx^{n+\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})} = Hx^{n-\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})} + \frac{\Delta t}{\mu}\left[\frac{Ey^n_{(i,j+\frac{1}{2},k+1)} - Ey^n_{(i,j+\frac{1}{2},k)}}{\Delta z} - \frac{Ez^n_{(i,j+1,k+\frac{1}{2})} - Ez^n_{(i,j,k+\frac{1}{2})}}{\Delta y}\right] \tag{2.14}$$

$$Hy^{n+\frac{1}{2}}_{(i+\frac{1}{2},j,k+\frac{1}{2})} = Hy^{n-\frac{1}{2}}_{(i+\frac{1}{2},j,k+\frac{1}{2})} + \frac{\Delta t}{\mu}\left[\frac{Ez^n_{(i+1,j,k+\frac{1}{2})} - Ez^n_{(i,j,k+\frac{1}{2})}}{\Delta x} - \frac{Ex^n_{(i+\frac{1}{2},j,k+1)} - Ex^n_{(i+\frac{1}{2},j,k)}}{\Delta z}\right] \tag{2.15}$$

$$Hx^{n+\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2},k)} = Hx^{n-\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2},k)} + \frac{\Delta t}{\mu}\left[\frac{Ex^{n}_{(i+\frac{1}{2},j+1,k)} - Ex^{n}_{(i+\frac{1}{2},j,k)}}{\Delta y} - \frac{Ey^{n}_{(i+1,j+\frac{1}{2},k)} - Ey^{n}_{(i,j+\frac{1}{2},k)}}{\Delta x}\right] \tag{2.16}$$

$$Ex^{n+1}_{(i+\frac{1}{2},j,k)} = Ex^{n}_{(i+\frac{1}{2},j,k)} + \frac{\Delta t}{\epsilon}\left[\frac{Hz^{n+\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2},k)} - Hz^{n+\frac{1}{2}}_{(i+\frac{1}{2},j-\frac{1}{2},k)}}{\Delta y} - \frac{Hy^{n+\frac{1}{2}}_{(i+\frac{1}{2},j,k+\frac{1}{2})} - Hy^{n+\frac{1}{2}}_{(i+\frac{1}{2},j,k-\frac{1}{2})}}{\Delta z}\right] - \sigma\Delta t\left[\frac{Ex^{n+1}_{(i+\frac{1}{2},j,k)} + Ex^{n}_{(i+\frac{1}{2},j,k)}}{2}\right] \tag{2.17}$$

$$Ey^{n+1}_{(i,j+\frac{1}{2},k)} = Ey^{n}_{(i,j+\frac{1}{2},k)} + \frac{\Delta t}{\epsilon}\left[\frac{Hx^{n+\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})} - Hx^{n+\frac{1}{2}}_{(i,j+\frac{1}{2},k-\frac{1}{2})}}{\Delta z} - \frac{Hz^{n+\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2},k)} - Hz^{n+\frac{1}{2}}_{(i-\frac{1}{2},j+\frac{1}{2},k)}}{\Delta x}\right] - \sigma\Delta t\left[\frac{Ey^{n+1}_{(i,j+\frac{1}{2},k)} + Ey^{n}_{(i,j+\frac{1}{2},k)}}{2}\right] \tag{2.18}$$

$$Ez^{n+1}_{(i,j,k+\frac{1}{2})} = Ez^{n}_{(i,j,k+\frac{1}{2})} + \frac{\Delta t}{\epsilon}\left[\frac{Hy^{n+\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})} - Hy^{n+\frac{1}{2}}_{(i,j+\frac{1}{2},k-\frac{1}{2})}}{\Delta z} - \frac{Hx^{n+\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2},k)} - Hx^{n+\frac{1}{2}}_{(i-\frac{1}{2},j+\frac{1}{2},k)}}{\Delta x}\right] - \sigma\Delta t\left[\frac{Ez^{n+1}_{(i,j,k+\frac{1}{2})} + Ez^{n}_{(i,j,k+\frac{1}{2})}}{2}\right] \tag{2.19}$$

Capítulo 3

Desenvolvimento do Software

O *software* de computadores é a tecnologia mais importante no palco mundial, pelo fato de ter assumido um duplo papel; é o produto e, ao mesmo tempo o veículo para entrega do produto. Como produto, é um transformador de informações - produzindo, gerindo, modificando e exibindo informações que podem ser tão simples como um *bit* ou tão complexas quando uma vídeo aula. Como veículo, o *software* age como base para controlar o computador (sistemas operacionais), para a comunicação da informação (redes de computadores) e para criação e controle de outros programas (ferramentas e ambientes de desenvolvimento) [20].

Com toda esse importância, a área de desenvolvimento de *softwares* cresceu de forma avassaladora, exigindo assim uma padronização, haja vista que todo sistema (*software*) necessita de reparos e melhorias à medida que o tempo passa. Por estes motivos surge a engenharia de *software*, que é um conjunto de métodos e ferramentas que auxiliam no projeto, desenvolvimento, testes e manutenção de um *software*.

3.1 Modelo de Processo de Software

O modelo de desenvolvimento de software utilizado neste projeto foi o incremental. Ele consiste em um processo iterativo, que objetiva entregar o produto operacional a cada incremento [20].

Os primeiros incrementos são versões simplificadas do produto final, que oferecem capacidades mínimas de utilização por parte do usuário. A figura 3.1 ilustra como este modelo se comporta durante o tempo estimado para o projeto.

Estão descritas abaixo, as etapas deste modelo:

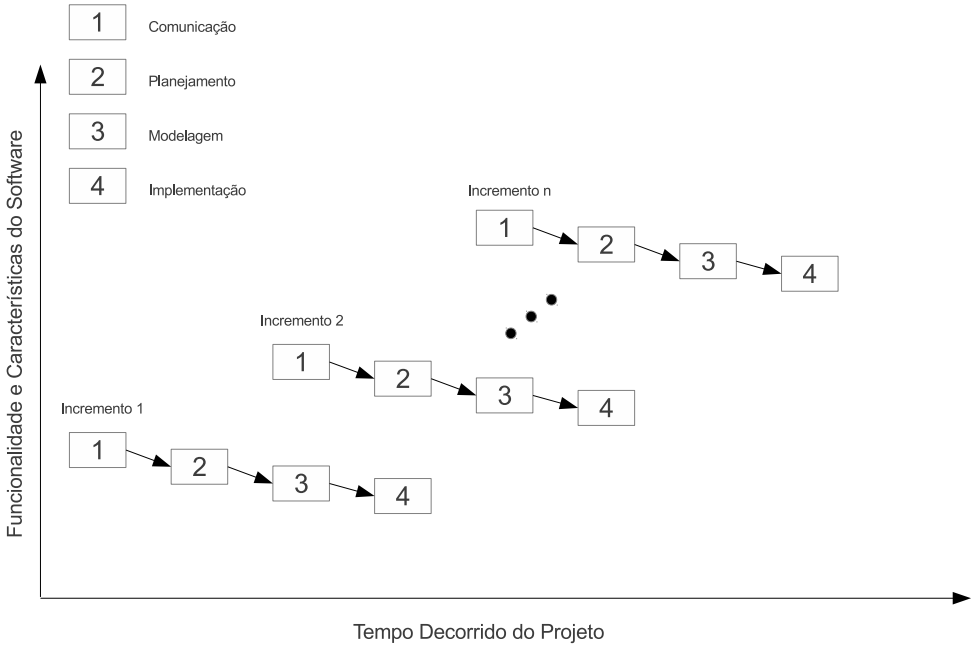


Figura 3.1: Diagrama do modelo de desenvolvimento incremental.

1. **Comunicação:** é responsável pelo levantamento do(s) propósito(s) do projeto.

2. **Planejamento:** tem o papel de definir os requisitos do trabalho.

3. **Modelagem:** define as ferramentas, linguagens, etapas a serem desenvolvidas.

4. **Construção:** é o processo de codificação e testes.

5. **Implementação:** responsável pela entrega do protótipo (inicial ou final).

3.2 Tecnologias de Programação Usadas

As tecnologias de programação usadas neste projeto foram: o motor de jogos *Irrlicht*, a plataforma *Qt-Creator* e a linguagem C++.

O *Irrlicht* é uma máquina de jogo de alta performance escrita em linguagem C++ [21]. Tem no fato de ser *open source* (código aberto), uma das suas principais vantagens em relação a outras *engines* (proprietárias). Abaixo estão listadas, outras características deste motor gráfico [22].

1. Renderização em tempo real de alta performance 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 : C/C++, Java, Delphi, entre outras.
6. É limpa, fácil de entender, e bem documentada.

O *Qt-Creator* é um poderoso ambiente multi plataforma que permite a criação de diversas aplicações *web* e também *desktop*. Sua importância, neste trabalho, está relacionada com oferecer suporte para a criação de uma interface *desktop*, a qual pudesse ser usada em diferentes sistemas operacionais (Linux, Windows ou Macintoch) e com arquiteturas dissemelhantes (32 ou 64 bits), para modelar tanto objetos 3D básicos como ambientes *indoor*.

O C++ é uma linguagem orientada a objeto utilizada por milhares de programadores em inúmeros domínios de aplicação. Ela é sustentada por um conjunto vasto de bibliotecas, livros, revistas técnicas e conferências. É tão robusta que está presente no desenvolvimento (total ou de partes) de muitos sistemas operacionais.

Tanto as plataformas (*Irrlicht* e *Qt*) quanto a linguagem de programação C/C++, foram escolhidas por serem muito usadas pelo comunidade desenvolvedora por mostrarem robustez, boa documentação e terem uma conexão (tanto o *Irrlicht* quanto o *Qt* usam a linguagem C++).

3.3 Classes

O uso do conceito de classes, na implementação deste trabalho, se dá ao fato de que a linguagem C++ baseia-se no paradigma da orientação a objeto. A orientação a objeto é um conjunto de conceitos no qual o desenvolvimento (analise, projeto e programação) de sistemas de software está baseada na composição e iteração entre diversas unidades de software chamadas de objetos. Alguns de seus conceitos essenciais são:

- **Classe**: representa um conjunto de objetos com características afins. Uma classe define o comportamento dos objetos através de seus métodos, e quais estados ele é capaz de manter através de seus atributos.

- **Objeto**: um objeto é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ele, assim como se relacionar e enviar mensagens a outros objetos.
- **Atributos**: são elementos que definem a estrutura de uma classe.
- **Métodos**: especificam a forma como os dados de um objeto são manipulados.
- **Herança**: é um princípio que permite que classes compartilhem atributos e métodos.

As classes criadas para este projeto foram: *IrrViewer, Cena, IrrNode, ReceiverEvent* e *MainWindow*. O diagrama *uml* mostrado na Figura 3.2, está representando o relacionamento entres estas classes.

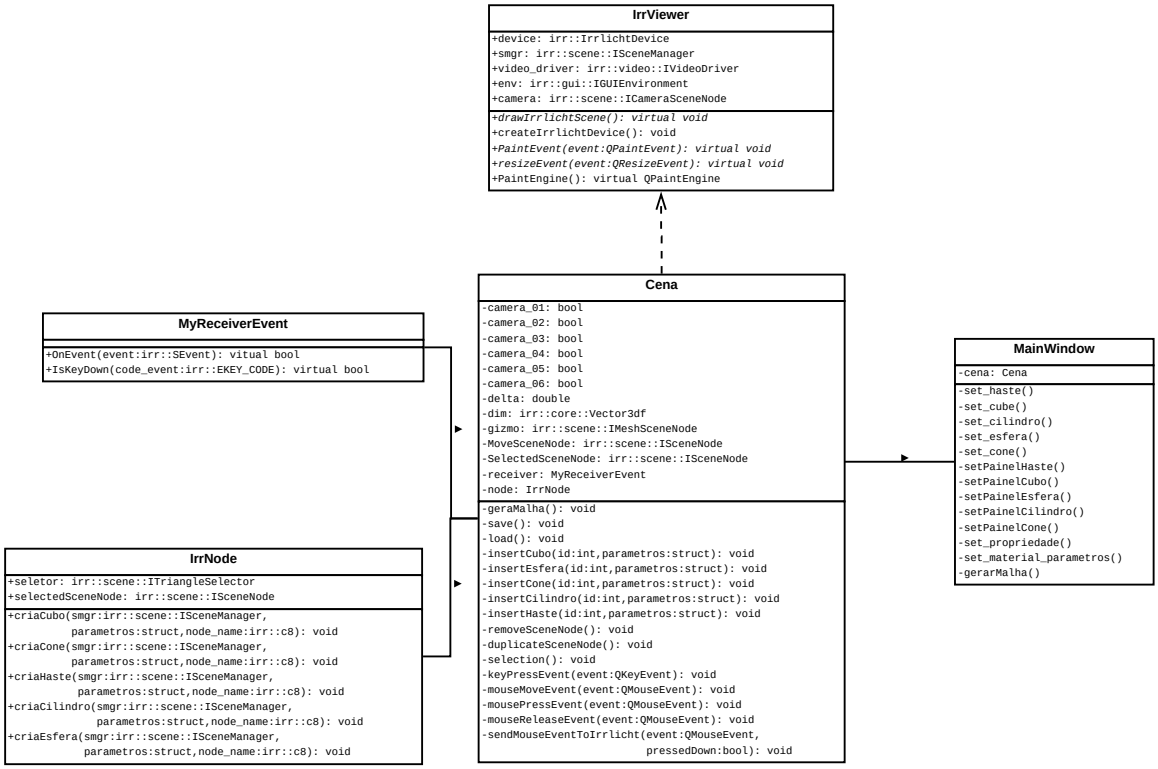


Figura 3.2: Diagrama *uml* do relacionamento das classes desenvolvidas neste projeto.

O primeiro passo no desenvolvimento deste trabalho foi a união do motor de jogos *Irrlicht* com a plataforma *Qt-Creator*, necessitava-se que a cena *Irrlicht* fosse reconhecida como uma janela padrão do *Qt* (para poder introduzi-la na interface). Para tanto, foi necessário o estudo das classes básicas destes dois universos, assim como suas principais características.

Esta união foi concretizada por meio da criação da classe *IrrViewer*, que contêm tantos os métodos que caracterizam uma classe *Widget* quanto os necessários para criação de um cenário *Irrlicht*. A Figura 3.3 ilustra os principais atributos e métodos desta classe.

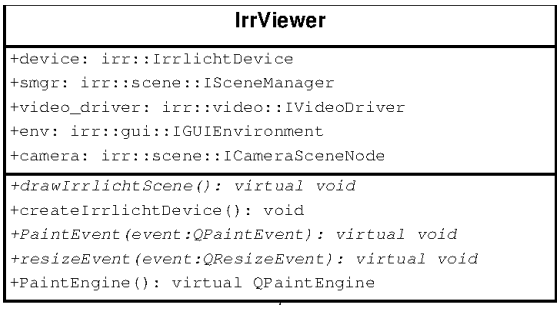


Figura 3.3: Classe IrrViewer.

Assim, com o *Qt-Creator* reconhecendo essa nova classe como uma de suas janelas(Widget), pode-se passar para segunda fase do projeto (desenvolvimento de outras classes e métodos necessários para preenchimento dos requisitos pré-estabelecidos).

A classe MyReceiverEvent(Fig. 3.4) foi criada com intuito de comunicar o evento de *mouse* do *Qt* com a *engine Irrlicht*. Desta forma, quando este evento for disparado (chamado), passará para o objeto desta classe o estado do *mouse* que será tratado e enviado para o *Irrlicht*.

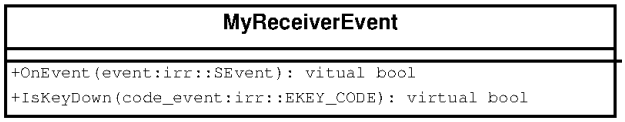


Figura 3.4: Classe ReceiverEvent.

O papel de dar suporte à criação de objetos tridimensionais para o cenário virtual foi designado a classe *IrrNode* (Fig. 3.5). Portanto esta contém a implementação dos métodos de construção dos todos os objetos 3D básicos suportados pela interface criada neste trabalho, tais como os de criação de cubos, esferas, hastes, cilindros e cones.

Já os atributos e métodos necessários para modificação do cenário virtual estão presentes na classe *Cena* (3.6). Sendo ela, desta forma, a responsável pelo gerenciamento de todo e qualquer evento que ocorra dentro do AV. Ela faz isso, utilizando-se dos : objetos das classes *IrrNode* e *MyReceiverEvent*, dos métodos herdados da classe *IrrViewer* (olhar Figura 3.2) e dos seus próprios métodos e atributos.

Por fim, temos a classe *MainWindow*, que utiliza-se de um objeto da classe *Cena* para comunicar as alterações realizadas na interface com a janela Irrlicht e vice-versa. Desta forma, fazendo o tratamentos dos eventos de botão, mudança nos valores dos painéis laterais e dos campos de posição e rotação. A Figura 3.7 mostra os principais atributos e métodos deste

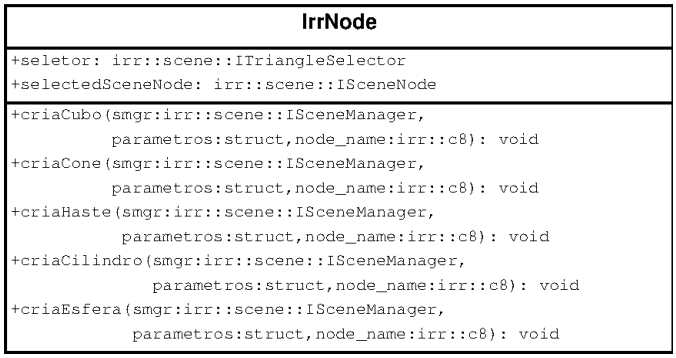


Figura 3.5: Classe IrrNode.

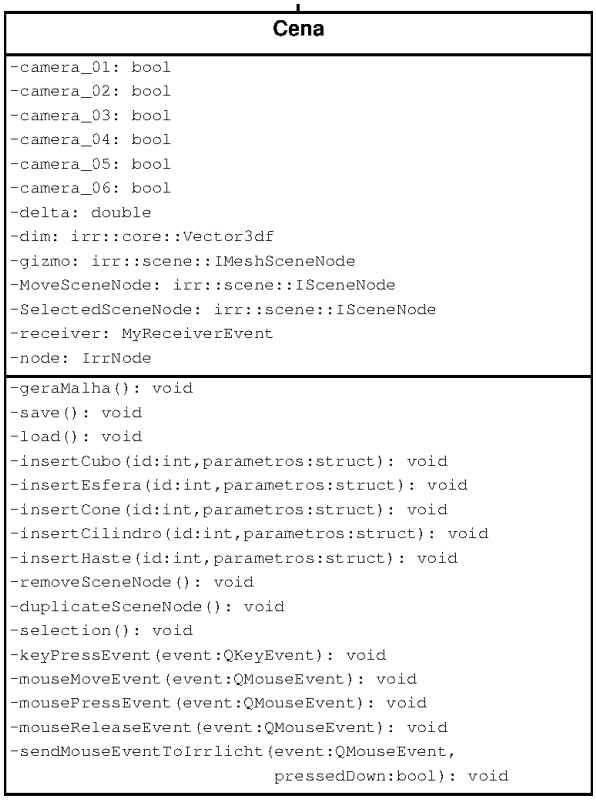


Figura 3.6: Classe Cena.

classe.

3.4 Interface

O software desenvolvido neste projeto está ilustrado na Figura 3.8. Seu layout e suas ferramentas foram baseada nos softwares de modelagem mais utilizados no mercado, que são: Blender, 3DStudio e Maya. A sua estrutura foi dividida em 4 áreas: barra superior, janela da

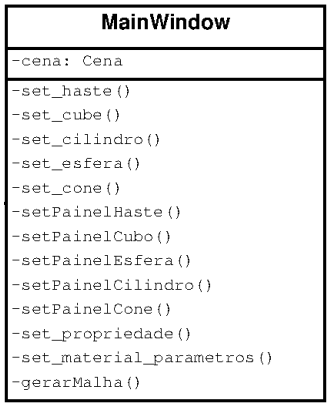


Figura 3.7: Classe *Mainwindow*.

cena, painéis laterais e barra inferior.

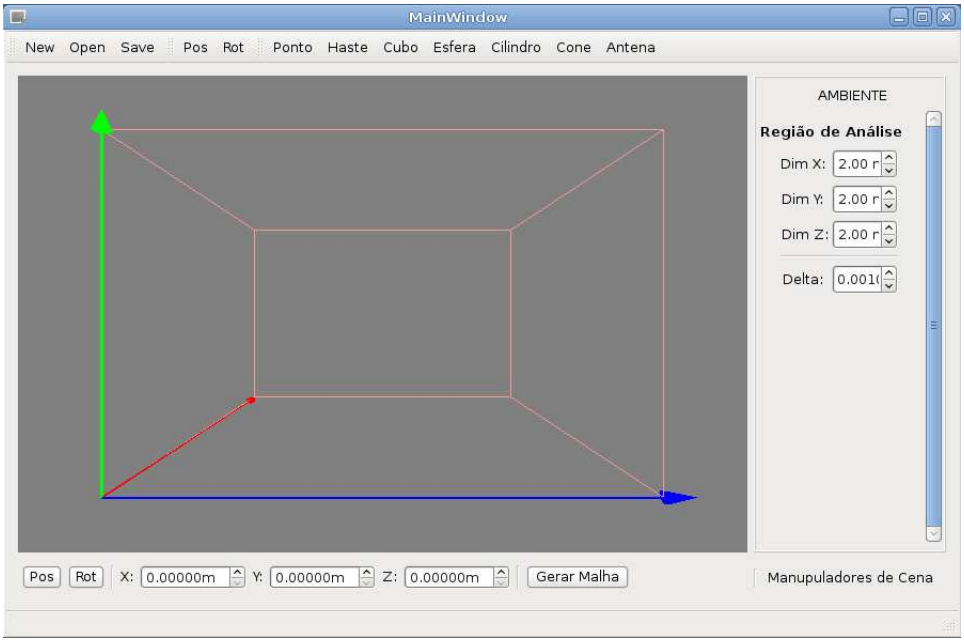


Figura 3.8: Layout da interface.

Na barra superior, Figura 3.9, encontra-se primeiramente o botão *New*, que tem o propósito de criar uma nova cena *Irrlicht*. Quando pressionado, ativa uma janela (Figura 3.10) que requisita os dados necessários para criação da região de análise, tais como valores de dimensionamento (tamanho em x, y e z) e o valor do $delta$ (dimensão da célula de Yee). Em seguida vem o botão *Open*, que carrega uma cena anteriormente salvada (caso exista). Depois dele tem-se o botão *Save*, que salva a cena atual em uma arquivo chamado *Map.in*, que contém todas as características dos objetos (tamanho, tipo e parâmetros físicos), assim como suas posições.

Figura 3.9: Barra Superior interface.

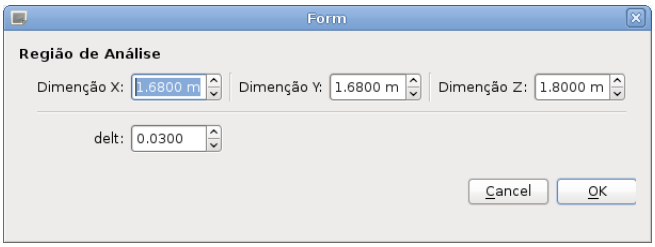


Figura 3.10: Janela com as característica da região de análise.

Mas em frente, encontram-se os botões *Pos* e *Rot*, os quais dão suporte à visualização e manipulação do posicionamento e dos ângulos de cada objeto selecionado, respectivamente. Logo em seguida encontram-se 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 a posição e os parâmetros previamente especificados.

A janela da cena, Figura 3.11 é uma região do software onde se visualiza o universo virtual. Nesta região, é possível realizar manipulações através do mouse como seleção, translação e rotação. Ela está diretamente relacionada a classe *IrrViewer*, pois é uma instância (objeto) desta classe.

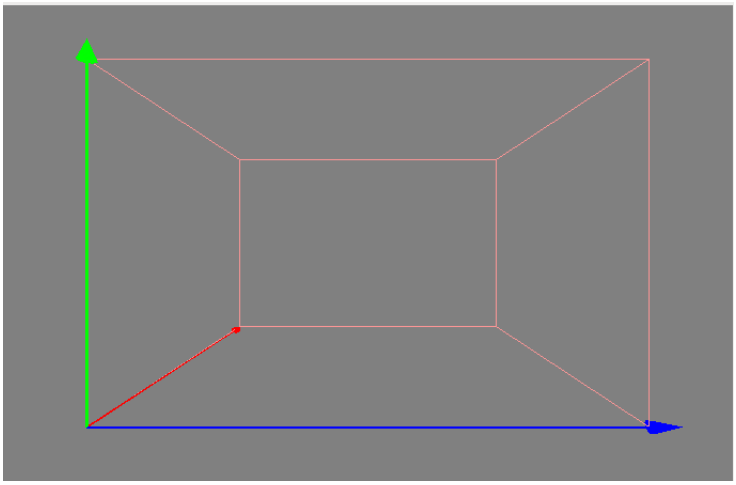


Figura 3.11: Janela da Cena.

O painel lateral, Figura 3.12 é uma área de visualização e manipulação das características

relacionadas ao tamanho e parâmetros físicos dos objetos.

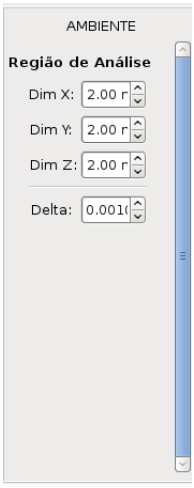


Figura 3.12: Painel lateral dos parâmetros da região de análise.

Na parte inferior da interface(Figura 3.13) encontram-se os botões *Pos* e *Rot*, que têm as mesmas funcionalidades dos já citados da barra superior. Em seguida há um conjunto de itens que possibilitam a visualização e manipulação das coordenadas relativas a posição e rotação do objeto selecionado (X, Y e Z). E por fim, o botão *gerar Malha*, o qual tem a função de gerar e salvar a malha representativa de todos os objetos construídos no cenário virtual.

Figura 3.13: Barra inferior da interface.

3.5 Funcionalidades

3.5.1 Atalhos de Teclado

A facilidade de navegação e manipulação, são características fundamentais para qualquer software de modelagem. Os atalhos de teclado são criados com o intuito de reduzir esforço e tempo do usuário. Esta ferramenta é um conjunto de teclas que ao serem pressionadas realizam uma ação. As ações realizadas pelas teclas de atalho, normalmente podem ser realizada por outras ferramentas como botões, menus, etc.

A Tabela 3.1 contém todos os atalhos de teclado criados para este projeto.

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.

3.5.2 Eventos de Colisão

Os eventos de colisão possibilitam que duas ou mais formas geométrica existentes no AV possam colidir. Neste trabalho utilizou-se a colisão por reta, que é ortogonal ao plano de visualização da câmera. Seu ponto inicial fica posicionado no centro ecrã (tela), já seu o ponto final é deslocado do inicial por uma distância consideravelmente maior do que qual uma das arestas da região de análise. Este ponto final responde ao evento de *mouse*, ou seja, no caso da ocorrência de um *click* em um determinado ponto do cenário 3D a reta de colisão terá o seu ponto inicial recebendo o valor central da tela e o final obtendo o valor da posição do *mouse*. No caso de esta reta encontrar (colidir) algum objeto entre seus pontos ele será o objeto selecionado.

Para diferenciar, visualmente, objetos selecionados dos não selecionados, utilizou-se da técnica de representação *wireframe*. Portanto, quando ocorrer um evento de colisão e um objeto é selecionado, a visualização deste objeto muda, passando a ser representado somente por suas arestas e vértices. A Figura 3.14 demostra este fato.

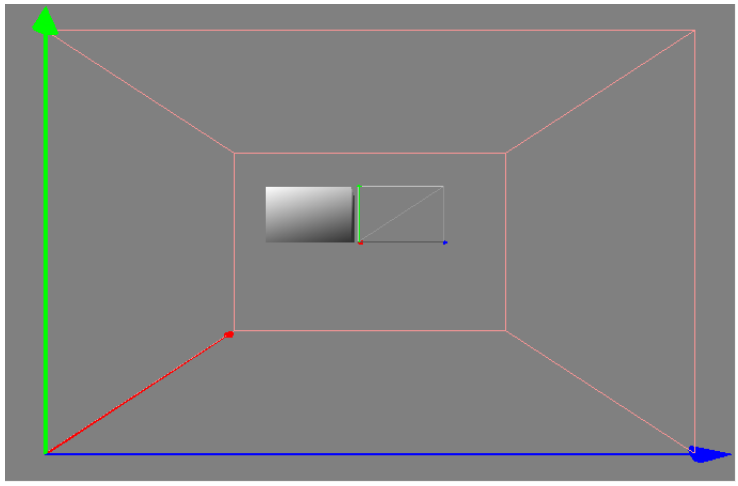


Figura 3.14: Ilustração do evento de colisão com o objeto cubo.

3.5.3 Visualizações

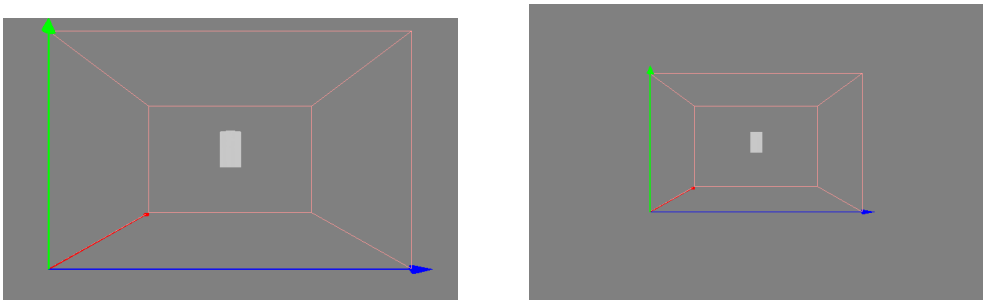
A possibilidade de visualização de um cenário por vários ângulos é fundamental quando se está modelando um ambiente 3D, principalmente quando o cenário modelado será utilizado em simulações eletromagnéticas, onde qualquer erro estrutural poderá gerar resultados não desejados (sem sentido). Para sanar esse problema, o software confeccionado neste projeto, permiti visualizar o AV por seis câmeras diferentes: frontal(padrão), lateral esquerda, lateral

direita, traseira, topo e base.

3.5.4 Aproximação e Afastamento

Tem a finalidade de, juntamente com as seis visualizações possíveis(seis câmeras diferentes), facilitar a navegação e alteração do cenário virtual. Sendo uma ferramenta importante na modelagem, por permitir a aproximação e afastamento de um determinado região ou objeto selecionado dentro do ambiente virtual. As teclas de atalho associadas a aproximação são $Shift + P$ ou W e as associadas aos afastamento são $Shift + O$.

As figuras a seguir ilustram o uso desta ferramenta. A Figura 3.15(a) mostra um objeto a uma distância padrão (sem afastamento ou aproximação). Na Figura 3.15(b) pode ser visto o mesmo objeto após um determinado afastamento (utilizando $Shift + O$). Já as Figuras 3.16(a) e 3.16(b) mostra os efeitos da aproximação normal (usando o atalho $Shift + P$) e a por seleção (dada pela tecla W).



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

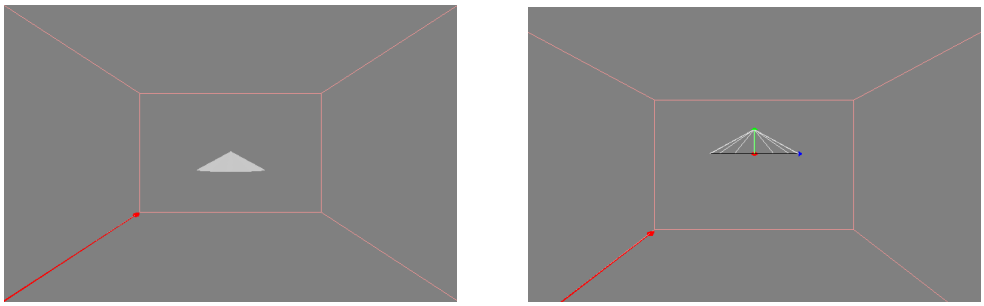
(b) Visualização de objeto cilindro com afastamento.

Figura 3.15: Visualização por afastamento.

3.5.5 Conexão com o LANE-SAGS

Depois de implementar todas as funcionalidades do software, passou-se para o último requisito proposto para este projeto, a conexão com o simulado LANE-SAGS. Esta conexão foi realizada por meio de dois arquivos.

O primeiro arquivo é o bd.in (bd - bloco dielétrico), que contém os prismas retangulares (células) que representam, por enumeração de ocupação espacial, os objetos contínuos mode-



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

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

Figura 3.16: Visualização por aproximação.

lados na interface. A sequência de gravação de cada prismas no arquivo segue a forma: célula inicial X, célula final X, célula inicial Y, célula final Y, célula inicial Z, célula final Z, ϵ_r, σ e μ_r. Além do arquivo bd.in, é gerado um segundo arquivo chamado *simconf.in*(simconf-configuração da simulação) contendo as informações referentes às: dimensões das células de Yee (em metros) e a dimensão em células do Espaço Delimitador (região de análise). Com os arquivos gerados e a conexão estabelecida, basta inserir (utilizando o LANE-SAGS) alguns fatores fundamentais para efetuar uma simulação de propagação eletromagnética, tais como: antena, planos de visualização, fonte e corrente. O diagrama mostrado na Figura 3.17, ilustra o processo de execução de uma simulação utilizando os dois softwares.

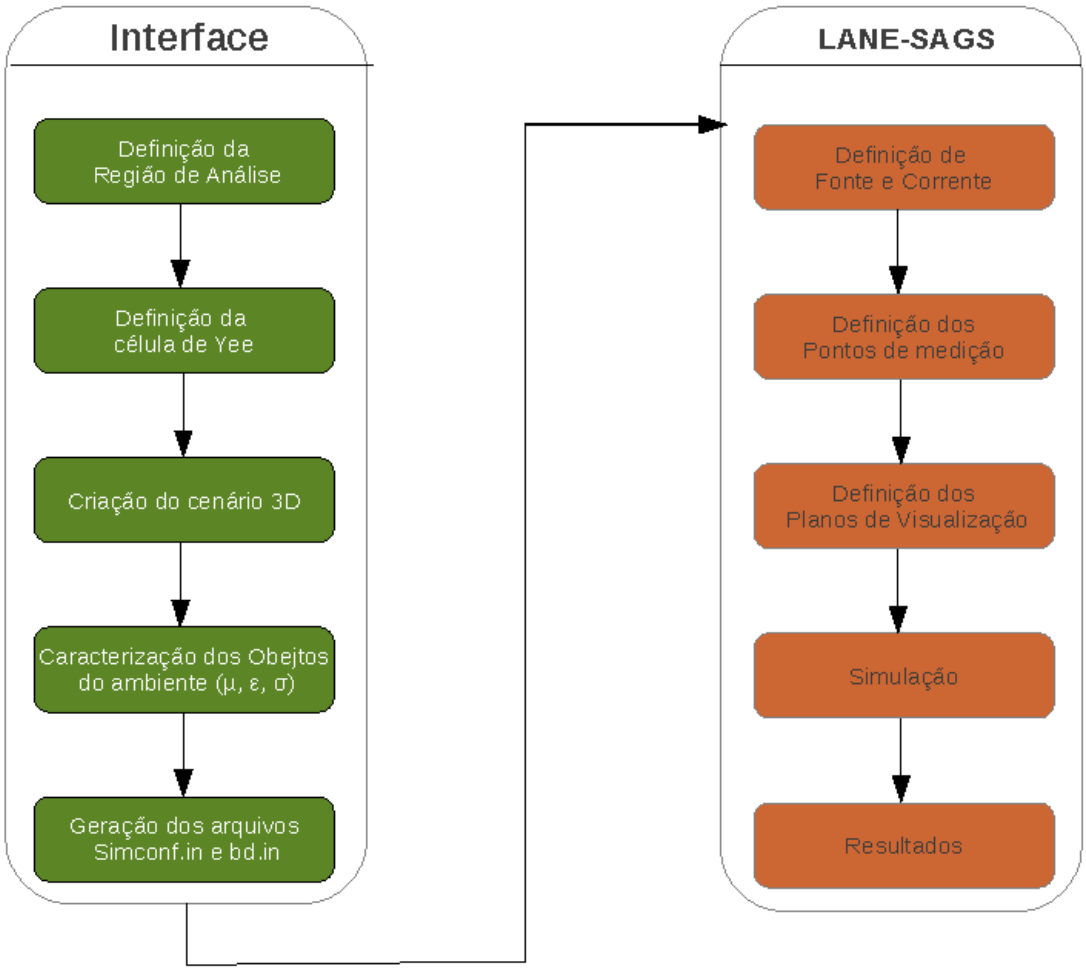


Figura 3.17: Diagrama de conexão com o LANE-SAGS.

Capítulo 4

Modelagem, Aplicação e Resultados

4.1 Modelagem e Aplicação

Com a finalidade de validar a interface proposta neste trabalho, foi construído um ambiente virtual baseado nas características de um ambiente real (escritório). Estás características foram obtidas com o auxílio da planta baixa do ambiente proposto, ilustrada pela Figura 4.1.

Neste ambiente, utilizando o modelo numérico aqui desenvolvido, foram feitos registro transitórios do campo elétrico transmitido por uma antena banda larga em pontos específicos do cenário (mantendo a porta fechada), o que possibilitou a obtenção da resposta à propagação do sinal desta antena.

A maioria dos objetos presentes no ambiente real, no momento da realização das medições experimentais [23], foram modelados. As especificações destes objetos estão na Tabela 4.1.

Alguns destes objetos estão dispostos em alturas diferentes em relação ao piso. As bancadas e o teto estão a uma altura de 0,75 metro e 2,73 metros, respectivamente. Os computadores e monitores foram colocados na célula imediatamente acima do nível das bancadas. Dessa forma, o plano relativo às bases dos computadores coincide com o plano das bancadas.

As dimensões gerais desse cenário usadas na interface foram:

- Δ(Aresta da célula de Yee) utilizada foi de $0,03m$.
- *Região de Análise* com dimensões: X = $15m$, Y = $3m$ e Z = $15m$.
- *A* lista das caraterísticas físicas com seus respectivos materiais é representada na Tabela 4.2 [23].

Tabela 4.1: Objetos e suas dimensões.

Objetos	**Dimensão** X	**Dimensão** Y	**Dimensão** Z
Computadores (gabinetes)	0,18m	0,43m	0,42m
Monitores	0,39m	0,31m	0,39m
Divisórias 1 e 2	3,27m	1,5m	0,03m
Paredes divisórias 1 e 2	0,03m	1,5m	1m
Bancadas de 1-4	3,21m	0,03m	0,5m
Bancada 5	2,94m	0,03m	0,5m
Bancada 6	4,68m	0,03m	0,5m
Bancada 7	0,5m	0,03m	3,18m
Bancada 8	4,26m	0,03m	0,5m

Tabela 4.2: Tabela de materiais e parâmetros físicos.

Elementos dos Ambiente	Material	ϵ_r	σ	μ_r
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	1×10^{11}	1
Divisórias	Vidro	5	5×10^{-4}	1
Piso	Concreto	5	1	1

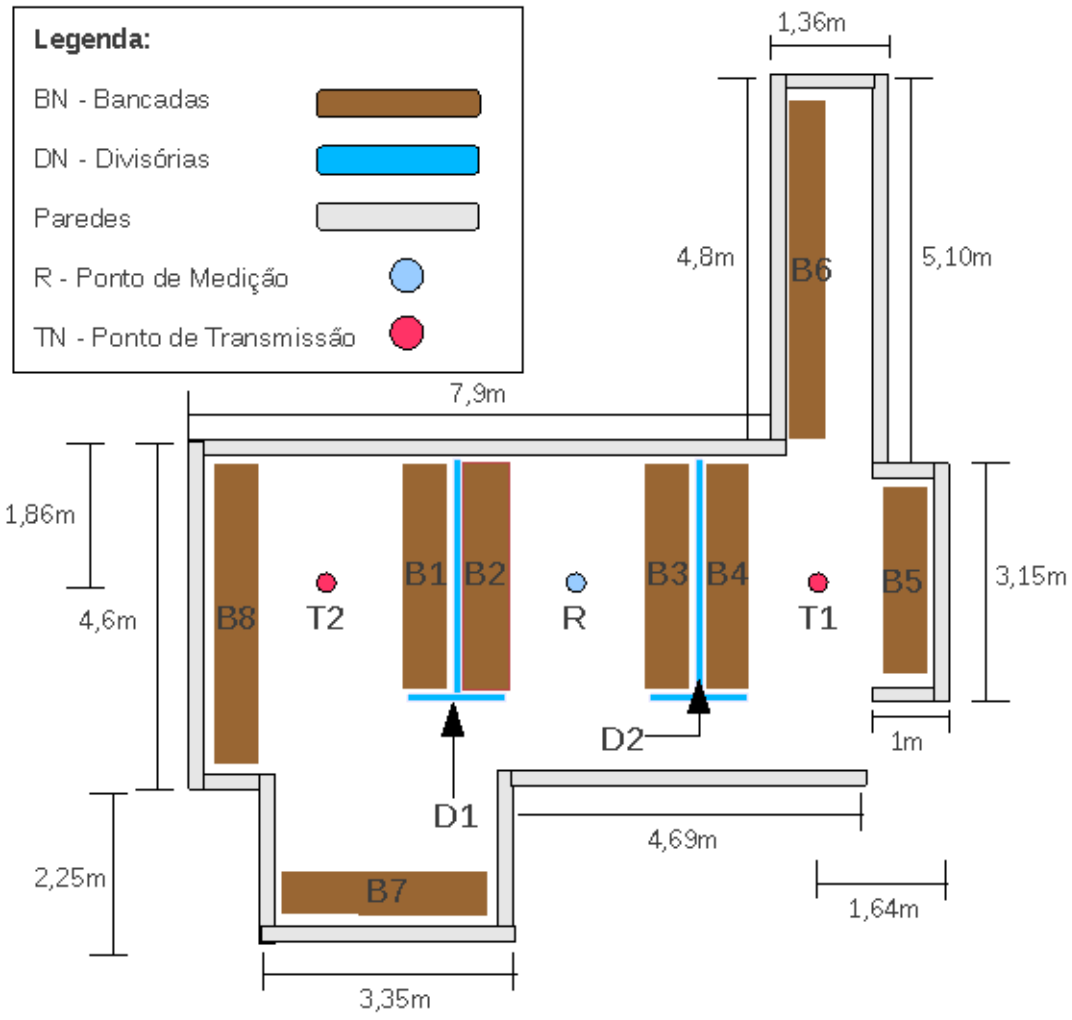


Figura 4.1: Planta baixa do ambiente simulado e pontos de transmissão e recepção.

As características do solo (abaixo do piso) são as de um meio rochoso. Os parâmetros $\epsilon_r = 50$ e $\mu_r = 2{,}28$, para este tipo de solo, foram obtidas em [24].

O resultado da modelagem (usando a interface) foi o conjunto das estruturas mostradas nas Figuras 4.2 e 4.3 . A disposição dos objetos no AV, foi realizada de forma a corresponder com o posicionamento real no momento das medições experimentais [23]. As Figuras 4.4, 4.5 e 4.6 mostram o ambiente modelado e seus objetos com mais detalhe.

4.1.1 Criação e Introdução da Antena

Depois de o ambiente ter sido modelado na interface, surgiu a necessidade de introduzir a antena. Como a que foi usada no experimento era altamente complexa (antena discônica), foi necessário projetar um antena com geometria mais simples, mas que de forma semelhante

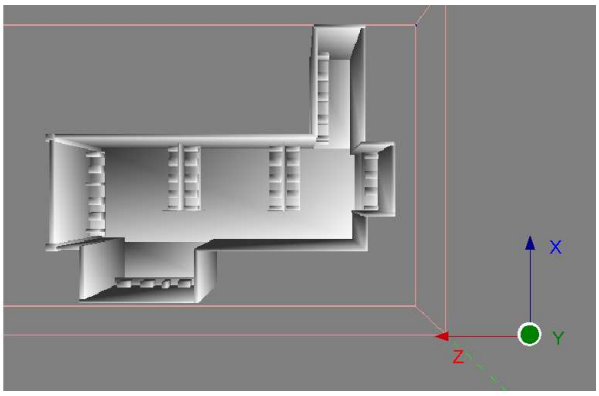


Figura 4.2: Visão superior do ambiente modelado neste trabalho.

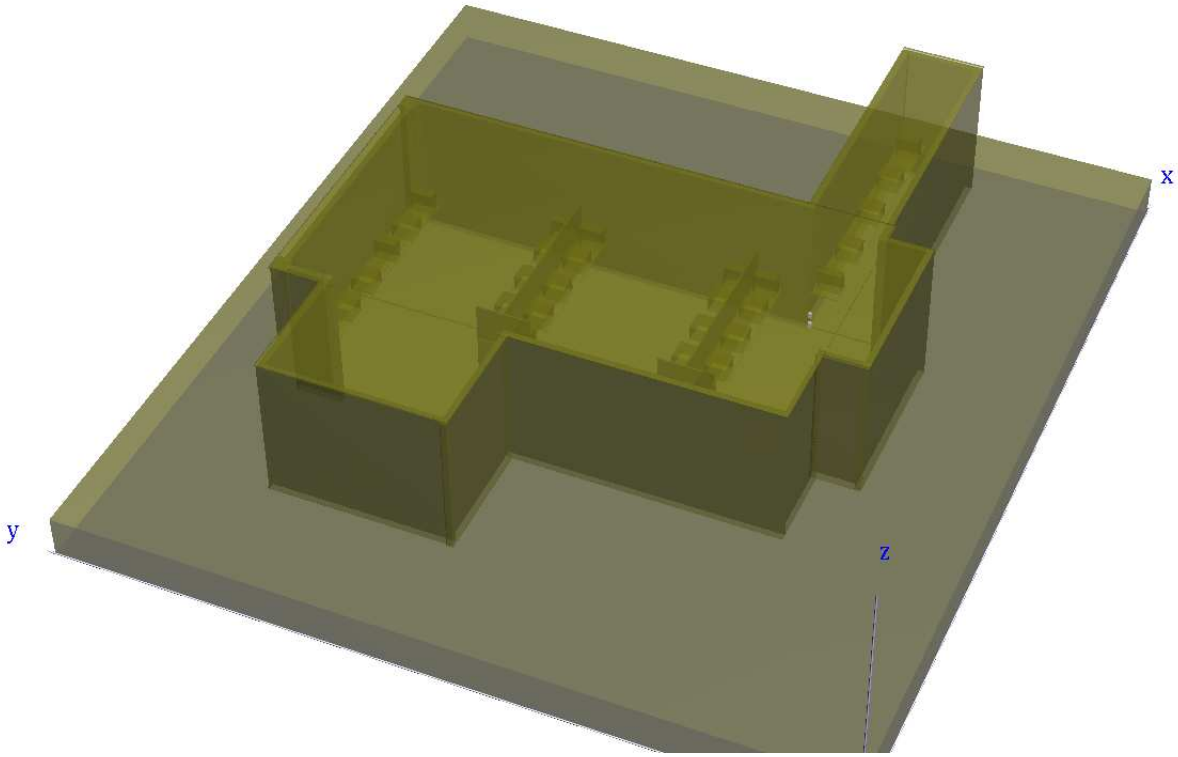


Figura 4.3: Visão em perspectiva do cenário no simulador LANE-SAGS.

à real. Após fazer uma análise, percebeu-se que antena usada durante as experimentações [23] funcionava como dipolo. Desta forma, alguns teste foram realizados, de forma a obter um representação aceitável para a antena. Esta deveria ter um diagrama omnidirecional e perda de retorno abaixo de -10dB na faixa de frequência de interesse.

Para todos os modelos de antenas testados, foi utilizada uma sinal que varia no tempo de acordo com a função monociclo gaussiano (derivada do pulso gaussiano), tal como descrito

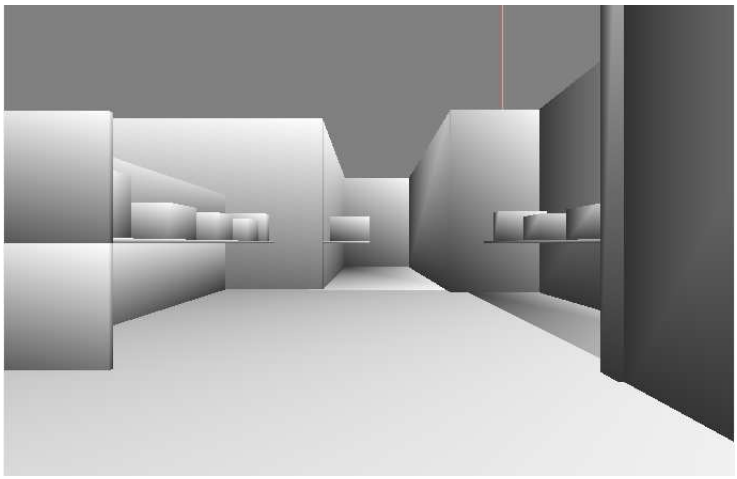


Figura 4.4: Visão interna do ambiente modelado (corredor 1).

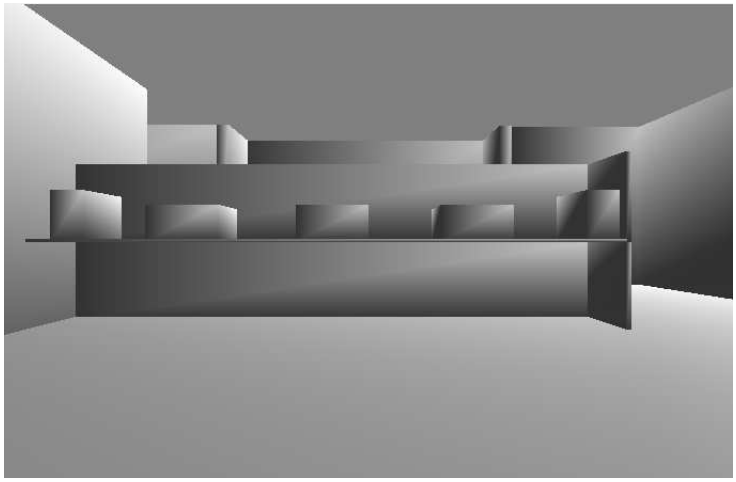


Figura 4.5: Visão interna do ambiente modelado (bancada).

matematicamente por (4.1) e graficamente pela Figura 4.7(a). Utilizou-se a transformada de Fourier para verificar se a faixa de frequência, do sinal usado, esta na banda adequada (entre 700MHz-900MHz). O espectro do sinal monociclo gaussiano (4.1) é ilustrado pela Figura 4.7(b).

$$f(t) = A_p \exp\left(-\left(\frac{t - 3.0\,T}{T}\right)^2\right)\left(\frac{t - 3T}{T}\right) \tag{4.1}$$

Onde:

$A_p = 1$ e T = 6.53846×10^{-10}

A primeira antena-teste montada ilustrada pela Figura 4.8(a), juntamente com sua representação no LANE-SAGS(Figura 4.8(b)). Esta antena dipolo é similar a usada em [25]. Os

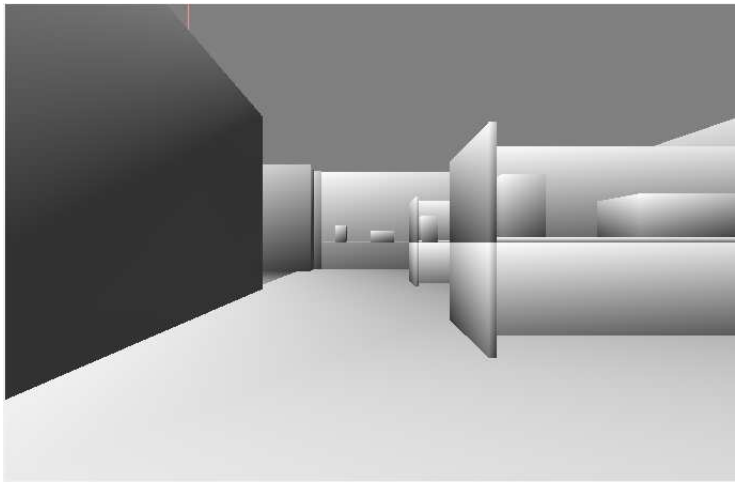
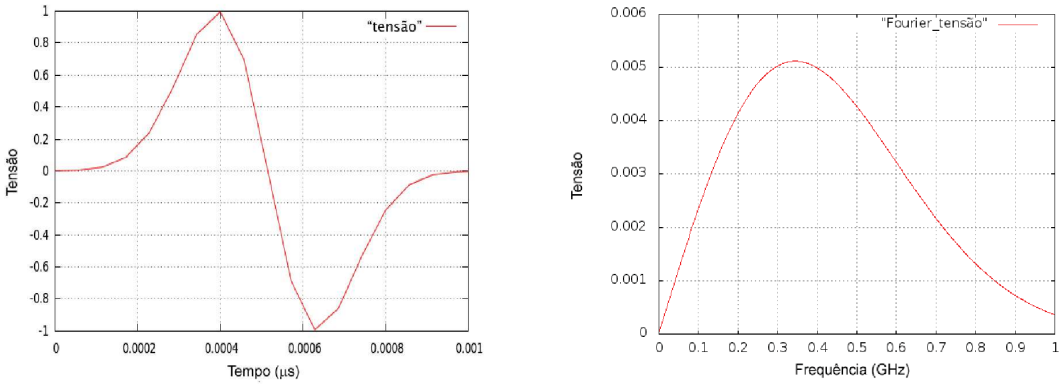


Figura 4.6: Visão interna do ambiente modelado (corredor 2).



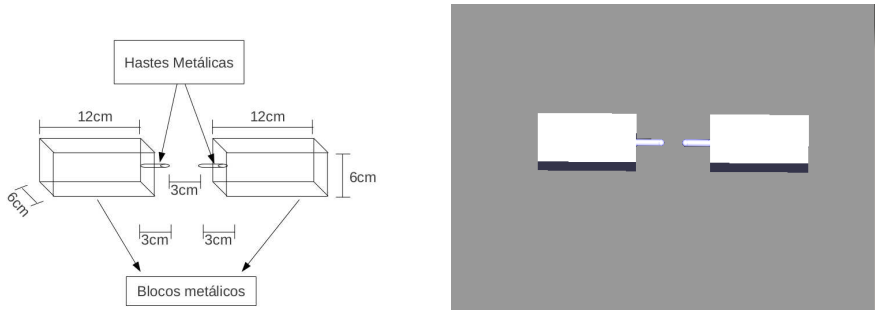
(a) Gráfico da fonte no tempo.

(b) Gráfico da transformada de Fourier da fonte.

Figura 4.7: Fonte de tensão usada para excitar as antenas modeladas.

dois blocos têm as mesmas dimensões, assim como as duas hastes metálicas. Entre os terminais da antena (espaço entre as duas hastes), a tensão mostrada na Fig. 4.7(a) foi estabelecida através da excitação do campo elétrico. Para fins de projeto, foi calculada a corrente transitória em uma das hastes (na célula vizinha à célula de excitação do campo elétrico) por meio do rotacional do campo magnético que circunda ao redor da haste.

Com os dados transitório de tensão $V(t)$ e corrente $I(t)$, foi calculado o coeficiente de reflexão através da relação 4.2. Em (4.2), Z é a impedância da antena, dada por $Z = F(V(t)/F(I(t)))$, onde F é o operador da transformada de Fourier e $Z_S = 50\Omega$ é a impedância da carga. Por meio deste coeficiente Γ, pode-se calcular a perda de retorno, associada a essa antena(Equação 4.3). Como pode-se observar na Figura 4.9, a antena está trabalhando abaixo



(a) Projeto inicial de antena.

(b) Visualização do protótipo no simulador LANE-SAGS.

Figura 4.8: Antena dipolo tradicional.

de -10dB em uma faixa diferente da desejada (700MHz-900MHz). Portanto, suas dimensões e parâmetros necessitaram ser modificados.

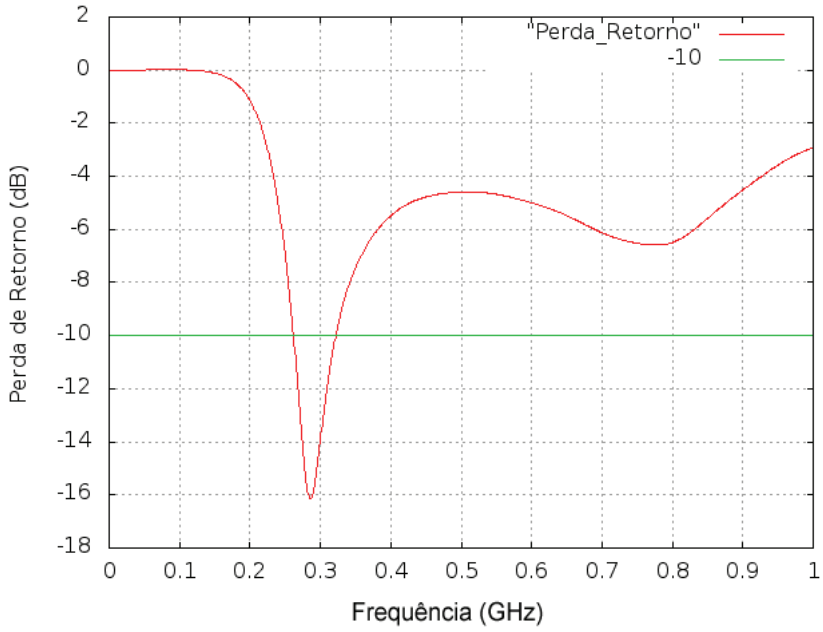


Figura 4.9: Perda de retorno da antena dipolo tradicional(primeira antena modelada).

$$\Gamma = \frac{Z - Z_S}{Z + Z_S} \tag{4.2}$$

$$RL(dB) = -20log_{10}|\Gamma| \tag{4.3}$$

Após alguns testes, verificou-se que a diminuição na dimensão dessa antena afetava de

forma positiva o resultado da perda de retorno. Porém, havia um problema ligado ao comprimento das arestas de célula de Yee (Δ), que diminuiria de forma proporcional à redução das dimensões da antena, inviabilizando as simulações devido ao alto custo computacional relativo ao método FDTD. Para solucionar este impasse, optou-se pelo uso de um capacitor ($3,7 \times 10^{-5}$ farad) acoplado a um dos blocos metálicos, afim de sintonizar a antena com a banda de interesse (casamento de impedâncias). Este arranjo possibilitou o uso da antena sem alterar suas dimensões (tamanho) e, principalmente, sem modificar o parâmetro Δ. Sabe-se que o aumento do comprimento da antena deixa-a indutiva e que a antena dipolo é omnidirecional [26].

A configuração da antena adaptada está ilustrada na Figura 4.11(a). A Figura 4.11(b) mostra o modelo no LANE-SAGS. Todos os cálculos tiveram que ser refeitos para nova antena, obtendo-se a perda de retorno desejada. A Figura 4.10 mostrada os resultados para a antena adaptada (em vermelho) e para a antena original (em verde).

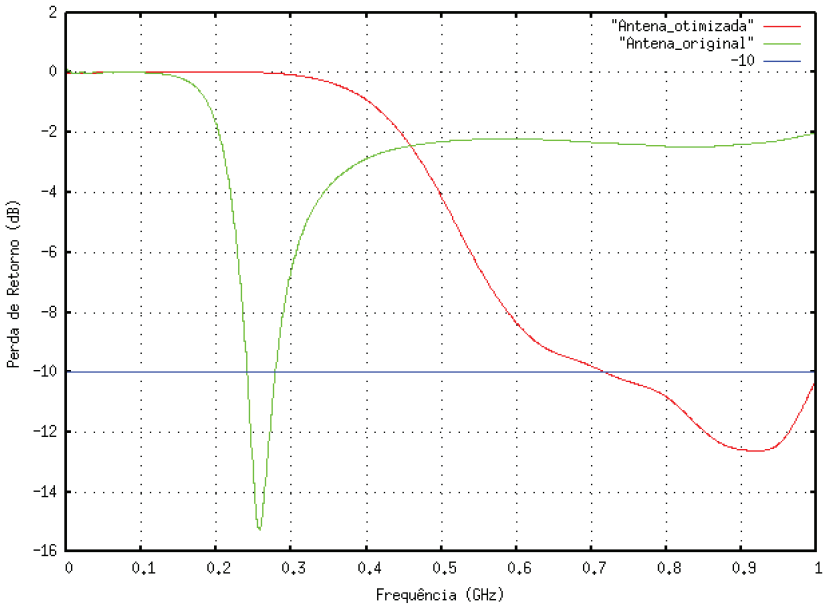
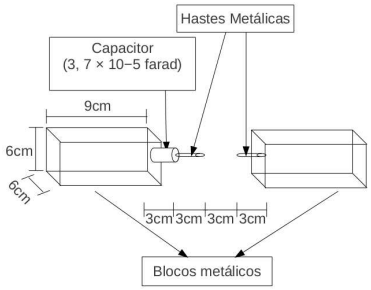


Figura 4.10: Comparação entre as perdas de retorno da antena otimizada(adaptada) e normal(original).

4.1.2 Resultados

4.1.2.1 Caso 01

O caso 01 refere-se à introdução da antena modelada na posição T1 da Figura 4.1. Ela foi posicionada a uma altura de 1,5m do piso e polarizada em z. Com isso, foi feita a simulação da propagação do pulso monociclo Gaussiano a partir da antena projetada. Obteve-se, além da

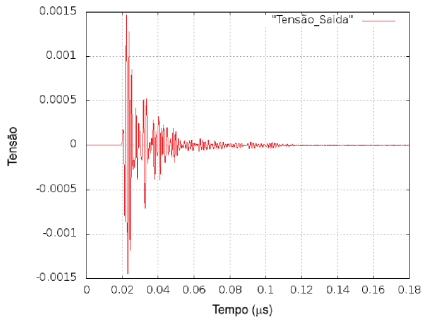


(a) Diagrama da antena dipolo adaptada.

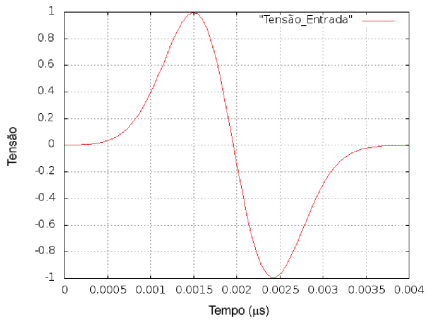
(b) Visualização da antena adaptada no simulador LANE-SAGS.

Figura 4.11: Antena dipolo adaptada.

tensão calculada no ponto R(mesma altura do ponto T1) do cenário modelado, a tensão entre nos terminais da antena transmissora. Esse sinais estão mostrados nas Figuras 4.12(a) e 4.12(b), respectivamente.



(a) Gráfico da tensão no Ponto R.



(b) Gráfico da tensão no Ponto $T1$(entre as hastes da antena).

Figura 4.12: Tensões obtidas pela simulação.

Por meio desses dados de entrada e saída, desejava-se calcular o perfil de potência e retardo do canal, dado por 4.4,para este ambiente virtual. Todavia, para que isso fosse possível, era necessário ter a resposta impulsiva $h(t)$ desse canal. A Figura 4.13 mostra o processo de obtenção da resposta em frequência e a Figura 4.14 mostra esta resposta.

$$P_h(\tau) = |h(\tau)|^2 \tag{4.4}$$

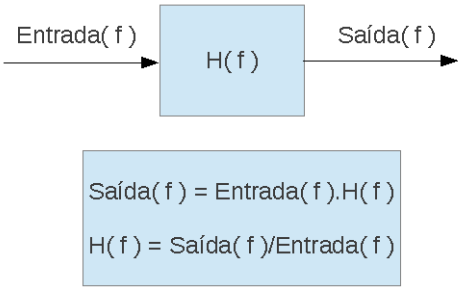


Figura 4.13: Diagrama de bloco ilustrando a operação linear de obtenção da função de transferência $H(f)$.

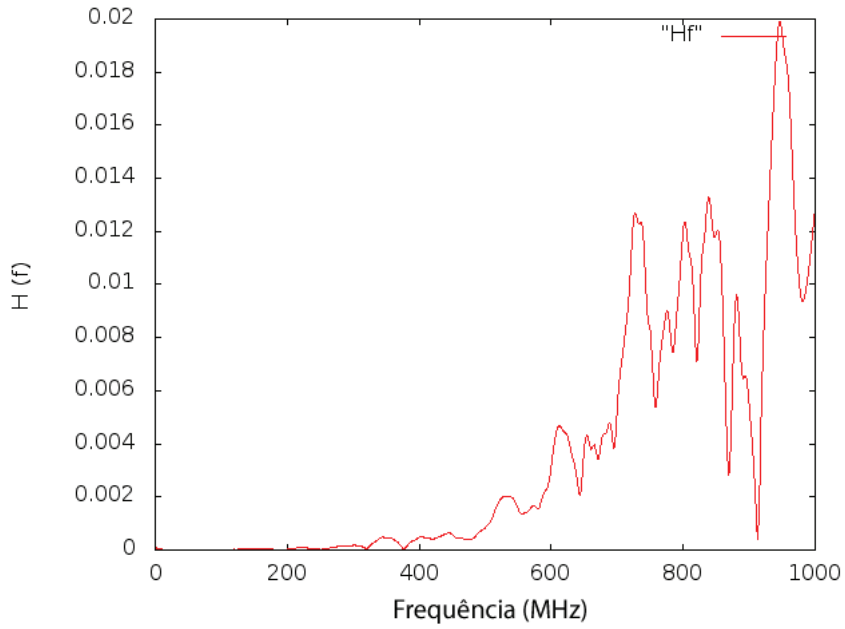


Figura 4.14: Função de transferência $H(f)$ do canal de propagação, considerando-se os pontos T1 e R.

Logo, as tensões de entrada(referente ao ponto T1) e saída (referente ao ponto R), usando a transformada de Fourier, foram passadas para o domínio da frequência e então divididas ponto a ponto. Com isso, obteve-se a resposta em frequência, $H(f)$ da Fig. 4.14. Usando a Equação 4.5, referente a anti-transformada de Fourier, obtém-se a resposta impulsiva do canal. Através dela, foi possível comparar o perfil de potência e retardo obtido pela simulação com o medido em [23]. A Figura 4.15 mostra esta comparação.

$$h(\tau) = \mathcal{F}^{-1}H(f) \tag{4.5}$$

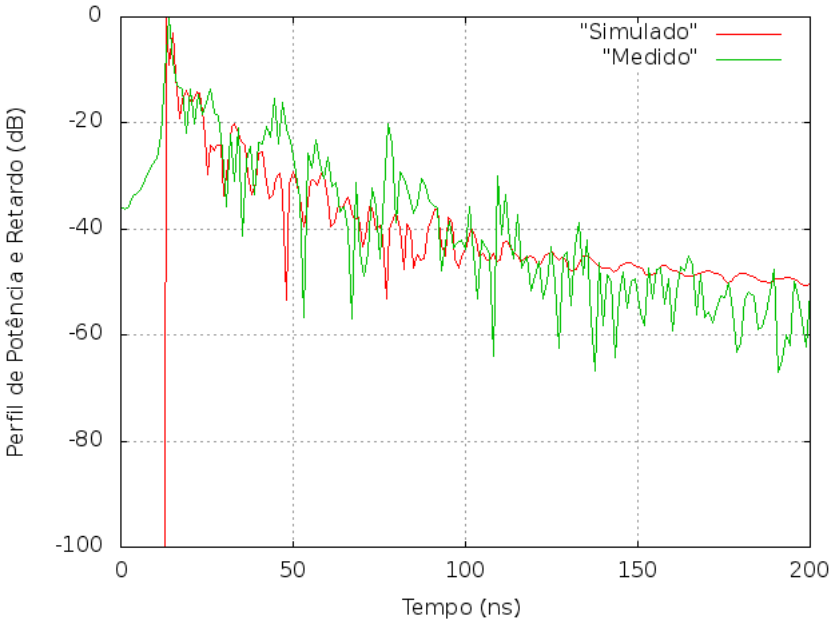


Figura 4.15: Comparação entre medição [23] e simulação referente aos pontos T1 e R.

4.1.2.2 Caso 02

O mesmo processo que foi realizado no caso 1 foi refeito neste caso, diferenciando somente no posicionamento da antena transmissora(agora posicionada em T2). O perfil de potencia e retardo obtido foi o mostrado na Figura 4.16.

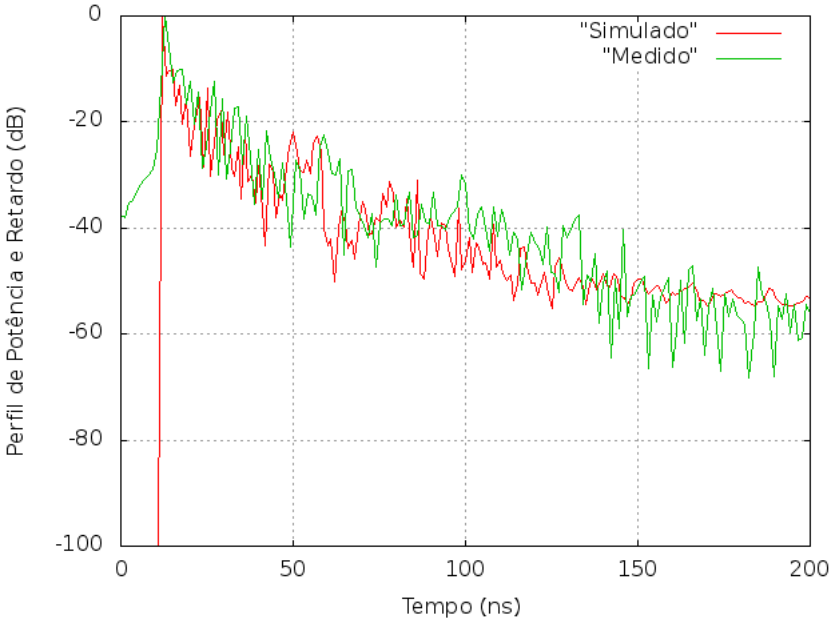


Figura 4.16: Comparação entre medição [23] e simulação referente aos ponto T2 e R.

4.1.2.3 Conclusão

A exelente concordância entre os resultados experimentais e simulados ocorre devido ao fato de método FDTD produzir soluções de onda completa, levando em conta todos os efeitos de refração, reflexão e difração eletromagnética em todos os pontos do domínio do problema e em todos os instantes de tempo. Desta forma, estes resultados validam a implementação da interface produzida neste trabalho. As diferenças principais ocorrem para valores abaixo de -20dB, de forma que pode-se atribuí-las a presença de ruídos e eventuais objetos pŕesentes no ambiente na hora da medição, que não foram inseridos no modelo. Ressalta-se que , mesmo abaixo de -20dB, os níveis de $h(t)$ são compatíveis com os níveis obtidos experimentalmente.

Capítulo 5

Considerações Finais

Ainda estou terminando.

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

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

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

[8] 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.

[9] N. Gebhardt, “Método de criação da malha de um cone,” 2003. [Online]. Available: http://irrlicht.sourceforge.net/docu/classirr_1_1scene_1_1_i_geometry_creator.html#af532c8fb5558cf274181eb81220db85b

[10] UNIOSTE, “Computação gráfica,” 2010, curso de Informática, Centro de Ciências Exatas e Tecnológicas - CCET.

[11] R. Tori, *Realidade Virtual Conceitos e Tendências*, 1st ed., C. Kirner, Ed. J. Garcia Comunicação Visual, 2004.

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

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

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

[15] F. S. C. da Silva, *Introdução a Ciência da Computação com Jogos*. Elsevier Ltda, 2010.

[16] 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.

[17] A. Taflove, *Numerical Solution of Eteady-state Electromagnetic Scattering Problems Using the Time-Dependent Maxwell’s Equations*, 1975.

[18] A. Taflove and S. C. Hangness, *Computacional Electrodynamics, Finite-Discrete Time-Domain Method*, 3rd ed. Artech House Inc., 2005.

[19] 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.

[20] R. S. Pressman, *Engenharia de Software*, 6th ed. Material, 2011.

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

[22] N. Gebhardt, 2003. [Online]. Available: http://irrlicht.sourceforge.net

[23] F. J. B. Barros, “Traçado tridimensional de feixes para a obtenção das características de propagação do canal de banda ultralarga em ambientes interiores,” Pontífica Universidade Católica, 2010, dissertação de Doutorado.

[24] T. K, “Novel method for analyzing the transient behavior of gounding systems based on the finiti-differente time-domain method.” *Power Engineering Review*, pp. 1128–1132, 2001.

[25] J. de Souza Aráujo, “Desenvolvimento de metodologias para localização de intruso em ambientes indoor,” Ph.D. dissertation, Instituto de Tecnologia da Universidade Federal do Pará, 2010.

[26] C. A. Balanes, *Antenna Theory: Analysis and Design*. John Wiley and Sons Publishers Inc, 1997.