



CENTRO UNIVERSITÁRIO RITTER DOS REIS

**CURSO DE BACHARELADO EM SISTEMAS DE
INFORMAÇÃO**



**SISTEMA DE SIMULAÇÃO INTERATIVA DE REGIMES CONSTRUTIVOS E
VOLUMÉTRICOS UTILIZANDO TÉCNICAS DE COMPUTAÇÃO GRÁFICA**

por

Tânia Regina Vieira Roque

Porto Alegre
2007.

Tânia Regina Vieira Roque

SISTEMA DE SIMULAÇÃO INTERATIVA DE REGIMES CONTRUTIVOS E
VOLUMÉTRICOS UTILIZANDO TÉCNICAS DE COMPUTAÇÃO GRÁFICA

Trabalho de Conclusão de
Curso II apresentado à Fa-
culdade de Informática,
como requisito parcial à ob-
tenção do título de Bacharel
em Sistemas de Informação.

Orientador: Prof^ª. Ms. Isabel
Cristina Siqueira da Silva

Porto Alegre
2007.

RESUMO

O presente trabalho tem por objetivo desenvolver uma ferramenta que faz uso de conceitos da Computação Gráfica para simulação gráfica voltada à área da Arquitetura e Urbanismo. A ferramenta permite, ao usuário, a entrada de dados referentes aos parâmetros normativos de regimes construtivos, mais especificamente, os dispositivos de controle das edificações traçados no Plano Diretor de Desenvolvimento Urbano e Ambiental Municipal. A partir de tais parâmetros, é gerada a representação visual tridimensional correspondente à edificação, sobre a qual o usuário pode interagir de modo a servir de apoio à validação do projeto.

Palavras-chave: Computação Gráfica, Regimes Construtivos, Interação Humano-Computador.

ABSTRACT

The present work aims to develop a graphic simulation tool that is use of Computer Graphics concepts for graphic simulation in the Urbanism and Architecture area. The tool allows the user the entrance of data referring to the normative parameter of constructive regimes, more specifically, the volumetric regimes established in the Major Plan of Urban Development and City Environment. From these parameters, the Three-dimensional visual representation of the building is generated, upon which the user can interact in a way to work as support to the project validation.

Key-words: *Computer Graphics, Constructive Regimes, Human Computing Interaction*

LISTA DE ABREVIATURAS

API - *Application Program Interface*

ARB - *Architecture Review Board*

BLAST - *Building Loads Analysis and Thermodynamics*

CAD - *Computer Aided Design*

DWG - *DraWinG*

DXF - *Drawing Interchange File*

FLTK - *Fast Light Toolkit*

FLUID - *Fast Light User Interface Designer*

GCC - *GNU Compiler Collection*

GL - *Graphical library*

GLU - *Graphical Utility library*

GLUT - *GL Utility Toolkit*

GUI - *Graphical User Interface*

LGPL - *Library General Public License*

Modo RGB - *red – green - blue*

OpenGL - *Open Graphic Library*

PDDUA - *Plano Diretor de Desenvolvimento Urbano Ambiental de Porto Alegre*

SMP - *Secretaria Municipal do Planejamento*

TIC - *Tecnologia de Informação e Comunicação*

UML - *Unified Modeling Language*

FIGURAS

Figura 1: Pipeline de visualização 3D.....	16
Figura 2: Exemplo de um objeto representado por uma malha de polígonos que compõem uma cena (adaptado de COHEN,MANSSOUR,2006).....	17
Figura 3: Exemplo de instâncias de um objeto para compor uma cena.....	17
Figura 4: Exemplo da aplicação de recorte.....	18
Figura 5: Exemplo da aplicação de mapeamento.....	18
Figura 6: Mapeamento de textura.....	19
Figura 7: Vista do interior do Hall de uma fábrica (adaptado de GRAZZIOTIN,2003)..	25
Figura 8: Código monolítico (adaptado de Neto, S/D).....	26
Figura 9. Aplicação com duas camadas físicas (adaptado de Neto, S/D).....	27
Figura 10. Aplicação com três camadas físicas (adaptado de Neto, S/D).....	27
Figura 11: Aplicação típica OpenGL.....	30
Figura 12: Primitivas geométricas da OpenGL.....	31
Figura 13 - Versão simplificada do pipeline OpenGL (adaptada de Manssour, 2007)..	32
Figura 14: Transformações geométricas em OpenGL: (a) rotação; (b) translação; (c) escala (Adaptado de SOBRINHO).....	32
Figura 15: Projeções disponibilizadas pela OpenGL: (a) paralela ortográfica; (b) perspectiva(Adaptada de SILVA).....	33
Figura 16: Câmera Sintética (Adaptada de SILVA).....	33
Figura 17. Técnicas de iluminação(Adaptada de COHEN, 2006)	34
Figura 18: Sombra planar (Adaptada de SOBRINHO,2003).....	35
Figura 19: widgets disponíveis pela FLTK.....	36
Figura 20: Fluid.....	36
Figura 21: Janela do ambiente Dev C++.....	38
Figura 22. Diagrama de Casos de Uso.....	49
Figura 23 : Diagrama de Seqüência Alterar ou Gerar Imagem.....	54
Figura 24 : Diagrama de Seqüência para tratamento de eventos.....	55
Figura 25: Diagrama de Atividades Alterar ou Gerar Imagem.....	56
Figura 26: Diagrama de Atividades Alterar Parâmetros.....	57
Figura 27: Diagrama de Atividades Rotação ou Zoom da Imagem.....	58
Figura 28: Diagrama de Classe.....	59
Figura 29: Interface do sistema proposto.....	61
Figura 30. Cálculo da “Área Total” do lote pelo sistema proposto.....	62
Figura 31. Mensagem de advertência para preenchimentos dos “Parâmetros do Lote” .	

.....	63
Figura 32. Imagem da entrada de dados “Índice de Aproveitamento” e a resposta “Potencial Construtivo”.....	64
Figura 33. Preenchimento do campo “Taxa Ocupação” e a resposta gerada no campo “Laje Máxima”.....	65
Figura 34. Mensagem de atenção para o preenchimento dos campos referentes aos “Regimes Construtivos”.....	66
Figura 35. Imagem do acionamento do botão “Atualizar” da Interface do sistema desenvolvido.....	67
Figura 36. Imagem do “Potencial Construtivo Realizado” maior que o “Potencial Construtivo”.....	68
Figura 37. Imagem rotacionada com o botão “Rotação Vertical”.....	69
Figura 38. Imagem rotacionada com o botão “Rotação Horizontal”.....	70
.....	70
Figura 39. Imagem aumentada com o botão “Zoom”.....	71

TABELAS

TABELA 1: Comparação entre características das ferramentas apresentadas e da ferramenta proposta.....	46
TABELA 2: Referências aos Casos de Uso.....	50
TABELA 3: Caso de uso Alterar ou Gerar Imagem do Edifício.....	51
TABELA 4: Caso de uso alterar parâmetros.....	52
TABELA 5: Caso de uso Limpar Parâmetros.....	52
TABELA 6: Caso de uso Rotacionar a Imagem.....	53
TABELA 7: Caso de uso Zoom da Imagem.....	53
TABELA 8: Respostas obtidas junto ao questionário de validação.....	72

SUMÁRIO

RESUMO.....	3
ABSTRACT.....	4
LISTA DE ABREVIATURAS.....	5
FIGURAS.....	6
TABELAS.....	8
INTRODUÇÃO.....	11
1. REFERENCIAL TEÓRICO.....	14
1.1 Computação Gráfica.....	15
1.2 Arquitetura e Urbanismo.....	20
1.2.1 Planos Diretores Municipais.....	21
1.2.2 Simulação Computacional voltada à Arquitetura.....	23
1.3 Padrão MVC	25
2. TECNOLOGIAS UTILIZADAS NA IMPLEMENTAÇÃO DA FERRAMENTA PROPOSTA.....	29
2.1 OpenGL	30
2.2 Ferramenta FLTK.....	35
2.3 Linguagem de Programação C++ e Ambiente de Desenvolvimento Dev C++.....	37
2.4 Notação UML (Unified Modeling Language).....	39
3. FERRAMENTAS PARA MODELAGEM E SIMULAÇÃO ESPACIAL.....	40
3.1 Ferramentas CAD	40
3.1.1 AutoCAD	41
3.1.2 ArchiCAD.....	42
3.1.3 DataCAD.....	42
3.1.4 VectorWorks.....	43
3.1.5 FormZ.....	44
3.2 Blender.....	44
3.3 CityZoom	45
3.4 Radiance.....	46
4. SOLUÇÃO PROPOSTA.....	48
4.1 Visão Geral.....	48
4.2 Modelagem.....	49
4.3 Interface Gráfica e Funcionalidade.....	60
4.4 Validação da Ferramenta.....	71

	10
CONCLUSÃO.....	73
BIBLIOGRAFIA.....	74
ANEXO A – Código que realiza os calculos dos resultados.....	83
ANEXO B – Código realizado pelo evento “Atualizar”.....	86
ANEXO C – Questionário aplicado aos sujeitos avaliadores.....	93

INTRODUÇÃO

Este trabalho tem seu objetivo geral motivado pelo crescente uso da Computação Gráfica no desenvolvimento de sistemas voltados à área da Arquitetura e Urbanismo no que tange projetos de edificações. Tal fato traz como benefícios uma maior facilidade, rapidez e segurança na comunicação entre os indivíduos diretamente atuantes no ciclo de vida da edificação. Cresce o interesse por parte de arquitetos, incorporadores, construtores e proprietários de terra, como também estudantes, pesquisadores e professores por ferramentas computacionais práticas e acessíveis, capazes de permitir que especulações numéricas e morfológicas dos regimes construtivos se dêem rápida e interativamente.

Atualmente, no Brasil, há poucas pesquisas e desenvolvimento tecnológico de meios alternativos e eficientes de visualizar e disponibilizar informações relativas a regimes urbanísticos, principalmente aos regimes construtivos. Pode-se citar a ferramenta CityZoom (GRAZZIOTIN, 2002), desenvolvida junto aos cursos de Arquitetura e Urbanismo e Ciência da Computação da Universidade Federal do Rio Grande do Sul (UFRGS). Portanto, nota-se que há espaço para o desenvolvimento computacional gráfico que vise suprir a necessidade da visualização gráfica e a automatização de processos, muitas vezes, realizados à mão, sobre mapas de papel, e acompanhadas de calculadora.

Outro fator importante e determinístico no desejo da realização deste trabalho se dá pelo aspecto acadêmico, visto que, está em desenvolvimento, na presente Instituição, um projeto de pesquisa voltado à construção de uma solução que permita trabalhar com regimes construtivos através de uma ferramenta computacional gráfica. O projeto é resultado da cooperação entre os cursos de Sistemas de Informação e Arquitetura e Urbanismo da presente

Instituição e, este trabalho de conclusão está sendo desenvolvido como parte de tal pesquisa.

Tendo em vista a crescente necessidade da visualização espacial nos projetos arquitetônicos urbanísticos, embasados em instrumentos legais como os Planos Diretores Municipais, desenvolveu-se uma ferramenta gráfica interativa que, a partir de parâmetros construtivos e volumétricos, valores de metragem de lote e de laje, gera a imagem gráfica do edifício almejado. Permite, assim, que a relação entre as partes envolvidas no projeto arquitetônico se dê com mais eficiência e cooperação, auxiliando o desenvolvimento, a validação e a tomada de decisão por parte do arquiteto.

De modo a atingir o objetivo geral, tem-se os seguintes objetivos específicos:

- Estudo da biblioteca gráfica OpenGL (Open Graphic Library) (OpenGL, 2007), da linguagem de programação C++, através do ambiente DEV C++, e o toolkit FLTK (Fast Light Toolkit) (Tutorial FLTK, 2007);
- Estudar conceitos de desenvolvimento de imagens espaciais 3D;
- Estudar conceitos do paradigma orientação a objetos;
- Estudar a notação UML (Unified Modeling Language) (UML, 2007)

Os capítulos seguintes abordam aspectos relativos aos conceitos que embasam a pesquisa e o desenvolvimento da ferramenta proposta. Tal ferramenta é voltada à área da Arquitetura e Urbanismo baseada em técnicas de Computação Gráfica. Assim, são contemplados aspectos tecnológicos envolvidos no desenvolvimento de tal ferramenta divididos e apresentados como segue:

1) Referencial Teórico

Neste capítulo, são abordados os aspectos relativos aos conceitos que embasam a pesquisa e o desenvolvimento da ferramenta proposta. Tal ferramenta é voltada à área da Arquitetura e Urbanismo baseada em técnicas de Computação Gráfica;

2) Tecnologias Utilizadas na Implementação da Ferramenta Proposta

Este capítulo aborda as informações pesquisadas com relação a biblioteca gráfica OpenGL, a ferramenta FLTK e a linguagem de programação orientada a objetos C++, assim como apresenta informações sobre o ambiente de desenvolvimento DEV C++.

3) Ferramenta Para Modelagem e Simulação Espacial.

Apresenta algumas ferramentas de representação espacial de objetos usadas na área da Arquitetura e Urbanismo. Tais ferramentas empregam técnicas de Computação Gráfica, algumas destas já citadas na seção sobre Referencial Teórico;

4) Solução Proposta

Descreve a funcionalidade do sistema proposto através de sua interface gráfica, bem como a modelagem do mesmo;

1. REFERENCIAL TEÓRICO

Neste capítulo, são contemplados aspectos tecnológicos envolvidos no desenvolvimento de tal ferramenta e, para tanto, inicialmente são apresentados os aspectos relativos a técnicas de Computação Gráfica. Essas são empregadas na geração da imagem resultante da interação do usuário com a ferramenta desenvolvida.

A seguir, descreve-se os aspectos da área da Arquitetura e Urbanismo relativos ao projeto arquitetônico. São discutidas as diretrizes de Planos Diretores Municipais bem como normas e regimes urbanísticos, regimes construtivos e volumétricos necessários à geração da imagem pela a ferramenta proposta. É abordada, também, a importância do uso da CG no projeto arquitetônico no sentido de orientar o trabalho do engenheiro ou arquiteto, reduzindo margens de erro durante a execução do projeto. No mesmo capítulo são, ainda, apresentadas algumas ferramentas de modelagem bem como ferramentas de simulação envolvendo as normas urbanísticas municipais as quais servem como base para este estudo.

Conceitos ligados ao padrão MVC também são abordados, destacando vantagens no uso do mesmo no desenvolvimento de ferramentas computacionais como escalabilidade, eficiência facilidade de manutenção e reutilização do código.

Por fim, são relacionadas as tecnologias que são aplicadas no desenvolvimento do trabalho proposto: biblioteca gráfica OpenGL, demonstrando suas particularidades e funcionalidades na geração de aplicações gráficas interativas; toolkit FLTK, utilizado na elaboração da interface gráfica da ferramenta proposta; linguagem de programação C++, bem como o ambiente de desenvolvimento Dev C++ utilizados na definição das

classes do programa; e a notação UML sua importância como forma de organizar e orientar o desenvolvimento do software.

1.1 Computação Gráfica

A CG é a área da Computação que estuda o processo da síntese da imagem por computador. Gera a imagem de acordo com uma descrição direta ou indireta do usuário. Permitindo assim, uma mais fácil comunicação e interação humano-computador.

De acordo com Azevedo e Conci (2003), CG trata das técnicas e de métodos computacionais que convertem dados em imagem. Permite a manipulação desta imagem através de dispositivos gráficos.

Para Foley (1990), CG é a criação, o armazenamento e a manipulação de modelos de objetos e suas imagens pelo computador. Busca dominar, assim, o conhecimento necessário para implementar ambientes gráficos básicos, tanto para o desenvolvimento de aplicações como para o estudo e a pesquisa no campo da modelagem geométrica.

O grande número de aplicações das técnicas de CG a coloca, algumas vezes, em um posicionamento de similaridade em relação a áreas próximas que podem ser confundidas (COHEN, 2006). Neste sentido, segundo Gomes e Velho (1998), pode-se definir três grandes áreas que se inter-relacionam: Visualização, onde a imagem é gerada através de um modelo matemático que contém os elementos gráficos básicos; Processamento de Imagens (PI), que busca o realismo da imagem digital, tentando torná-la mais acessível à percepção humana; e Visão Computacional (VC), que se baseia na obtenção da descrição da imagem digital comumente por um conversor analógico/digital.

Em CG, modelos são usados para representar entidades e fenômenos do mundo físico real no computador através da imagem. Para tanto, o *pipeline* (estágios de processos) apresentado na Figura 1 é aplicado.

Pipeline é a forma ou disposição de etapas de elaboração da imagem computacional com o intuito de ganhar rapidez no processamento desta. Existem várias categorias ou métodos de construção de modelos bidimensionais (2D) e tridimensionais (3D), cada um com vantagens e desvantagens, adaptando-se melhor para uma ou outra aplicação.

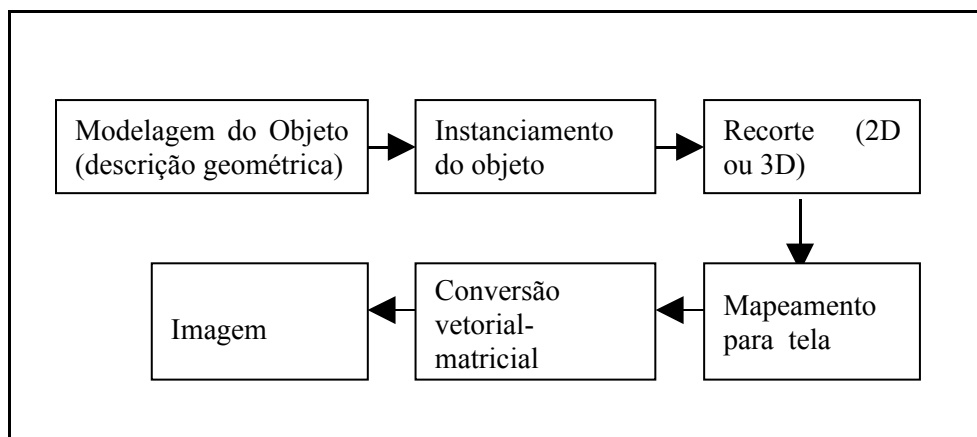


Figura 1: Pipeline de visualização 3D.

A forma mais comum de representar modelos 3D é através de uma malha de polígonos. Para tanto, define-se um conjunto de vértices no espaço (geometria) e como esses vértices devem ser ligados para formarem polígonos fechados ou faces (topologia) (COHEN, 2006). (Ver Figura 2)

Logo, a modelagem do objeto consiste no processo de descrever um modelo, objeto ou cena, de forma que se possa desenhá-lo. A modelagem engloba dois tópicos de estudo: formas de representação dos objetos, que se preocupa com a forma (ou estruturas de dados) como os modelos são armazenados; e técnicas de modelagem dos objetos, que trata das técnicas interativas (e também das interfaces) que podem ser usadas para criar um modelo de objeto. Portanto, considera-se que uma imagem consiste em uma matriz de pontos e um modelo é uma representação computacional de um objeto.

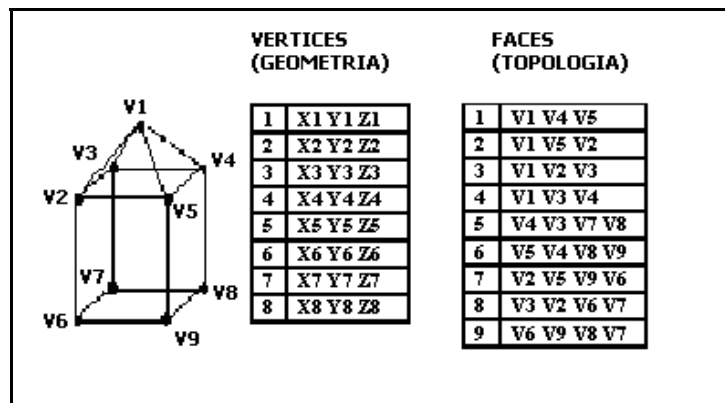


Figura 2: Exemplo de um objeto representado por uma malha de polígonos que compõem uma cena (adaptado de COHEN,MANSSOUR,2006).

Uma das funcionalidades mais importantes das aplicações gráficas é a possibilidade de manipular e alterar interativamente as características dos objetos através de operações matemáticas realizadas sobre vértices, mudando, uniformemente, o aspecto de um modelo já armazenado no computador. As alterações não afetam a estrutura do desenho, mas sim o aspecto que ele vai assumir. Tais operações são realizadas através de transformações geométricas que permitem que um objeto modelado possa ser instanciado (ver Figura 3).

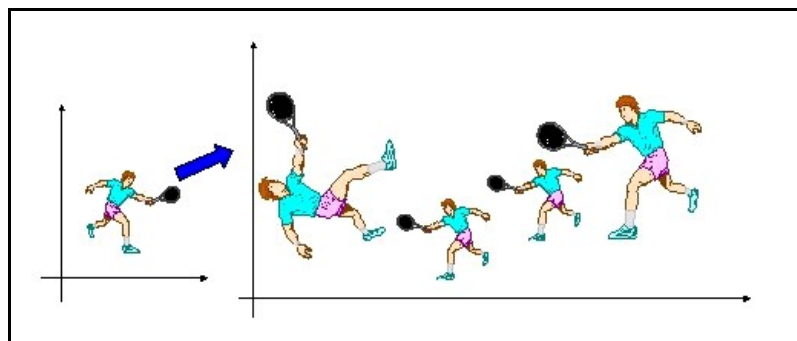


Figura 3: Exemplo de instâncias de um objeto para compor uma cena.

Os tipos fundamentais de transformações geométricas a serem aplicadas a um objeto são a translação, a rotação e a escala.

A translação é usada para definir a posição de um objeto ou cena. Matematicamente, esta operação consiste em adicionar constantes de deslocamento a todos os vértices, ou seja, trocando o objeto ou cena de lugar.

A escala, por sua vez, serve para exibir o objeto ou cena com tamanho modificado. Esta operação consiste em multiplicar um valor de escala por todos os vértices do objeto (ou conjunto de objetos) alterando o tamanho do objeto.

Por fim, a rotação é usada para rotacionar um objeto ou cena em torno de um ou mais eixos de um sistema de coordenadas cartesianas. Consiste em aplicar cálculos utilizando o seno e cosseno do ângulo de rotação a todas as coordenadas dos vértices que compõem o objeto ou cena.

Uma vez que o objeto foi instanciado, parte-se para a etapa de recorte, a qual permite que seja definida qual a região do desenho será exibida (Figura 4).

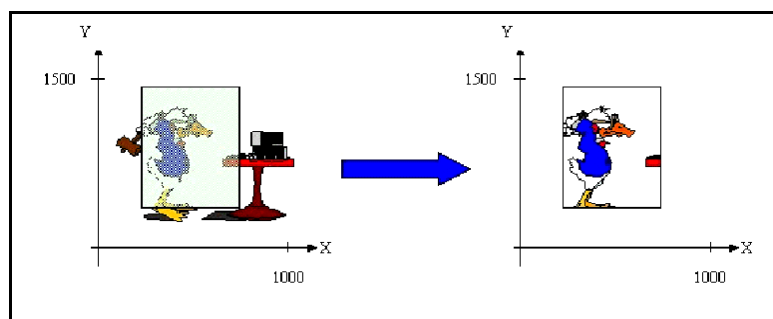


Figura 4: Exemplo da aplicação de recorte.

Com a cena recortada, aplica-se a etapa do mapeamento das coordenadas do objeto definidas no sistema de referência do universo para o sistema de referência da tela (Figura 5).

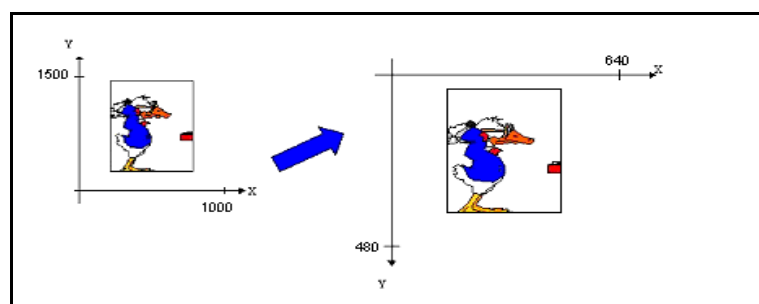


Figura 5: Exemplo da aplicação de mapeamento.

Por fim, tem-se a etapa da conversão vetorial-matricial (rasterização), onde o objeto terá seus vértices e faces convertidos para *pixels*, formando a imagem a ser exibida no dispositivo gráfico.

As técnicas explicadas permitem a geração de imagens simples, cuja aparência é artificial. Para que essas imagens tenham mais realismo, é preciso considerar outros aspectos. Dois desses aspectos são: a iluminação e o mapeamento de texturas.

A luz é necessária para que os objetos possam refletir e absorver seus raios, permitindo que o olho humano perceba diferentes cores e tonalidades. Mesmo através do recurso de iluminação, freqüentemente os objetos não terão ainda uma aparência realística. Isso acontece porque os materiais na realidade não têm simplesmente cores diferentes, mas sim uma série de características físicas, como rugosidade e textura, que determinam como exatamente refletem os raios de luz.

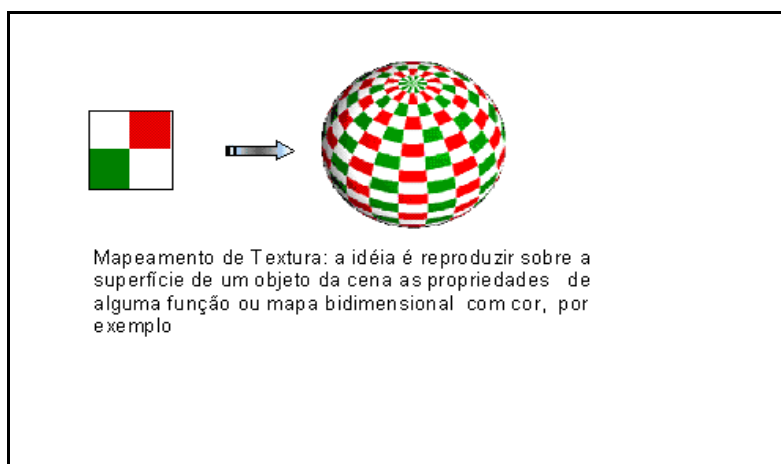


Figura 6: Mapeamento de textura.

A textura funciona então, como um decalque ou um papel de parede, sendo “colada” à superfície (Figura 6). É importante observar que o mapeamento de textura pode, e geralmente é, combinado com o processo de iluminação, gerando um resultado ainda melhor (COHEN, 2006).

A CG oferece a simulação de diversos materiais e de propriedades geométricas dos objetos e, por isso, possui várias áreas de aplicação, a saber: Arquitetura, Engenharia, Meteorologia, Medicina, Cinema, Jogos, Indústria,

Educação entre outras que necessitam representar e manipular objetos gráficos, dos quais serão extraídas ou geradas informações.

1.2 Arquitetura e Urbanismo

A área da Arquitetura e Urbanismo constitui a arte, a ciência e a técnica de planejar, projetar e recuperar equipamentos, espaços e construções para o desempenho de um amplo leque de atividades, incluindo artes gráficas, edifícios, paisagens e cidades. Arquitetura, segundo Corona e Lemos (1972), é:

A arte de compor e construir toda a sorte de edifícios, segundo as regras e proporções convenientes (...) é construção concebida com a intenção de ordenar plasticamente o espaço, em função de uma determinada época, de um determinado meio, de uma determinada técnica e de um determinado programa.” e “o conceito de urbanismo, originariamente, se refere à arte e à técnica do arranjo das cidades.

A Arquitetura e Urbanismo abrange o conhecimento tecnológico necessário à materialização das idéias de projeto para os espaços arquitetônicos e urbanos, compreendendo os aspectos da tecnologia da construção, dos sistemas estruturais e do controle ambiental e infra-estrutura urbana. Neste contexto, a estratégia de qualificação do território da cidade implica uma estratégia espacial e uma estratégia social, que devem ser articuladas no projeto e nos planos.

Muitas cidades, como a maioria dos municípios de porte médio do Brasil, apresentaram, nos últimos vinte anos, um crescimento urbano acelerado, ocasionando efeitos e impactos urbanísticos na qualidade dos tecidos urbanos, particularmente, dos espaços públicos. Em decorrência disto, o planejamento urbano tem tido um maior destaque visto que existe tal transformação do uso

do solo nas áreas centrais, de lotes residenciais para lotes comerciais e de serviços, bem como de edificações residenciais multifamiliares estimulando, assim, a elaboração de Planos Diretores e o planejamento do território das cidades como um todo.

1.2.1 Planos Diretores Municipais

Para que a cidade seja um local de vida em comunidade, é necessário que ela seja bem administrada e organizada, incluindo a ocupação e o uso de seu solo.

O Plano Diretor é uma lei municipal obrigatória para os municípios com cidade de população superior a 20.000 habitantes. Este deve ser o instrumento básico da política municipal de desenvolvimento e expansão urbana, a qual tem como objetivo ordenar o pleno desenvolvimento das funções sociais da cidade e garantir o bem-estar de seus habitantes (Constituição Federal e Estatuto da Cidade). Pode-se citar, como exemplo, o Plano Diretor de Desenvolvimento Urbano Ambiental de Porto Alegre – PDDUA que, conforme VARGAS (2003), está fundamentalmente baseado na questão da densidade.

Esta medida originária da geografia e que mensura a relação entre quantidade de pessoas e quantidade de espaço foi a variável-chave utilizada para determinar os regimes urbanísticos e orientar o desenvolvimento urbano conforme expresso nos textos e capítulos que conformam o arcabouço conceitual do Plano Diretor de Porto Alegre.

A cidade de Porto Alegre tem tradição em planejamento urbano, constituindo-se na primeira capital do país a contar com um PDDUA. O plano regulador existente no PADDUA de Porto Alegre (SMP-Secretaria Municipal do Planejamento) diz:

- Art. 93: Plano Regulador é o instrumento que define os dispositivos que regulam a paisagem da cidade, edificada ou não. Parágrafo único. O uso e a ocupação do solo no território de Porto Alegre serão disciplinados através do regime urbanístico, do traçado do PDDUA e acompanhados através de monitoramento.

Existem normas para construir, instalar uma atividade (comércio, serviços, moradias ou indústrias) e dividir terrenos na cidade. Plano Regulador é a parte do Plano Diretor que estabelece as formas e condições que devem ser atendidas. O conjunto destas normas se chama regime urbanístico. Uma cidade dinâmica, que cresce e se modifica todos os dias, deve ter um acompanhamento constante (monitoramento) para que suas normas estejam sempre atualizadas e ajustadas às necessidades das pessoas. Também é fundamental verificar se o crescimento da cidade está ligado a uma melhoria da qualidade de vida (SMP).

- Art. 94: Regimes urbanísticos são as normas ou regras relativas à densificação, que é a quantidade de pessoas que vão morar ou trabalhar em cada parte da cidade; às atividades, ou seja, quais os negócios que poderão ser instalados; aos Dispositivos de Controle das Edificações, que definem o tamanho e a forma que os prédios podem ter, e ao Parcelamento do Solo, que define o tamanho dos lotes e os espaços ocupados pelas futuras ruas, praças e escolas. Os regimes urbanísticos ou instrumentos urbanísticos são parâmetros específicos de uma determinada zona da cidade. Qualquer prédio, da casa ao edifício, deve respeitar as regras de construção previstas no Plano Diretor.

Os dispositivos de controle das edificações, os regimes construtivos e volumétricos, tratam do tamanho e da forma que os edifícios poderão ter. Eles são um dos meios usados para controlar o crescimento da cidade porque definem, zona a zona, qual o tamanho e a altura das construções, como as alturas máximas, que variam de 9,00 (o equivalente a três andares) a 52

metros (cerca de 17 pavimentos), A taxa de ocupação (TO) define o percentual das áreas que podem ser ocupadas e as que devem ficar livres no terreno. Os recuos de frente, lateral e de fundos, que não poderão ser inferiores a 18% (dezoito por cento) da altura da edificação, garantido um mínimo de 3m (três metros), aplicados a partir da base da edificação; os recuos para ajardinamento terão dimensão mínima de 4m (quatro metros), dependendo da zona da cidade, bem como a questão dos estacionamentos. Respeitando estas normas cada rua adquire uma característica própria que, no conjunto, acaba dando um "desenho" à cidade.

Estes regimes são de suma importância para as pessoas voltadas diretamente à Arquitetura e Urbanismo e são a base do estudo e desenvolvimento da ferramenta desenvolvida.

A Arquitetura, além de observar os dispositivos legais estabelecidos, utiliza no processo de desenvolvimento do projeto arquitetônico e urbanístico, as ferramentas gráficas como metodologia no planejamento arquitetônico e como referência para sugestões de verificações dos Planos Diretores Municipais

1.2.2 Simulação Computacional voltada à Arquitetura

O projeto arquitetônico tem sua representação gráfica através do desenho arquitetônico que corresponde a um conjunto de normas internacionais sob a supervisão da ISO.

O desenho arquitetônico, nas etapas mais comuns do projeto como estudo preliminar, anteprojeto, projeto legal e projeto executivo, pode variar normalmente na complexidade e na quantidade de informação. De acordo com normas técnicas, pode ser feito à mão ou auxiliado por computador através de ferramentas gráficas como, por exemplo, sistemas CAD (*computer aided design*).

O grande avanço metodológico e qualitativo do sistema de planejamento e gerenciamento de informações se dá pela elaboração do desenho

arquitetônico na forma tridimensional (3D), através do uso de recursos e aplicativos no tratamento e simulações de cenários urbanos (PENEAU, 1990).

Por isso, atualmente, o trabalho de investigação do espaço urbano e arquitetônico requer a exploração das potencialidades da Computação Gráfica (CG) e de novas tecnologias da informação tendo em vista o processo decisional de planejamento e desenho urbano.

O tratamento das informações e entidades espaciais representa um grande passo para atividades de planejamento urbano. Papa (2006) afirma:

Aplicada à área da Arquitetura, a CG tem como um dos seus objetivos intermediar a relação arquiteto-cliente, facilitando assim a percepção espacial do projeto elaborado, bem como o efeito da obra no entorno construído.

No âmbito de tecnologias de CG aplicada, muitos sistemas firmaram-se como ferramentas de representação na escala arquitetônica, customizando tarefas, modelando tridimensionalmente desenhos de paredes, escadas, telhados ou passando a limpo desenhos técnicos (pranchetas eletrônicas). Como exemplo de tais sistemas pode-se citar: CAD, AutoCAD (AUTODESK, 2007); VectorWorks (VECTORPRO, 2007); FormZ (FORMZ, 2007); ArchiCAD (GRAPHISOFT, 2007); DataCAD (DATACAD, 2007); Rhino3D (RHINO3D, 2007);

Hearne (1997) afirma que o uso principal da CG está na elaboração de projetos, particularmente para a Engenharia e Arquitetura:

Os produtos projetados são quase todos consultados em ferramentas como a CAD (Computer Aided Design) que são usadas rotineiramente em projetos de edificação, de automóveis, aeronaves, embarcações e naves espaciais, dos computadores, dos textos e de muitos outros produtos.

Questões como o acesso ao sol, a disponibilidade de luz natural e elétrica, o conforto térmico, a necessidade de ventilação, entre outras também

têm sido alvo de ferramentas computacionais que visam auxiliar planejadores e arquitetos. Por exemplo, a ferramenta Radiance (RAD,1997) (Figura 7), que será tratada mais adiante, é voltada para análise e visualização de iluminação em projetos de edificações.

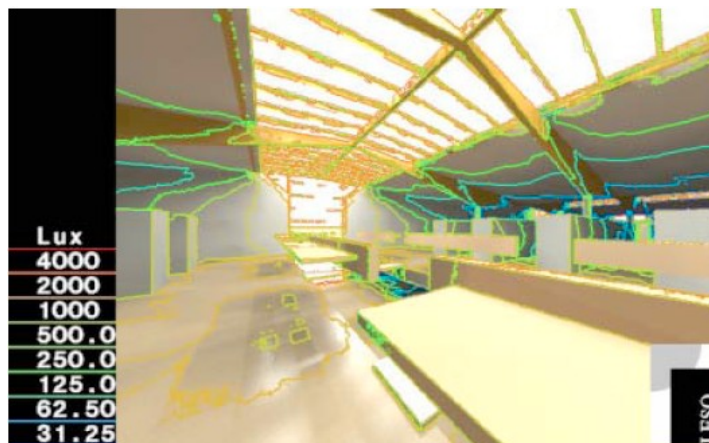


Figura 7: Vista do interior do Hall de uma fábrica (adaptado de GRAZZIOTIN,2003).

No entanto, em relação a normativas urbanísticas (regras de construção da cidade), não constam na literatura muitas ferramentas capazes de realizar simulações. Pode-se citar, neste sentido, a ferramenta CityZom (GRAZZIOTIN, 2004), que será abordada mais adiante neste trabalho.

Nota-se, assim, que a proposição de novos ambientes de simulação de edificações com base em normativas urbanísticas seria de grande valia aos projetistas, construtores, conselheiros municipais, ou qualquer pessoa interessada no assunto.

1.3 Padrão MVC

O grande desafio dos desenvolvedores de software é produzir aplicativos seguros, eficientes, de fácil manutenção, reutilizáveis e em prazos cada vez menores. Neste sentido, optou-se por basear a ferramenta proposta no padrão MVC (*Model – View – Controler* ou modelo - visão - controle), pois a organização de um software em camadas possibilita a independência entre os

componentes propiciando eficiência, escalabilidade, reutilização e facilidade de manutenção do sistema (MACORATTI, 2007).

A arquitetura MVC não é nova e foi originalmente desenvolvida para mapear as tarefas tradicionais de entrada, processamento e saída para o modelo de interação com o usuário. Foi desenvolvida, assim, para projetos com arquitetura de software de aplicação em 3 camadas.

Inicialmente, os sistemas eram desenvolvidos em uma única camada (monolítica), onde toda a lógica de programação estava em um único bloco gerando muitas linhas de código e uma manutenção nada fácil (Neto, S/D) (ver Figura 10). Surgiram, então, as aplicações em duas camadas, onde a lógica de acesso aos dados estava separada das lógicas de negócio e de apresentação (ver Figura 8).

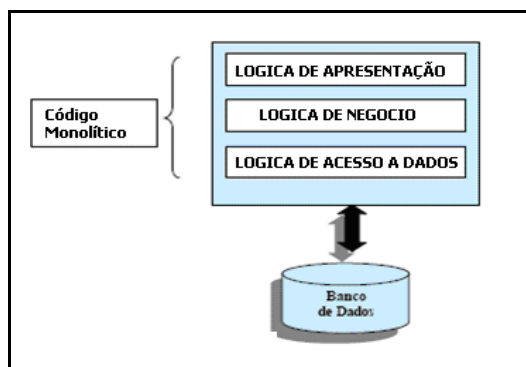


Figura 8: Código monolítico (adaptado de Neto, S/D).

Com o passar do tempo e com o aumento do volume de dados a ser manipulado, a estrutura teve que ser alterada para 3 camadas. A idéia é que o usuário acesse as mesmas aplicações sem ter que instalá-las em sua máquina local (ver Figura 9).

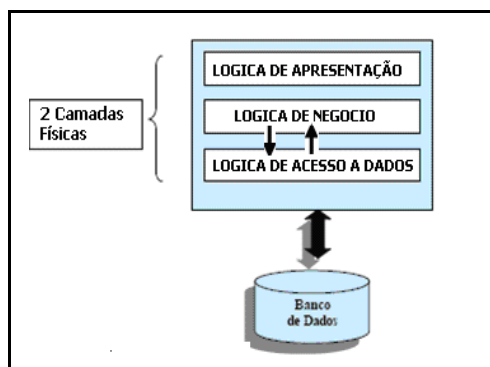


Figura 9. Aplicação com duas camadas físicas (adaptado de Neto, S/D).

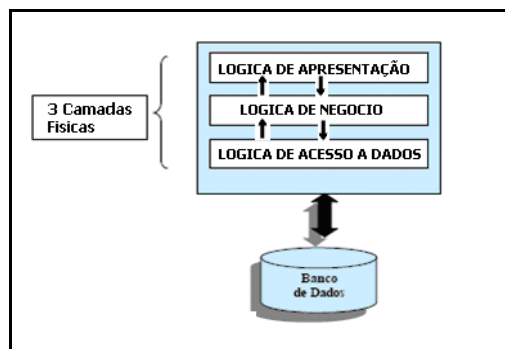


Figura 10. Aplicação com três camadas físicas (adaptado de Neto, S/D).

- Nesse modelo, o sistema se divide em camadas de tal modo que a camada em que o usuário manipula os dados não é a mesma camada em que os dados são realmente processados.
- A camada de apresentação ou, simplesmente, a visão não considera o processo de obtenção da informação, apenas a exibição da mesma.

Já a camada de lógica da aplicação, ou o modelo, é o “coração” da aplicação, responsável pelas seguintes tarefas:

- modela os dados e o comportamento por trás do processo de negócios;
- preocupa-se apenas com o armazenamento, manipulação e geração de dados;

- encapsula dados e comportamento, independente da apresentação.

Por fim, a camada de controle determina o fluxo da apresentação, servindo como uma camada intermediária entre as camadas visão e modelo, controlando e mapeando as ações.

O MVC permite que futuras melhorias e revisões na interface, e até mesmo na capacidade da aplicação, possam ser efetuadas sem muitas dificuldades e restringindo-se a cada camada.

Esta técnica é de grande valia no projeto proposto, já que a ferramenta implementada oferece a possibilidade de interação do usuário com a representação gráfica gerada (interface) de modo a servir de apoio à validação.

2. TECNOLOGIAS UTILIZADAS NA IMPLEMENTAÇÃO DA FERRAMENTA PROPOSTA

A biblioteca gráfico mais bem sucedido no mercado é OpenGL¹ (*Open Graphics Library*), um produto aberto que executa em praticamente todas as plataformas. Assim, a biblioteca OpenGL está sendo utilizada no desenvolvimento deste trabalho para a geração das imagens de acordo com as definições realizadas pelo usuário relativas a normas de regimes urbanísticos usadas na área da Arquitetura e Urbanismo. Através de eventos, seja pelo mouse ou teclado, na interface gráfica da ferramenta permitirá ao usuário a navegação interativa por um cenário tridimensional correspondente a quadra de uma cidade sobre a qual se está trabalhando.

As classes que tratam dos eventos de interface e dos cálculos que permitirão a geração da imagem são construídas através da linguagem de programação C++, através do ambiente Dev C++². Já a interface gráfica está sendo desenvolvida com o uso do toolkit FLTK³.

Além disso, para a modelagem da ferramenta, é empregada a notação UML (*Unified Modeling Language*), de modo a guiar o processo de desenvolvimento do sistema proposto. A seguir, tais tecnologias serão brevemente descritas.

A seguir, cada uma destas tecnologias são abordadas com mais detalhes.

¹ www.opengl.org

² www.bloodshed.net/devcpp.html

³ www.fltk.org

2.1 OpenGL

OpenGL (*Open Graphics Library*) é uma biblioteca portátil, de rotinas gráficas e de modelagem bi (2D) e tridimensional (3D) (OpenGL, 2007). Ela permite desenvolver aplicações interativas e gerar imagens de cenas 3D com um alto grau de realismo, incluindo o desenho de primitivas gráficas, mapeamento de textura e outros efeitos especiais.

A biblioteca OpenGL foi introduzida em 1992, pela *Silicon Graphics*, no intuito de conceber uma API (*Application Program Interface*) gráfica independente de dispositivos de exibição (ver Figura 11). Com isto, era estabelecida uma ponte entre o processo de modelagem geométrica de objetos, situadas em um nível de abstração mais elevado, e as rotinas de exibição e de processamento de imagens, implementadas em dispositivos (*hardware*) e sistemas operacionais específicos.

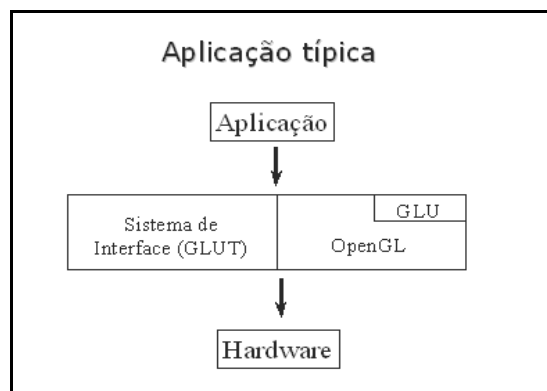


Figura 11: Aplicação típica OpenGL.

As aplicações OpenGL variam de ferramentas CAD (*Computer Aided Design*) a poderosos programas de modelagem. Além do desenho de primitivas gráficas, tais como linhas e polígonos, a OpenGL possui funções que permitem empregar técnicas de iluminação, colorização, definição de materiais, mapeamento de textura, transparência, animação entre vários outros efeitos

especiais. Atualmente, a OpenGL é reconhecida e aceita como um padrão de API para o desenvolvimento de aplicações gráficas 3D interativas e que geram imagens em tempo real (MANSSOUR, 2007).

OpenGL é uma interface para aplicações gráficas que não possui rotinas de alto nível de abstração. Sendo assim, as primitivas geométricas são construídas a partir de seus vértices (SOBRINHO, 2003). Um vértice é representado em coordenadas homogêneas (x, y, z) .

Estão disponíveis 10 tipos de primitivas distintas, porém com a devida organização destas primitivas é possível a criação de estruturas mais complexas (Figura 12).

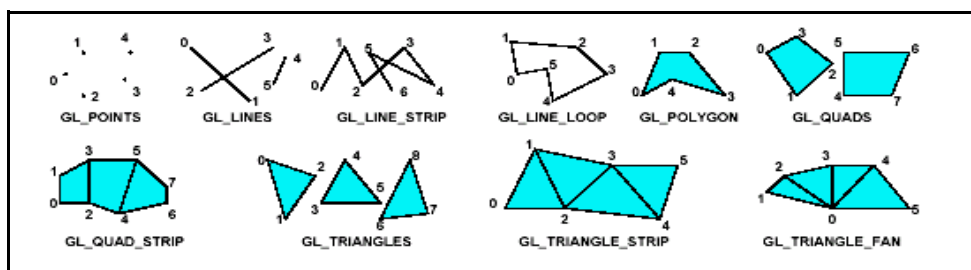


Figura 12: Primitivas geométricas da OpenGL.

A maior parte das implementações do OpenGL tem uma ordem de operações a serem executadas. Uma série de estágios de processos chamam o “*pipeline*” de renderização do OpenGL (SOBRINHO, 2003).

As implementações do OpenGL geralmente provêm bibliotecas auxiliares, tais como a GLU (*Graphical Utility library*), utilizada para realizar tarefas comuns, tais como manipulação de matrizes, geração de superfícies e construção de objetos por composição, assim como a GL (*Graphical library*) utilizada para controle de atributos, primitivas geométricas e gráficas e a GLUT (*GL Utility Toolkit*) que implementa um sistema de janelas simples, é um *toolkit* independente de plataforma, que inclui alguns elementos GUI (*Graphical User Interface*), tais como menus *pop-up* e suporte para *joystick*. O seu principal objetivo é esconder a complexidade das APIs dos diferentes sistemas de janelas (FREITAS, 2003).

Neste trabalho é utilizada a biblioteca GLUT (OpenGL ToolKit) para gerenciamento de janelas.

As bibliotecas GLU e GLUT possuem uma série de funções para desenhar primitivas 3D, tais como esferas, cones, cilindros e *teapot*, além de permitir a criação de outras formas através do conjunto de primitivas disponíveis no OpenGL.

A maior parte das implementações do OpenGL tem uma ordem de operações a serem executadas (tal como no pipeline de CG apresentado na Figura 1). Uma série de estágios de processos chamam o “*pipeline*” de renderização do OpenGL (ver Figura 13) (SOBRINHO, 2003). Como a aplicação faz chamadas a várias funções da API OpenGL, os comandos são armazenados em uma memória específica (buffer de comandos).



Figura 13 - Versão simplificada do pipeline OpenGL (adaptada de Manssour, 2007.)

No processo de transformação geométrica, três operações são possíveis em OpenGL: rotação, translação e escala (ver Figura 14).

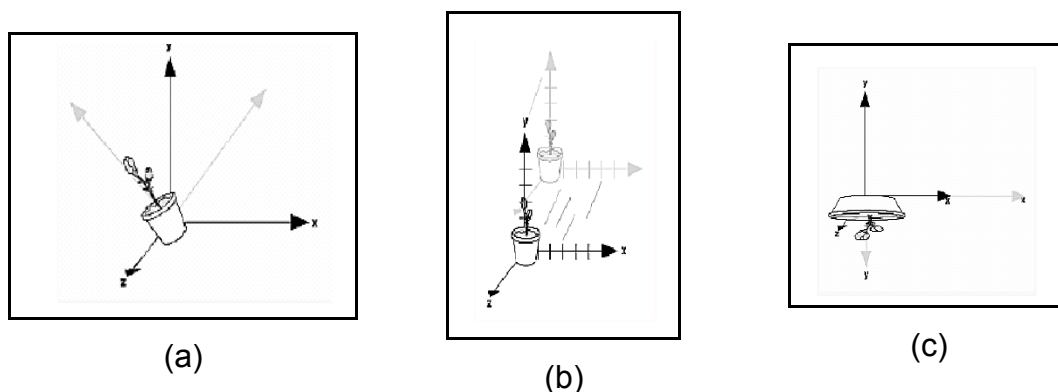


Figura 14: Transformações geométricas em OpenGL: (a) rotação; (b) translação; (c) escala (Adaptado de SOBRINHO).

Em OpenGL, dois tipos de projeções da imagem são possíveis: projeção paralela ortográfica, na qual as projetantes são paralelas entre si, passam pelos vértices dos objetos e interseccionam o plano com um ângulo de 90° (Figura 15 (a)); e projeção perspectiva, quando as projetantes emanam de um único ponto que está a uma distância finita do plano de projeção e passam pelos vértices (Figura 15 (b)) (SILVA, 2007).

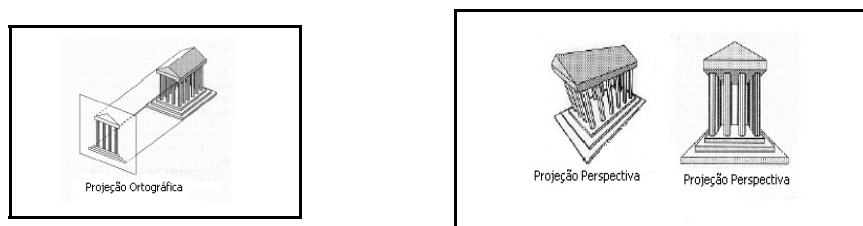


Figura 15: Projeções disponibilizadas pela OpenGL: (a) paralela ortográfica; (b) perspectiva (Adaptada de SILVA).

Em OpenGL também é possível trabalhar com cores de dois modos distintos: modo RGB (*red – green - blue*) e modo índice de cores.

Para a produção de uma determinada cena com OpenGL, deve-se definir, além da projeção, a câmera sintética cuja função é análoga à de uma máquina fotográfica real (WOO 1999). Ver Figura 16.

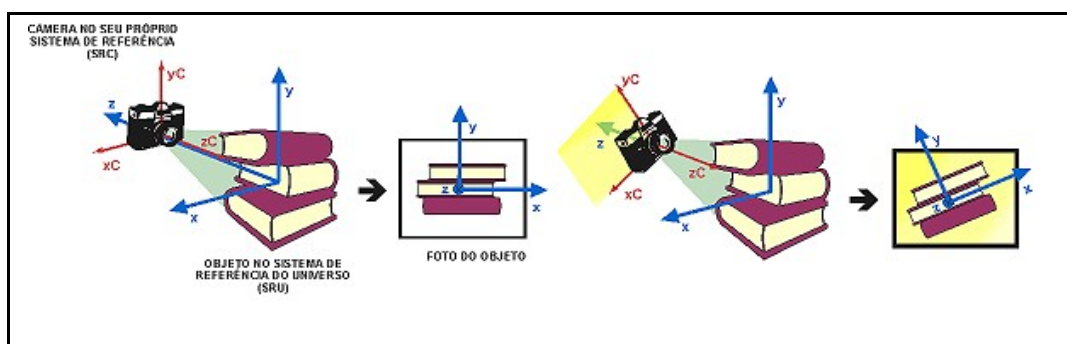


Figura 16: Câmera Sintética (Adaptada de SILVA).

Outra técnica muito utilizada e importante em OpenGL é a iluminação, que permite a inserção de maior realismo à cena. Três componentes estão

disponíveis na biblioteca: luz ambiente, luz difusa, luz especular e quantidade (ver Figura 17) (SOBRINHO, 2003). A luz ambiente ilumina o ambiente uniformemente, vindo de todas as direções. Já a luz difusa vem de uma única direção e é difundida igualmente em todas as direções. A luz especular, por sua vez, vem de uma única direção e tende a ser refletida, também, em uma única direção.

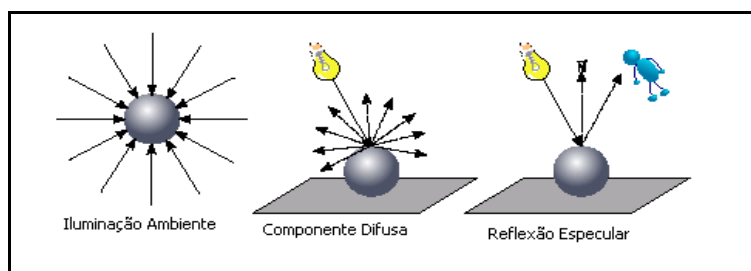


Figura 17. Técnicas de iluminação(Adaptada de COHEN, 2006)

Além das três componentes citadas, uma quarta componente participa do cálculo de iluminação de uma cena em OpenGL – a emissão de luz (*shininess*) - que simula a luz que se origina de um objeto.

Além da iluminação, deve-se definir as propriedades dos materiais a fim de compor a aparência final de um objeto (DAVE SHREINER, 1999) e o modo de sombreado, responsável pelo cálculo das componentes de luz em relação às faces dos objetos.

Por fim, pode-se definir, ainda, as sombras geradas pelos objetos da cena. “Uma sombra é produzida quando uma fonte de luz incide sobre um objeto, este, então, retém tal luz de forma que ela não incida sobre um outro objeto qualquer ou superfície.” (SOBRINHO,2003). As sombras podem ser aplicadas a superfícies planas tipo paredes e chão. A “cor” da sombra é uma versão comprimida das cores do objeto que gera a sombra e não da superfície. O volume da sombra nada mais é do que o volume espacial gerado pela silhueta do objeto. Ver Figura 18.

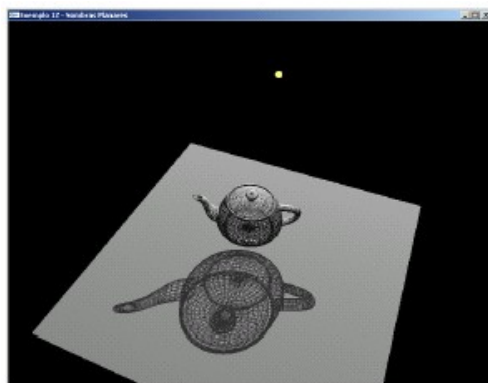


Figura 18: Sombra planar (Adaptada de SOBRINHO,2003).

Apesar de abordar muitos assuntos referentes à CG e à biblioteca OpenGL, o presente estudo não abrange todos os tópicos que a API fornece. Estes podem ser observados no site oficial (OpenGL, 2007) para maiores informações.

2.2 Ferramenta FLTK

Além da biblioteca OpenGL, está sendo utilizada a ferramenta FLTK (*Fast Light Toolkit*) (*FLTK 1.1.7 Programming Manual*) para a implementação da interface gráfica da ferramenta proposta. Através da interface, o usuário pode definir parâmetros para a geração da imagem bem como interagir com esta solicitando serviços.

O FLTK é mantida, atualmente, por um pequeno grupo de colaboradores de diversas partes do mundo com um repositório central nos Estados Unidos. É um *toolkit* para rápida criação de interfaces gráficas voltada ao desenvolvimento de interfaces gráficas, disponível para diferentes sistemas operacionais e podendo ser integrada a algumas linguagens de programação (Tutorial FLTK, 2006). Além disso, FLTK é uma ferramenta livre, portátil tanto em tamanho de código como em desempenho independente de interface, facilitando tanto a extensão do sistema com a alteração da interface.

O conjunto de elementos (*widgets*) disponível é amplo e inclui desde os mais simples como botões, caixas de textos, etiquetas, barras de rolagem,

entre outros, até os menos usuais como, por exemplo, *dials*, navegadores e *timers*, além do fato de que novos elementos são facilmente construídos/adaptados através de derivações dos pré-existentes (ver Figura 19). A FLTK suporta gráficos 3D via OpenGL e provê emulação à biblioteca GLUT(*FLTK 1.1.7 Programing Manual*).

A janela principal da aplicação consiste em uma instância da classe `UserInterface`, que contém objetos do FLTK.

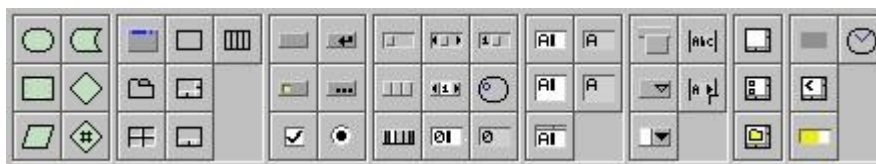


Figura 19: widgets disponíveis pela FLTK.

Adicionalmente, é fornecida com o FLTK a ferramenta FLUID (*Fast Light User Interface Designer*), que permite de maneira fácil e visual a construção de todo o esqueleto da interface gráfica. Através do FLUID, gera-se, automaticamente, o código fonte da interface FLTK na linguagem c++ (Ver Figura 20).

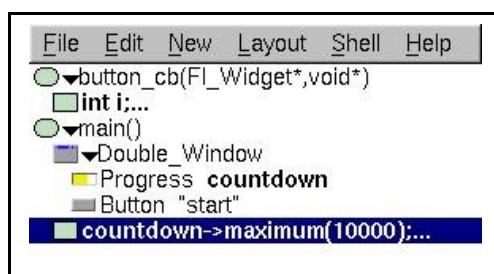


Figura 20: Fluid

O FLUID armazena o arranjo da interface em arquivos de extensão `.fl`, que podem ser editados para ajustes finos nos elementos gráficos (*FLTK 1.1.7 Programing Manual*).

2.3 Linguagem de Programação C++ e Ambiente de Desenvolvimento Dev C++

Além da biblioteca OpenGL, usada para a geração da imagem, e do *toolkit* FLTK, empregada na geração da interface gráfica do sistema proposto, está sendo utilizada a linguagem C++ (abordada a seguir) para a definição das demais classes do sistema proposto.

A linguagem de programação C++ é, na verdade um superconjunto da linguagem de programação C (seu primeiro nome foi “C com Classes”). A razão da linguagem C ter sido chamada de C é simplesmente porque ela foi sucessora de uma linguagem chamada B, desenvolvida por Ken Thompson em 1970 e que rodava em um DEC PDP-7, um computador muito menos potente do que um PC moderno (MACIEL,2004).

C++ é uma versão realçada do C e inclui todas as características desta adicionando a sustentação para a programação orientada do objeto, possuindo, assim, características como encapsulamento, ligações de controle de linha, sobrecarga de operadores, herança e polimorfismo (ARNAUT, 2007).

Segundo Barreto (2005), o objetivo do paradigma da programação orientada a objeto é modelar um objeto de software da forma mais semelhante aos objetos da vida real. Para tanto, define o objeto em termos de suas características.

Neste contexto, o objeto é uma entidade limitada e independente, onde os dados podem ser armazenados e que contenha operações que podem ser realizadas nesses dados (ANDRADE, 2007). A herança é a definição como extensão ou modificação de uma classe (subclasse) a partir de uma ou mais classes existentes (superclasses).

O polimorfismo é o uso de uma mesma funcionalidade por mais de um objeto com estrutura internas diferentes. O encapsulamento é o ato de esconder do usuário informações que não são de seu interesse. O objeto atua

como uma caixa preta, onde usuário não toma conhecimento de como seus métodos atuam.

A escolha da utilização da linguagem de programação C++ deu-se por ser esta uma linguagem de alta flexibilidade, portabilidade e consistência adequada para grandes ou pequenos projetos. C++ também tem ampla disponibilidade e suporte e a conveniência de não estar sobre o domínio de nenhuma empresa.

Optou-se, a exemplo do uso do OpenGL e do FLTK, por um ambiente livre para programação com o C++ - a ferramenta Dev C++ desenvolvida por Bloodshed Software, sendo a última versão a 4.9.9.2 de 22 de fevereiro de 2005. (DEV-C++). DevC++ (também conhecido como Devcpp)⁴ é um ambiente de desenvolvimento livre, que utiliza os compiladores de licença GCC (*GNU Compiler Collection*) (ver Figura 21). Tal ambiente suporta as linguagens de programação C e C++, possui toda a biblioteca ANSI C, além de algumas bibliotecas similares às da Borland Turbo C, como, por exemplo, a conio2.h (DEV-C++).

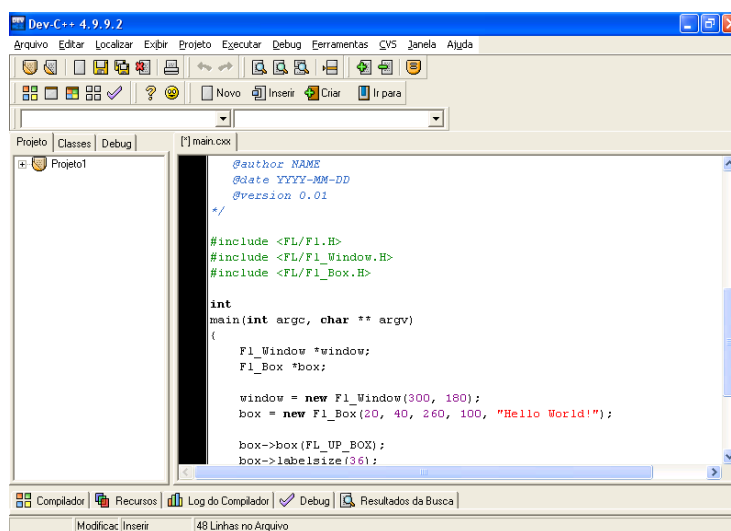


Figura 21: Janela do ambiente Dev C++.

Assim como a FLTK, o Dev C++ é de fácil operacionalidade e possui uma ampla documentação de apoio. É distribuída sob a licença LGPL (Library

⁴www.bloodshed.net/devcpp.html

General Public License) permitindo que os programas criados sejam comercializados sem distribuição do código_fonte.

2.4 Notação UML (*Unified Modeling Language*)

O sistema proposto é modelado a partir do uso da notação UML (*Unified Modeling Language*) (UML,2007), a qual será descrita a seguir.

Segundo Fowler (2000), UML é a sucessora de métodos de análise e projeto orientado a objetos que surgiu entre o final dos anos 80 e no início dos anos 90. UML é chamada de linguagem de modelagem e é uma notação gráfica utilizada por métodos para expressar projetos. Larman (2000) afirma que a UML é uma linguagem padrão para especificação da modelagem orientada a objetos, de forma que qualquer sistema possa ser modelado corretamente, com consistência, simplicidade.

A UML é considerada uma das linguagens mais expressivas para modelagem de sistemas orientados a objetos. Por meio de seus diagramas, é possível representar sistemas de softwares sob diversas perspectivas de visualização. Facilita, assim, a comunicação, entre as partes envolvidas, no processo de desenvolvimento de um sistema por apresentar um vocabulário de fácil entendimento (UML,2007).

A UML utiliza-se de muitos elementos (gráficos) de modelos utilizados na criação dos diagramas que representam partes do sistema e, dentre os diagramas suportados, estão o diagrama de caso de uso, o diagrama de classe, o diagrama de seqüência, o diagrama de colaboração, o diagrama de estado, o diagrama de atividade, o diagrama de componente e o diagrama de distribuição (HENSGEN, 2003).

3. FERRAMENTAS PARA MODELAGEM E SIMULAÇÃO ESPACIAL

Para embasar o desenvolvimento do sistema de simulação interativa de regimes urbanísticos proposto, foram estudadas algumas ferramentas de representação espacial de objetos usadas na área da Arquitetura e Urbanismo. Tais ferramentas empregam técnicas de Computação Gráfica, algumas destas já citadas na seção sobre Referencial Teórico.

Assim, serão abordadas neste capítulo ferramentas CAD (AutoCAD, ArchiCAD, DataCAD) e as ferramentas VectorWorks, FormZ, Blender, CityZoom, Radiance, EnergyPlus.

3.1 Ferramentas CAD

CAD (*Computer Aided Design*) é o nome genérico de softwares utilizados por áreas como a Engenharia, a Geologia, a Arquitetura e o Design para facilitar projetos e desenhos técnicos. Os sistemas CAD consistem numa série de ferramentas para construção de entidades geométricas planas (como linhas, curvas, polígonos) ou mesmo objetos tridimensionais (3D) (cubos, esferas, etc.)

Segundo KERRY (1997), os sistemas CAD propõem-se a auxiliar a manipulação e a criação das informações, sistematizando os dados de projetos envolvidos e possibilitando uma rápida reutilização de informações quando necessário.

Atualmente, os sistemas CAD são aplicados tanto para projetos bi (2D) como para projetos tridimensionais(3D).

Os softwares CAD, apesar de oferecerem uma boa funcionalidade, costumam ser utilizados por um nicho pequeno de usuários devido a sua intensa especialização e seu alto custo. Infelizmente, há pouca existência de ferramentas livres nessa área que, em muitos aspectos, ficam aquém dos softwares comerciais.

Além disso, uma das grandes decepções do CAD tem sido a dificuldade de se reutilizar dados, diz o Dr. Karthik Ramani professor da escola da Engenharia Mecânica de Purdue.

Uma vez que a informação tenha sido criada no CAD e utilizada, ela é freqüentemente guardada e esquecida. Como resultado, a indústria perde recursos por não ser capaz de reutilizar peças já projetadas anteriormente. A roda é reinventada muitas vezes.

O principal software CAD para indústrias pequenas, arquitetos e treinamento, é o AutoCAD (AUTODESK, 2007) (apresentado com mais detalhes na próxima seção).

Ainda neste capítulo, serão abordadas outras ferramentas CAD similares ao AutoCAD, como o ArchiCAD e o DataCAD.

3.1.1 AutoCAD

O AutoCAD é um software criado e comercializado pela Autodesk Inc., em 1982 e usado para geração e manipulação de informações vetoriais no computador. Tais informações são representadas por uma série de pontos que formam linhas, dando origem à representação gráfica correspondente ao tipo de informação contida em uma figura.

A precisão do AutoCAD é de até 12 casas decimais e permite, por exemplo, o desenho de uma cidade inteira em escala. Com as capacidades de geração de modelos em 2D e 3D.

O AutoCAD auxilia o profissional da construção desde a criação do projeto até seu detalhamento, passando pela documentação, gerenciamento de desenhos e visualização do projeto (BARRADAS,2007).

3.1.2 ArchiCAD

O ArchiCAD é um aplicativo CAD especialmente desenvolvido para o profissional de Arquitetura (ARCHICAD, 2007). Com este software, é possível a criação de plantas, cortes e elevações desde o nível de anteprojeto até a planta de execução. Também pode-se criar maquetes eletrônicas e perspectivas.

Com o ArchiCAD, pode-se trabalhar com vistas 2D e 3D; as alterações aplicadas na vista 3D são automaticamente passadas para 2D, e vice-versa. É um aplicativo que integra as diferentes partes de um projeto arquitetônico – documentação, modelo, quantitativo e apresentação - e esta é uma de suas principais vantagens. Outra característica relevante está na facilidade de alterar os diferentes elementos construtivos em um projeto como, por exemplo, paredes e lajes.

3.1.3 DataCAD

O DataCAD é um software criado com o objetivo de auxiliar a realização de projetos na área da Arquitetura e da Construção Civil (DATACAD, 2007).

O DataCAD é uma ferramenta de projeto com todos os recursos para desenho em 2D e em 3D, incluindo maquete eletrônica e renderização com textura e cores. Desenhos criados em outros softwares, como o AutoCAD, são convertidos pelos formatos DWG (*DraWinG*) ou DXF (*Drawing Interchange File*) padrão de arquivo utilizado que não depende da versão do software CAD ou do fabricante do software.

Os arquivos DWG são gravados em formato binários, o que dificulta a transferência via Internet. Os arquivos DXF possuem o mesmo conteúdo dos arquivos DWG, porém no formato texto ideais para internet. Daí surgiu o

DWF(Drawing Web Format) que possuem o mesmo tipo de estrutura do DWG e do DXF, porém compactados.

As principais inovações da Versão 12 são:

- entidades Inteligentes;
- operações Booleanas em 3D;
- paredes Inteligentes, Portas Inteligentes, Janelas Inteligentes;
- símbolos Inovadores; Planilhas do Excel Inseridas no Desenho;
- planos de Edição em 3D;
- textos em Multi-Linhas;
- importação de arquivos/objetos do formato Google SketchUp;
- novos Símbolos Prontos para Renderizar; Editor de Propriedades de Entidades;
- criação de MATERIAIS no DataCAD 12.

3.1.4 VectorWorks

VectorWorks é um programa de propriedade da Nemetschek NA e que integra toda a facilidade de uso dos mais avançados recursos de parametrização e desenho orientado a objetos (VECTORWORKS, 2007).

O VectorWorks traz para arquitetos, engenheiros e *designers* uma nova forma de gerar projetos em 2D e 3D com precisão e produtividade incomparáveis, combinando recursos avançados de apresentação, geração de planilhas e programação (*scripting*) (DUFFY, COLLINS, 2004). Possui um ambiente híbrido, permitindo inserir símbolos em seus desenhos que irão aparecer em vista 2D e em vista 3D.

Comandos e ferramentas são dispostos de forma lógica em uma interface totalmente gráfica. Pode-se, ainda, personalizar a interface, retirando

ou inserindo comandos e ferramentas como desejar, ou ainda mudando os atalhos de acesso a eles.

3.1.5 FormZ

Da empresa AutoDesSys, FormZ é um modelador de sólidos e de superfícies que contém uma variedade de ferramentas para criar e manipular a geometria 3D (TUTORIAL FORMZ, 2007). É uma ferramenta para projetos arquitetônicos, desenhos urbanos, ilustrações e animação de objetos industriais e de interiores (FORMZ, 2007). O programa possui dois ambientes de trabalho: modelagem e rascunho (*drafting*). O modo *drafting* pode ser usado para construir planos 2D para modelagem ou para manipular segmentos 2D extraídos de modelos 3D.

O ambiente de modelagem vem em três módulos: modelador (formZ), modelador e renderizador (formZ RenderZone) e com a opção de radiosidade (formZ RenderZone RadioZity) (FORMZ, 2007). O seu recurso mais significativo, sem dúvida, é o uso de modelagens de superfícies e de sólidos podendo-se, ainda, criar objetos sólidos a partir de superfícies e vice-versa. No formZ, o volume do objeto é reconhecido o que facilita a criação de objetos complexos. Atualmente, o formZ está na versão 3.9.5.

3.2 Blender

O Blender é um software livre e multiplataforma essencialmente para modelagem tridimensional (3D) (ver Figura 34), desenvolvido pela Fundação Blender, uma fundação sem fins lucrativos,(BRITO, 2006). Possui diversos recursos comparáveis aos dos softwares proprietários similares, como 3D Studio, Maya (AUTODESK, 2007) e Rhinoceros (RHINO3D, 2007) mas com a vantagem de estar disponível para download gratuito na internet (BRITO, 2006).

O Blender é uma ferramenta de recursos avançados, que disponibiliza uma interface gráfica sem programação, flexível e configurável pelo usuário (BALDEVI).

O Blender salva todas as informações da cena em um único formato, suporta compressão, criptografia, assinaturas digitais e pode ser utilizado como biblioteca(BALDEVI).

Tal versatilidade encontra aplicação nas áreas de animação e efeitos especiais para jogos e filmes, Arquitetura, Publicidade, Engenharia, Design e outra que necessite pré-visualização 3D. O Blender também possui um mecanismo de jogo (game engine), que permite criar cenas e objetos interativos, que podem reagir a movimentos e cliques do mouse, toques no teclado, e até a outros objetos e eventos da mesma cena.

3.3 CityZoom

CityZoom é uma ferramenta de suporte à tomada de decisão para projetos urbanos. Ele provê um ambiente onde diferentes modelos podem operar interativamente com o objetivo de otimizar o processo de planejamento urbano (LEDER et al., 2007)

O programa Cityzoom foi desenvolvido no Laboratório para Simulação e Modelagem em Arquitetura e Urbanismo (*SimLab*) da Universidade Federal do Rio Grande do Sul (UFRGS) pelo Professor Benamy Turkienicz e sua equipe. Este programa computacional, através da inserção de parâmetros reais, como a legislação urbana e dados cadastrais das zonas, simula tridimensionalmente o cenário urbano com o intuito de apoiar processo de planejamento e validar o projeto (Ver Figura 36).

Pode-se obter, por exemplo, dados numéricos de área ou população de um edifício, estudos de insolação e envelope solar, morfologia urbana e outros (GRAZZIOTIN, 2004).

Atualmente o *CityZoom* esta na sua versão 1.0 Beta e esta sendo liberado para testes em outras instituições.

3.4 Radiance

Radiance é um dos programas de simulação de iluminação com validade visual/fotométrica mais conceituado (CARDOSO e SOUSA, 2007). Esta ferramenta é um conjunto de subprogramas para a simulação da iluminação de um projeto para predizer a luminosidade (luz natural e artificial), a qualidade visual e a aparência de espaços em ambientes internos para avaliar tecnologias novas de iluminação. Os resultados da simulação podem ser indicados como imagens da cor, valores numéricos e lotes de contorno

O Radiance é muito sofisticado, mas por outro lado é difícil de usar, exigindo uma aprendizagem demorada. Funciona essencialmente em plataformas UNIX (em quase todos as variantes: Linux, Solaris). (CARDOSO, SOUSA, 2007).

A tabela abaixo (Tabela 1) compara as principais características entre a ferramenta proposta e os ambientes que foram apresentados neste capítulo.

TABELA 1: Comparação entre características das ferramentas apresentadas e da ferramenta proposta.

Principais características das ferramentas apresentadas					
	Ambiente de Modelagem	Visualização 3D	Utilizado para projetos arquitetônicos	Software livre	Utilização de regimes urbanísticos
AutoCAD	X	X	X	Não	
ArchiCAD	X	X	X	Não	
DataCAD	X	X	X	Não	

FormZ	X	X	X	Não	
Blender	X	X	X	Sim	
CityZoom	X	X	X	Não	X
Radianc e		X	X	Não	
Trabalho Proposto	X	X	X	Sim	

4. SOLUÇÃO PROPOSTA

Em função da carência de ferramentas de simulação e modelagem voltadas à área da Arquitetura e Urbanismo, capazes de permitir as especulações numéricas e morfológicas dos regimes construtivos e volumétricos, foi desenvolvida uma solução computacional que venha a suprir tais necessidades.

A funcionalidade, modelagem e interfaces gráficas são apresentadas nas próximas seções.

4.1 Visão Geral

A solução proposta visa permitir, ao usuário, trabalhar com os regimes construtivos e volumétricos previstos no Plano Diretor dos Municípios, tais como informações numéricas relativas à área, às dimensões do lote e do edifício de maneira gráfica e textual. A partir destas, deverá ser gerada, como resposta, a visualização das alternativas de forma do edifício bem como seus parâmetros numéricos resultantes. A cada proposição do usuário, o sistema retorna as possibilidades, enunciando sua adequação ou não às normas dos regimes construtivos e volumétricos.

4.2 Modelagem

Como forma de expressar a modelagem do projeto proposto, optou-se por apresentar diagrama de casos de uso (ver Figura 22), diagrama de seqüência (ver Figura 23 e 24) e diagrama de classes (ver Figura 28) e diagrama de atividade (Ver Figura 25, 26, 27) descritos a seguir.

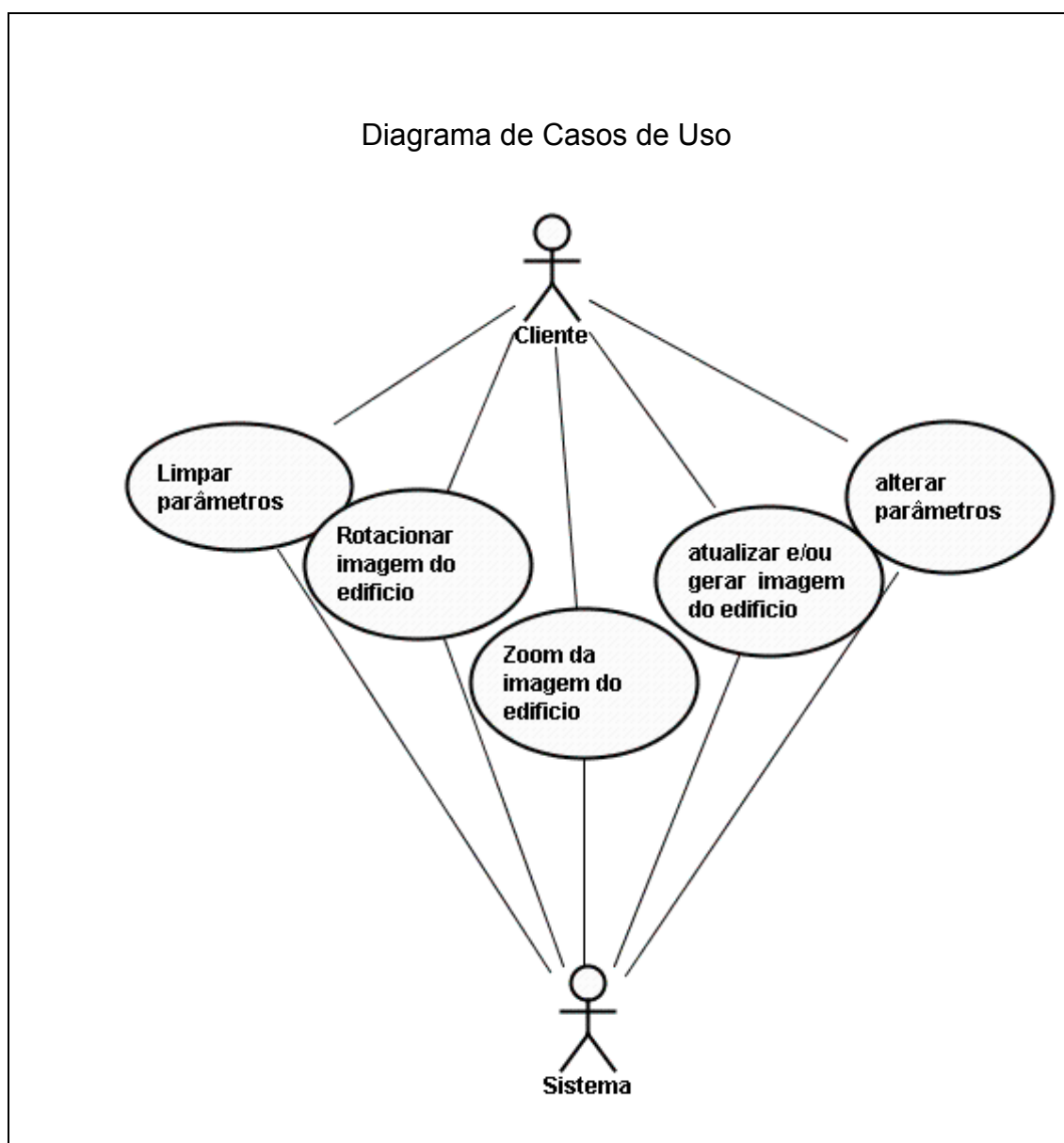


Figura 22. Diagrama de Casos de Uso

Os casos de uso apresentados na Figura 22 são descritos nas Tabelas 3, 4, 5, 6 e 7 descritas após a Tabela 2, a qual traz as referências empregadas em tais descrições.

TABELA 2: Referências aos Casos de Uso

Tabela de referências.		
Referencia	Descrição	tipo
A	Largura (lote)	Campo
B	Profundidade (lote)	Campo
C	Índice de Aproveitamento	Campo
D	Taxa de Ocupação	Campo
E	Recuo Jardim	Campo
F	Altura máxima (edifício)	Campo
G	Altura da divisa (edifício)	Campo
H	Recuos Lateral/fundos	Campo
I	Laje	Campo
J	Frente do Edifício	Campo
K	Profundidade do Edifício	Campo
L	Numero de Pavimento	Campo
M	Altura do Pavimento	Campo
N	Área Total	Campo
O	Potencial Construtivo	Campo
P	Laje Máxima	Campo
Q	Potencial Construtivo Realizado	Campo
R	Recuos Realizados	Campo
S	Altura Realizada	Campo
R01	Aumenta Frente	Botão de rolagem
R02	Aumenta Profundidade	Botão de rolagem
R03	Aumenta Altura	Botão de rolagem
01	Zoom	Botão
02	Rotação vertical	Botao
03	Rotação Horizontal	Botão

04	Limpar	Botão
05	Atualizar	Botão
06	Sair	Botão
Men01	Necessário preenchimento dos regimes construtivos	Mensagem
Men02	Necessário preenchimento dos valores referentes ao lote	Mensagem
Men03	Necessário que preencha o numero de pavimentos	Mensagem
Men04	Frente do edificio excedeu a largura do lote ou ultrapassou recuos laterais	Mensagem
Men05	Profundidade do edificio excedeu profundidade do lote	Mensagem
Men06	Laje excedeu valor máximo pra laje	Mensagem
Men07	Numero de pavimentos ultrapassou a altura máxima do edificio	Mensagem

TABELA 3: Caso de uso Alterar ou Gerar Imagem do Edifício

Caso de Uso: Atualizar ou Gerar Imagem do Edifício
Atores: Cliente
Finalidade: Gerar ou atualiza uma Imagem do Edifício
Tipo: Primário e Real
Descrição: Usuário Informa valores nos campos A ,B, D, E, F, G, H, e mais dois campos quaisquer dos campos I, J, K, L,M . Usuário aperta o botão 05. Caso haja campos não preenchido, o sistema emite uma das mensagens Men01 à Men03. Após o usuário pressionar o botão 05, o sistema gera os resultados nos campos O, P, Q, R, S, verifica-os e gera mensagem caso o valor do campo J ou K tenha ultrapassado o percentual informado no campo H. O sistema emite a mensagem Men04 ou Men05 e, caso o campo Q tenha um resultado superior ao do campo O, o sistema emitirá o resultado do campo Q em vermelho. Caso o resultado obtido no campo P seja inferior ao informado no campo I, o sistema emitirá a mensagem Men06. Caso o campo S tenha como resultado um valor superior ao valor digitado no campo F, o sistema emitirá a mensagem Men07. Após correções, o sistema gera imagem

TABELA 4: Caso de uso alterar parâmetros

Caso de Uso: Alterar parâmetros
Atores: Cliente
Finalidade: Alterar os parâmetros geradores da Imagem do Edifício
Tipo: Primário e Real
Descrição: Usuário pode apenas selecionar o campo que deseja modificar o valor de entrada e digitar o novo valor ou ele pode pressionar para o botão de rolagem 01 para esquerda, caso queira diminuir o valor, ou direita, caso queira aumentar o valor do campo J, ou pressionar o botão de rolagem 02 para a direita ou esquerda para aumentar ou diminuir o valor do campo k ou ainda pressionar o botão de rolagem 03 para a direita ou esquerda se quiser aumentar ou diminuir o campo S. Porém, se ele quiser alterar todos os campos referentes ao edifício, poderá pressionar o botão 4 para limpar todos os campos e preenchê-los novamente e, por fim, pressionar o botão 05 para atualizar a imagem. Caso haja erros, será tratado como foi explicado no caso de uso atualizar ou gerar imagem.

TABELA 5: Caso de uso Limpar Parâmetros

Caso de Uso: Limpar Parâmetros
Atores: Cliente
Finalidade: limpar os campos referentes aos parâmetros do Edifício
Tipo: Primário e Real
Descrição: Usuário após informar valores de todos os campos I, J, K, L, M, N - ou apenas parte deles - e desejar alterá-los, deve pressionar o botão 04 que limpará todos esses campos que não poderão ser mais recuperados. Poderá, a partir daí, ser digitado novos valores para a geração de uma nova imagem.

TABELA 6: Caso de uso Rotacionar a Imagem.

Caso de Uso: Rotacionar a Imagem
Atores: Cliente
Finalidade: Rotacionar a imagem no sentido horário e anti-horário verticalmente e horizontalmente
Tipo: Primário e Real
Descrição: Usuário após a alteração ou geração da imagem, poderá rotacioná-la, através do botão 02, para a esquerda no sentido anti-horário ou direita no sentido horário. Caso deseje rotacionar a imagem verticalmente, poderá, também através do botão 03, rotacionar para a esquerda no sentido anti-horário ou direita no sentido horário caso deseje rotacionar a imagem horizontalmente.

TABELA 7: Caso de uso Zoom da Imagem.

Caso de Uso: Zoom da Imagem
Atores: Cliente
Finalidade: Aumentar ou diminuir o tamanho da imagem gerada
Tipo: Primário e Real
Descrição: Usuário após a alteração ou geração da imagem, poderá aumentá-la ou diminuí-la através do botão 01; para a esquerda, aumentará a imagem, e para a direita, diminuirá a imagem.

A seguir, serão demonstrados os Diagramas de Seqüência (Figuras 23 e 24) Atividades (Figuras 25, 26 e 27), referentes aos Casos de Usos referenciados nas Tabelas 2, 3, 4, 5 e 6, bem como o Diagrama de Classes (Figura 28).

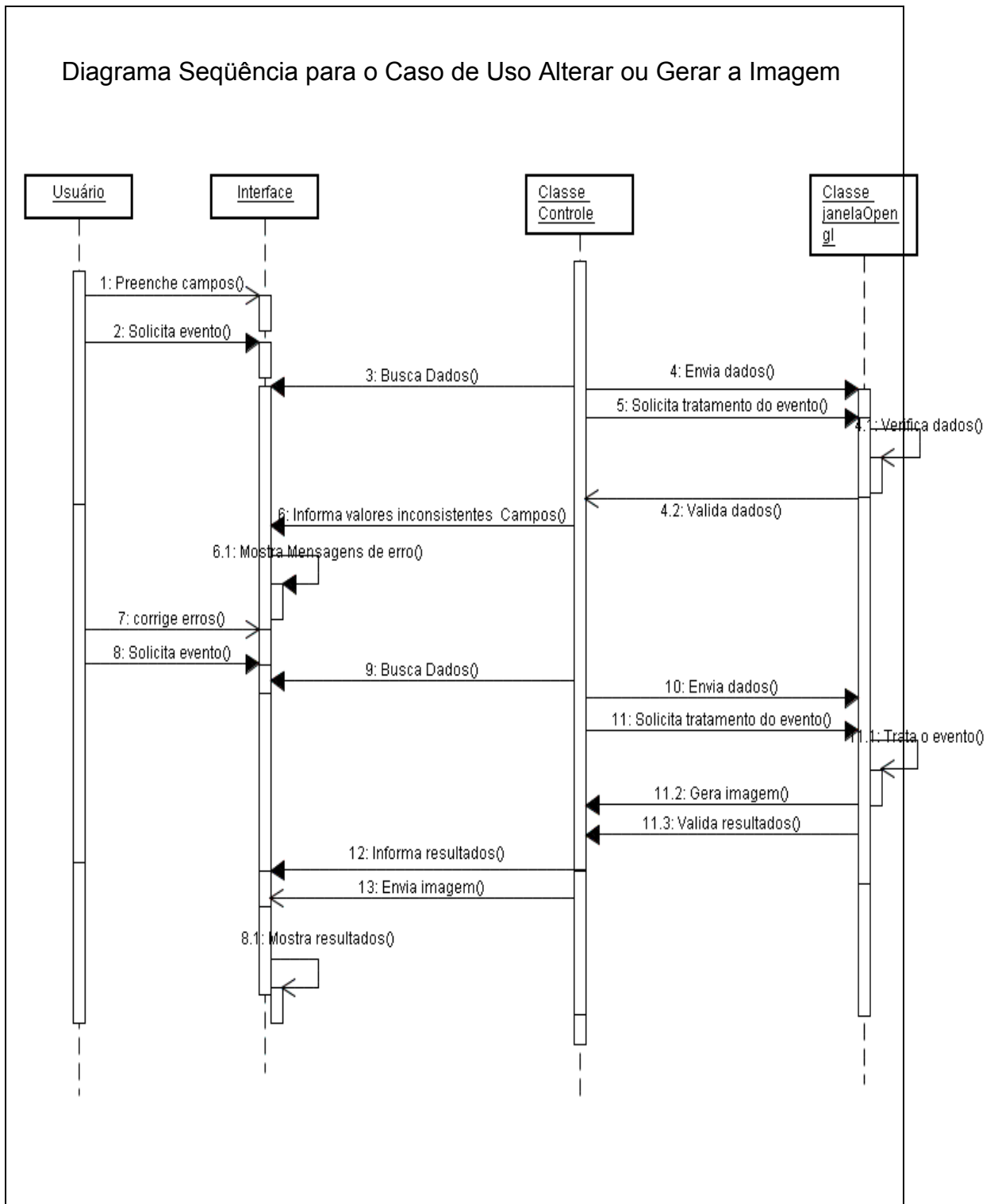


Figura 23 : Diagrama de Seqüência Alterar ou Gerar Imagem.

Diagrama Seqüência para o Caso de Uso Alterar parâmetros, Rotacionar Imagem, limpar parâmetros ou zoom da Imagem.

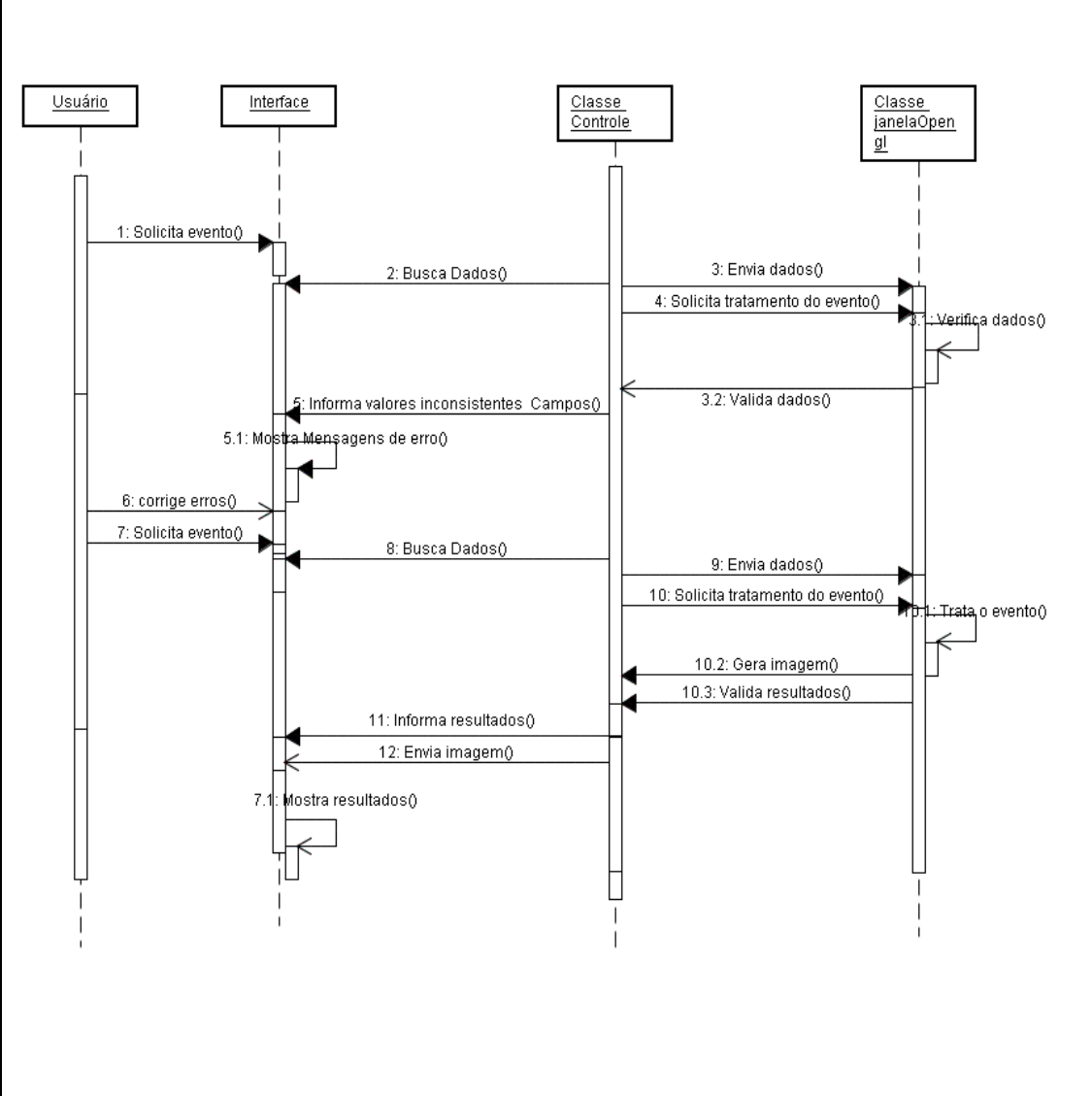


Figura 24 : Diagrama de Seqüência para tratamento de eventos.

Diagrama de Atividades para o Caso de Uso Alterar ou Gerar Imagem

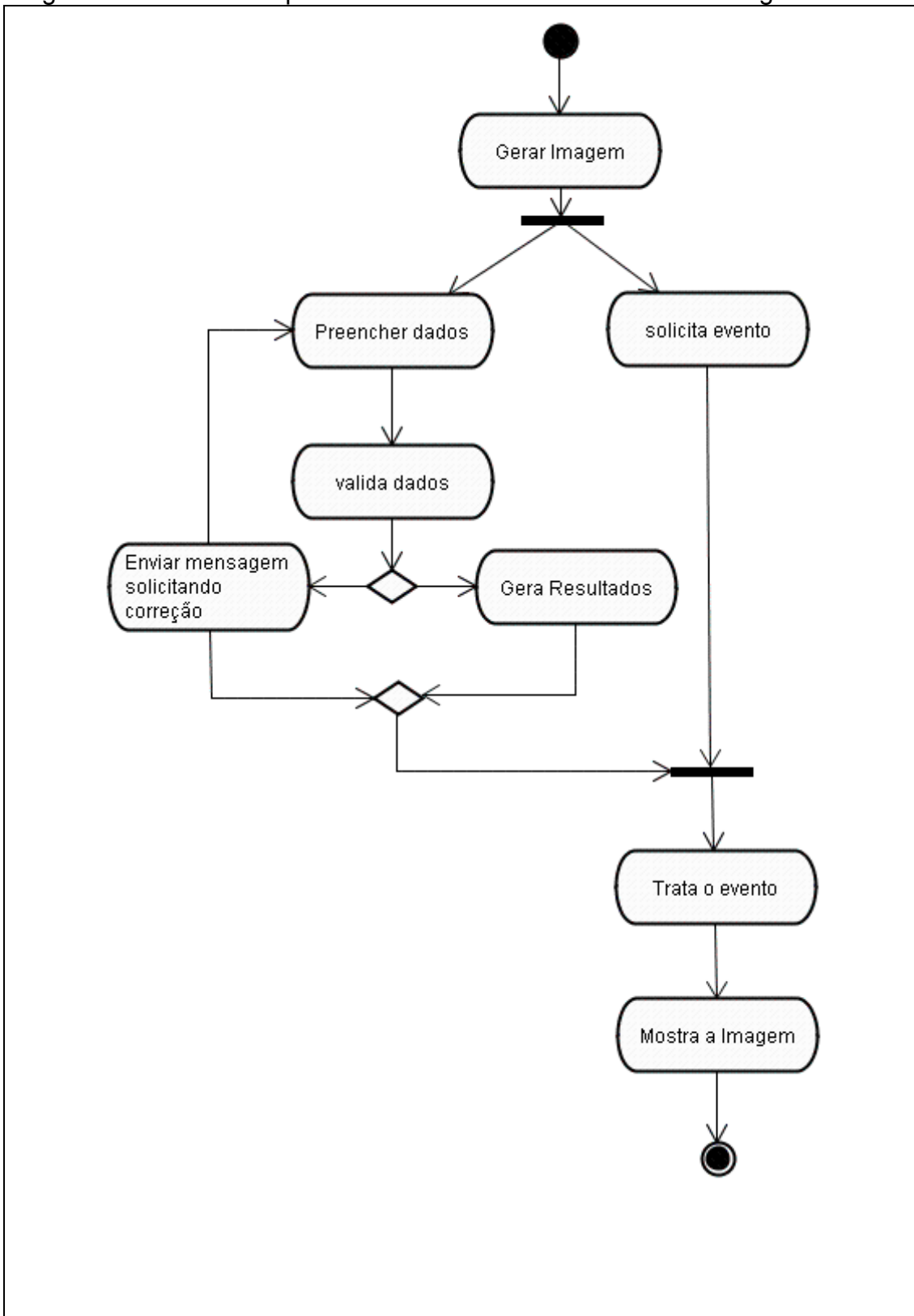


Figura 25: Diagrama de Atividades Alterar ou Gerar Imagem.

Diagrama de Atividades para Caso de Uso Alterar ou limpar Parâmetros

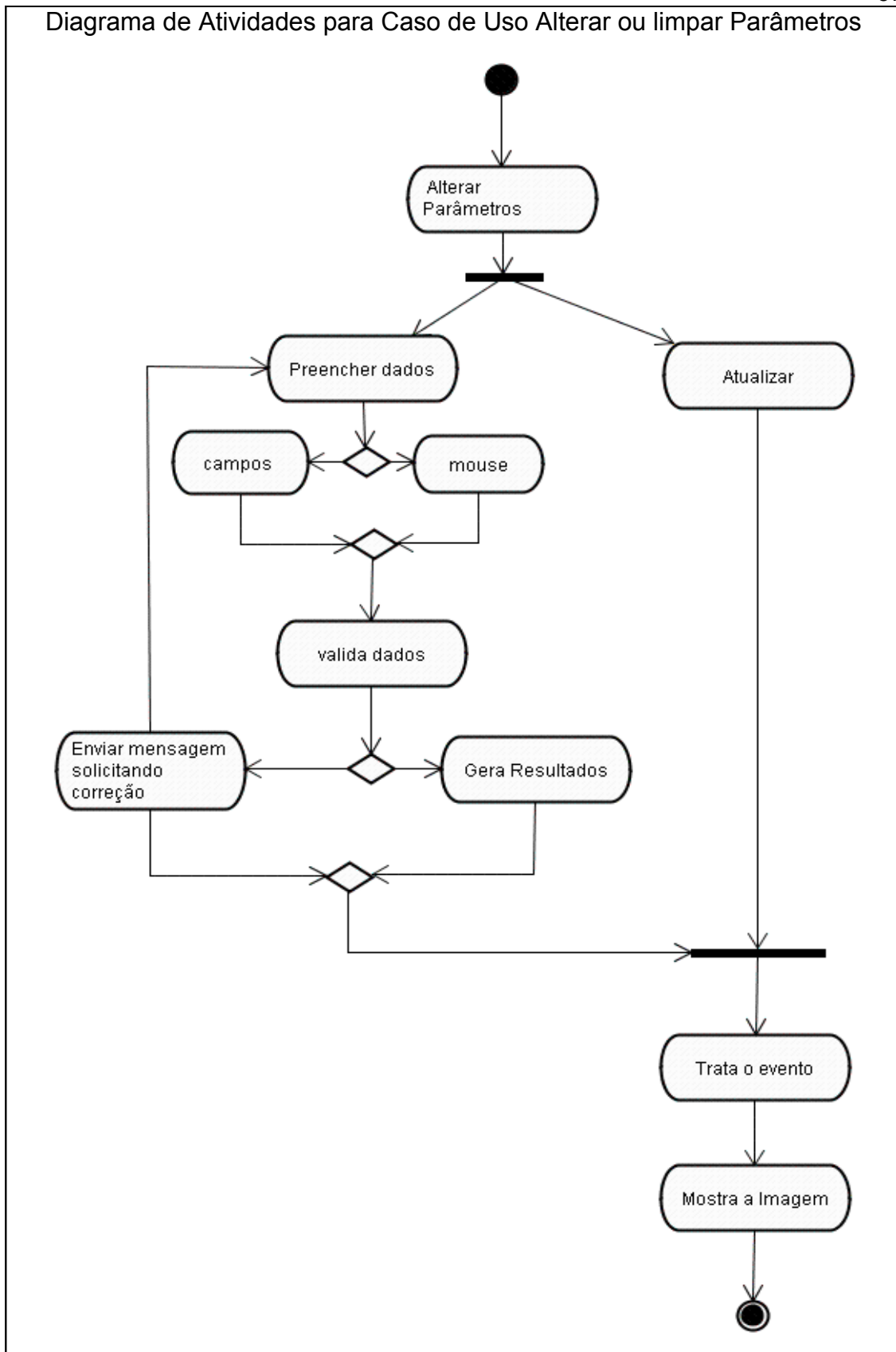


Figura 26: Diagrama de Atividades Alterar Parâmetros.

Diagrama de Atividades para Caso de Uso Rotação ou Zoom da Imagem

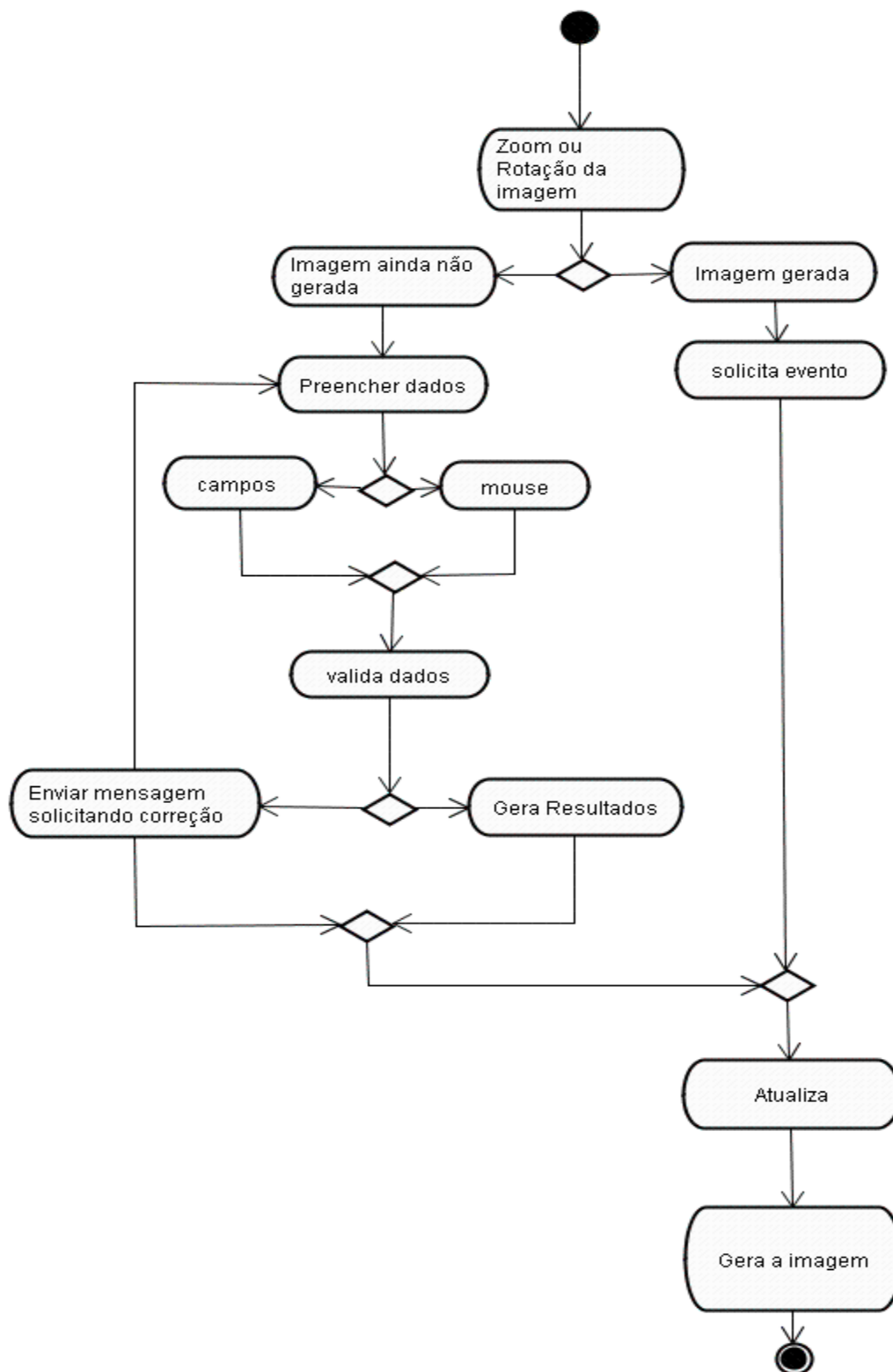


Figura 27: Diagrama de Atividades Rotação ou Zoom da Imagem.

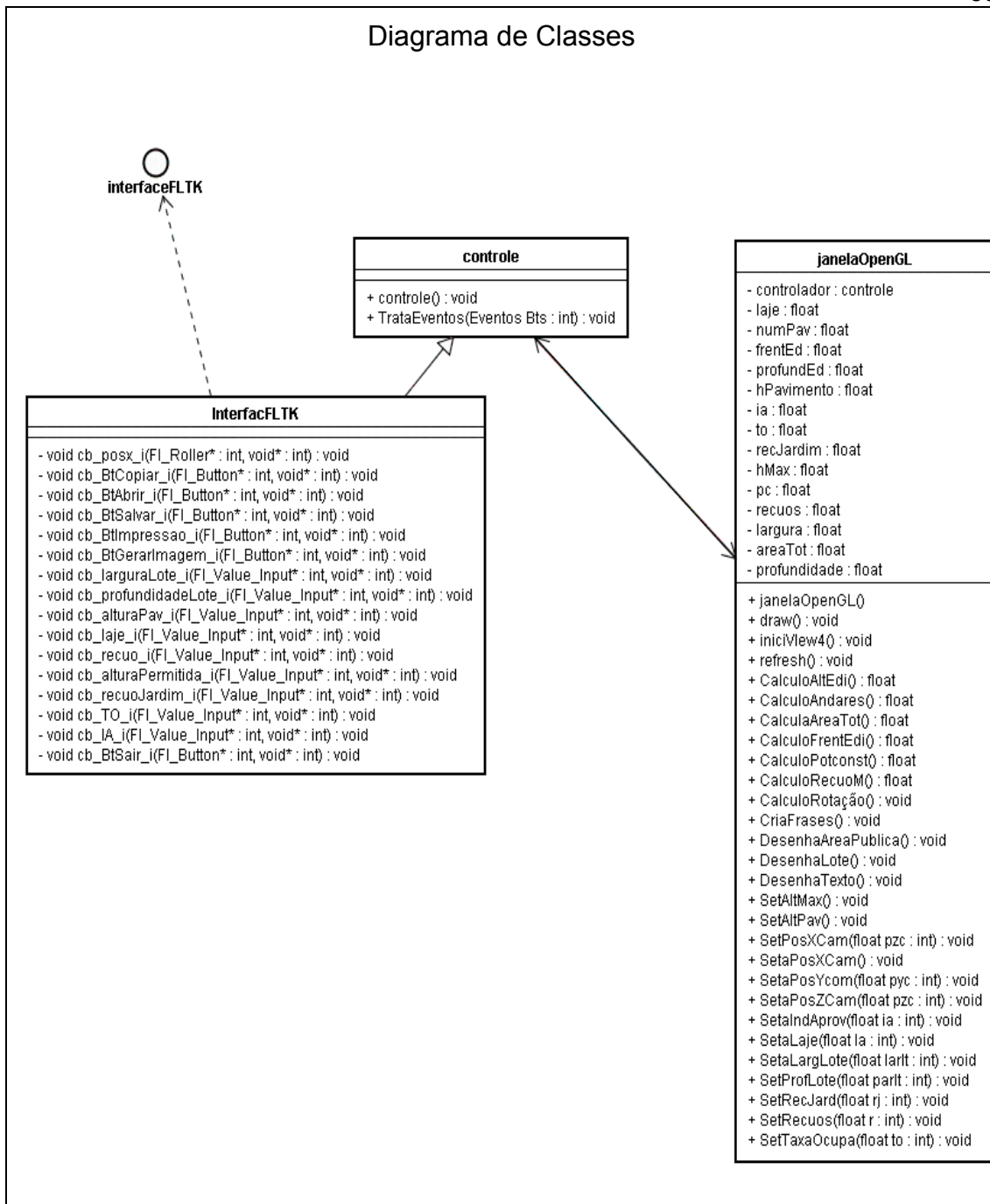


Figura 28: Diagrama de Classe.

Conforme mostra a Figura 28, a arquitetura do sistema proposto está baseada no padrão *Model-View-Controller* (MVC). A arquitetura desenvolvida inclui, assim, as seguintes classes: InterfaceFLTK, JanelaOpengl e Controle.

A classe InterfaceFLTK é responsável pela interação com o usuário com a interface e, juntamente com a classe Controle permitem a especificação da funcionalidade do sistema – Controle do MVC. A classe JanelaOpengl é responsável pelo recebimento dos parâmetros informados pelo usuário e, execução de cálculos, validação de informações e geração da imagem – Modelo do MVC. A imagem gerada pela classe JanelaOpengl e os objetos da interface responsáveis pela exibição da mesma através da classe InterfaceFLTK representam a Visão do MVC.

4.3 Interface Gráfica e Funcionalidade

Como já mencionado, a ferramenta proposta visa prover maior rapidez nas simulações urbanísticas, permitindo aos arquitetos uma maneira de expor os dados de forma transparente para as demais partes interessadas. Pretende-se, assim, aumentar nível de participação destes profissionais em decisões estratégicas das cidades, tais como a formulação de um novo Plano Diretor ou a alteração de parte de um existente.

De modo a atender tais questões, a ferramenta possui interface gráfica interativa que, à medida que o usuário informa valores, valida-os acusando erros. Embora o usuário tenha liberdade para informar diferentes valores, estes devem seguir uma hierarquia de modo que os resultados possam ser calculados corretamente. Assim, a ferramenta chama a atenção do usuário para o caso deste entrar com valores inconsistentes. Alguns dos campos onde são exibidos os valores calculados pela ferramenta não permitem a entrada de dados pelo usuário.

A ferramenta possui diversos campos numéricos e alguns destes devem ser preenchidos pelo usuário enquanto outros a ferramenta fornecerá como resultado (ver Figura 29). Tais parâmetros estão divididos em três categorias: “Parâmetros do Lote”, onde encontram-se as medidas de largura e profundidade do terreno, “Regimes Urbanísticos”, onde constam valores

referentes às regras construtivas do Plano Diretor de uma determinada zona, e “Parâmetros do Edifício”, cujos parâmetros estão relacionados às dimensões da construção. A seguir, cada um destes parâmetros será abordado com mais detalhe.

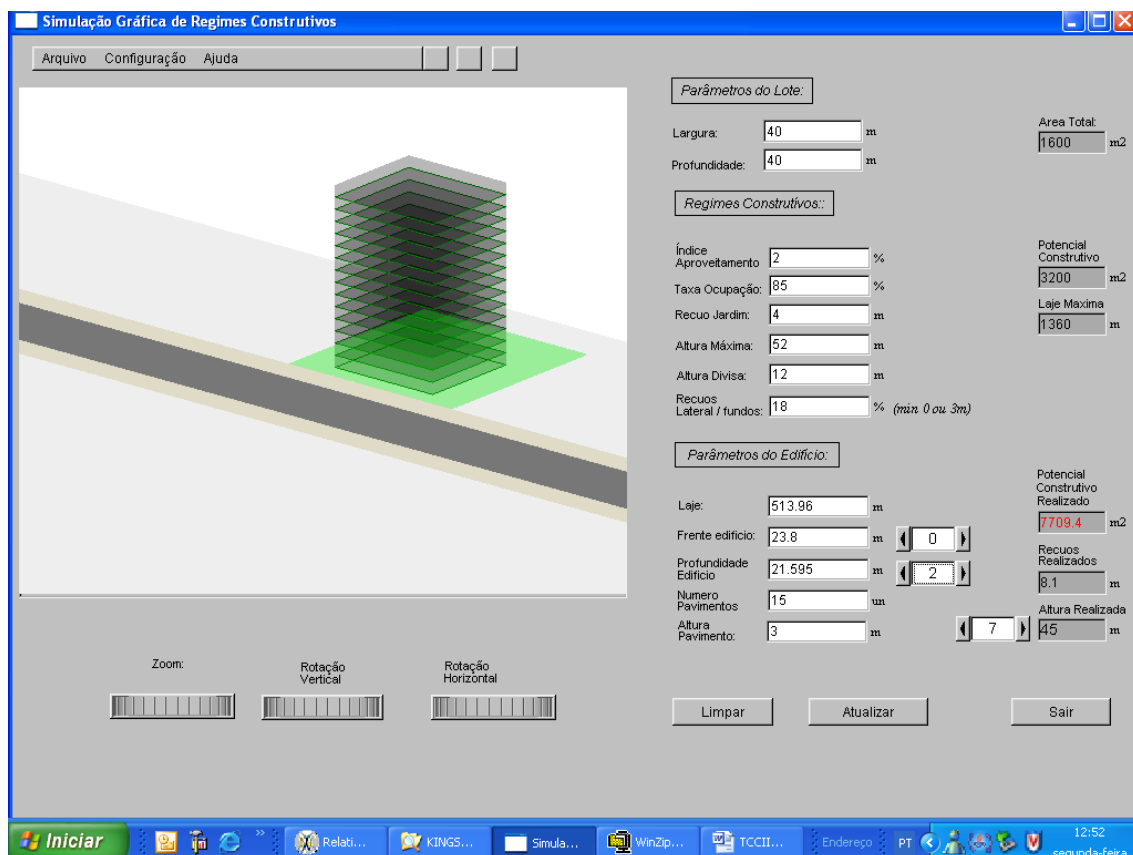


Figura 29: Interface do sistema proposto.

Os campos a serem preenchidos pelo usuário são:

- Valores referentes ao lote: “Largura” e “Profundidade”;
- Valores referentes aos regimes construtivos: “Índice de Aproveitamento”, “Taxa de Ocupação”, “Recuos de Jardim”, “Altura Máxima”, “Altura da Divisa”, “Recuos”, “Largura Lateral/fundos”.

O usuário deverá, também, preencher pelo menos dois campos dos parâmetros referentes ao edifício que são: “Laje”, “Frente do Edifício”, “Profundidade do Edifício”, “Número de Pavimentos” e “Altura do Pavimento”.

A partir de tais parâmetros, são calculadas e fornecidas, pela ferramenta, as informações dos campos a seguir: “Área Total”, “Potencial Construtivo”, “Laje Máxima”, “Potencial Construtivo Realizado”, “Recuos Realizados” e “Altura Realizada”. Gerará, também como respostas, os valores dos campos referentes ao edifício que não foram preenchidos pelo usuário.

O campo “Área” é calculado pela multiplicação dos valores dos campos “Largura” e “Profundidade” (Ver Figura 30)(Veja Código no anexo A).

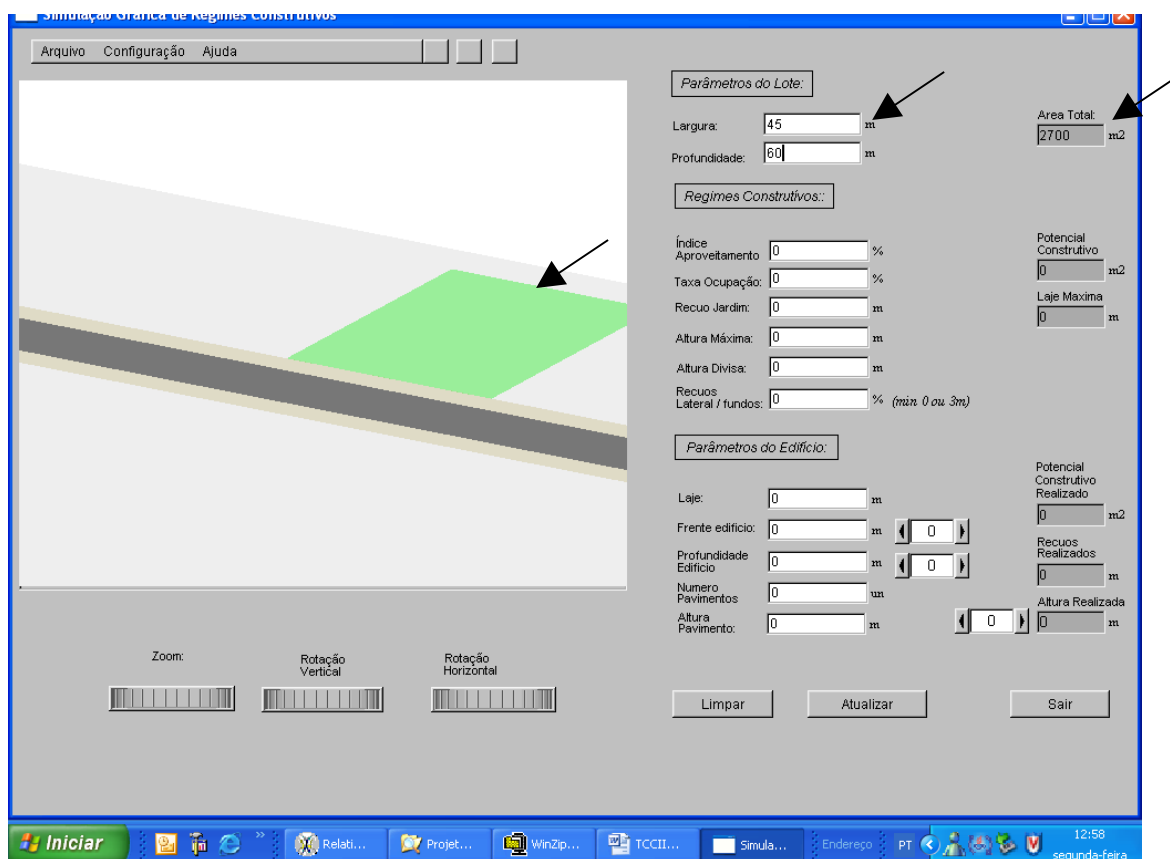


Figura 30. Cálculo da “Área Total” do lote pelo sistema proposto.

De modo a dar *feedback* para o usuário sobre a validade dos parâmetros informados, a ferramenta mostra mensagem de advertência caso este deixe de preencher algum ou todos os campos referentes aos parâmetros do lote e passe a preencher os campos referentes aos “Regimes Construtivos” (Ver Figura 31).

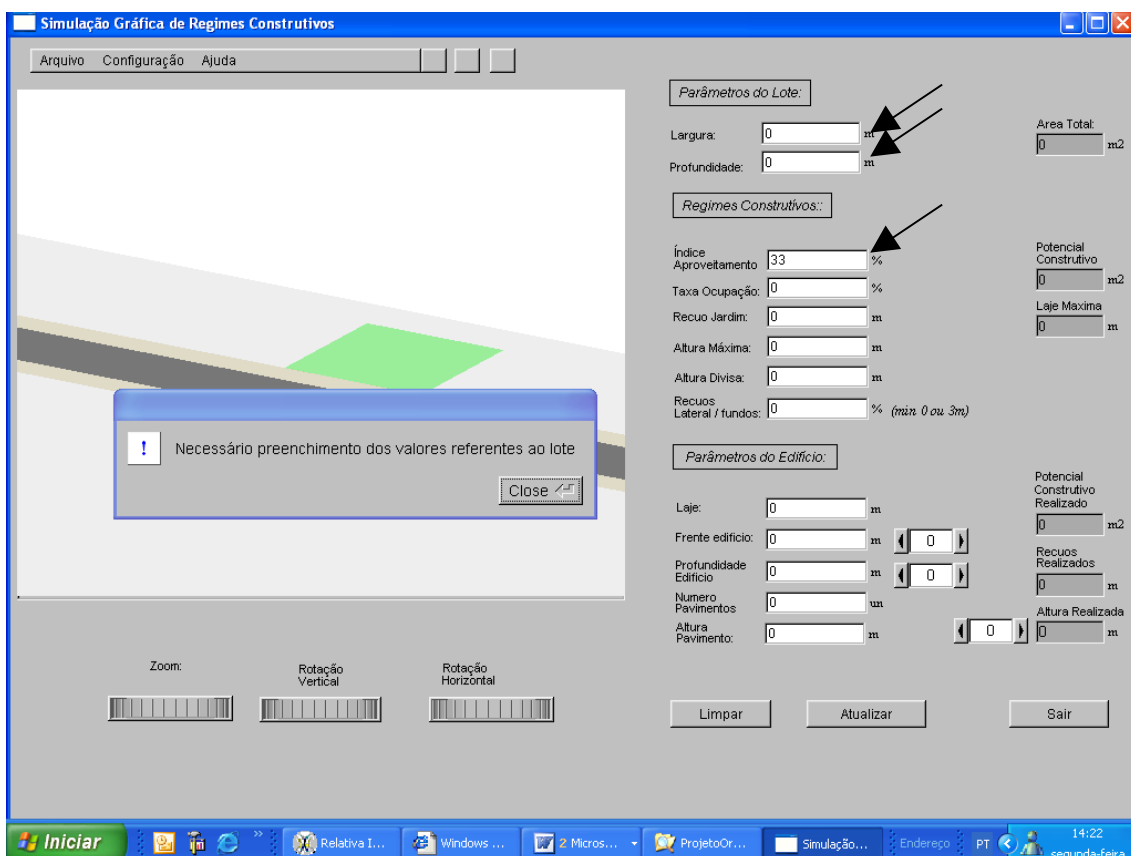


Figura 31. Mensagem de advertência para preenchimentos dos “Parâmetros do Lote” .

O campo “Potencial Construtivo” será obtido pela multiplicação do campo “Área Total” pelo campo “Índice de Aproveitamento”. Este valor será o limite máximo para o “Potencial Construtivo Realizado” (Ver Figura 32).

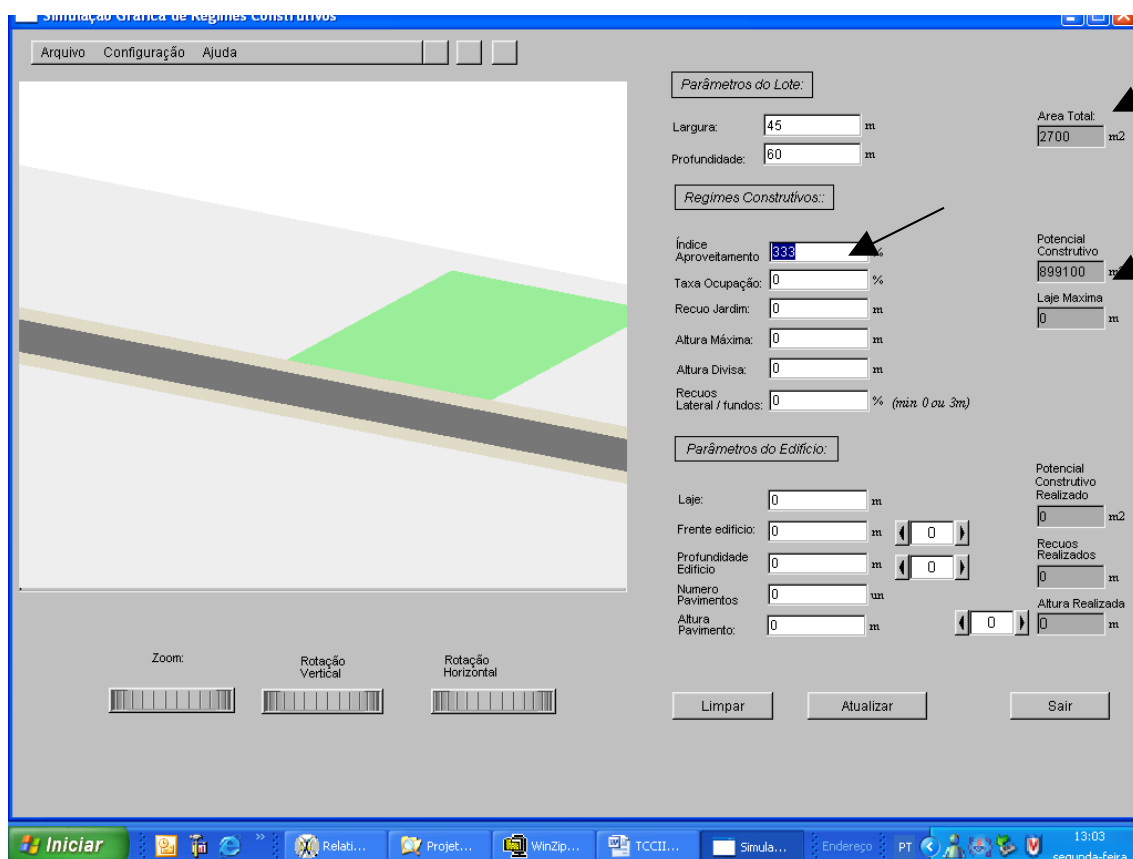


Figura 32. Imagem da entrada de dados “Índice de Aproveitamento” e a resposta “Potencial Construtivo”.

O valor da “Laje Máxima” será obtido pelo cálculo da multiplicação da “Área Total” pela “Taxa de Ocupação”. Este limita o valor máximo da laje que poderá ser gerada no lote estabelecido pelos parâmetros “Largura” e “Profundidade” (ver Figura 33). Caso o valor da “Laje” ultrapasse o valor da “Laje Máxima”, uma mensagem de advertência será mostrada para o usuário.

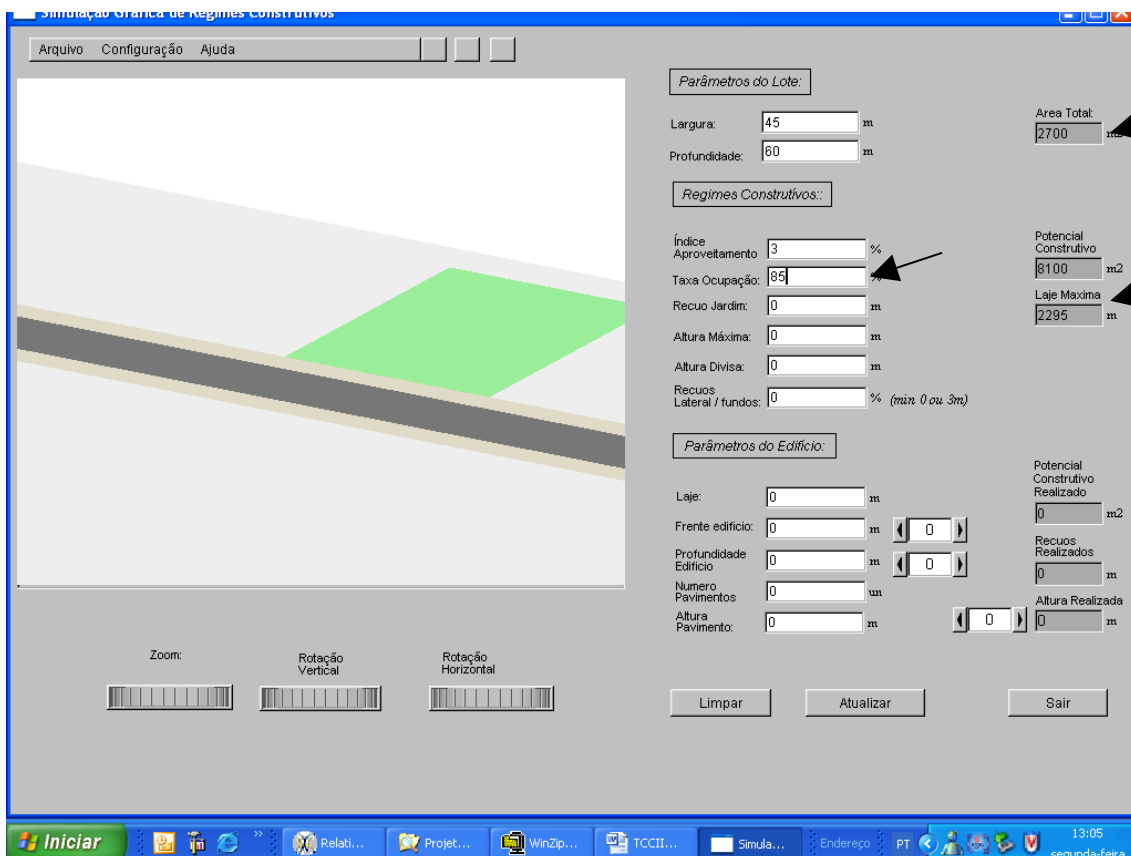


Figura 33. Preenchimento do campo “Taxa Ocupação” e a resposta gerada no campo “Laje Máxima”.

A partir deste ponto, o usuário deverá preencher os campos restantes dos parâmetros construtivos antes de preencher algum campo dos parâmetros do edifício. Caso o usuário não preencha os demais campos dos regimes construtivos e comece a preencher algum campo dos parâmetros do edifício, uma mensagem de advertência será mostrada para o usuário (ver Figura 34).

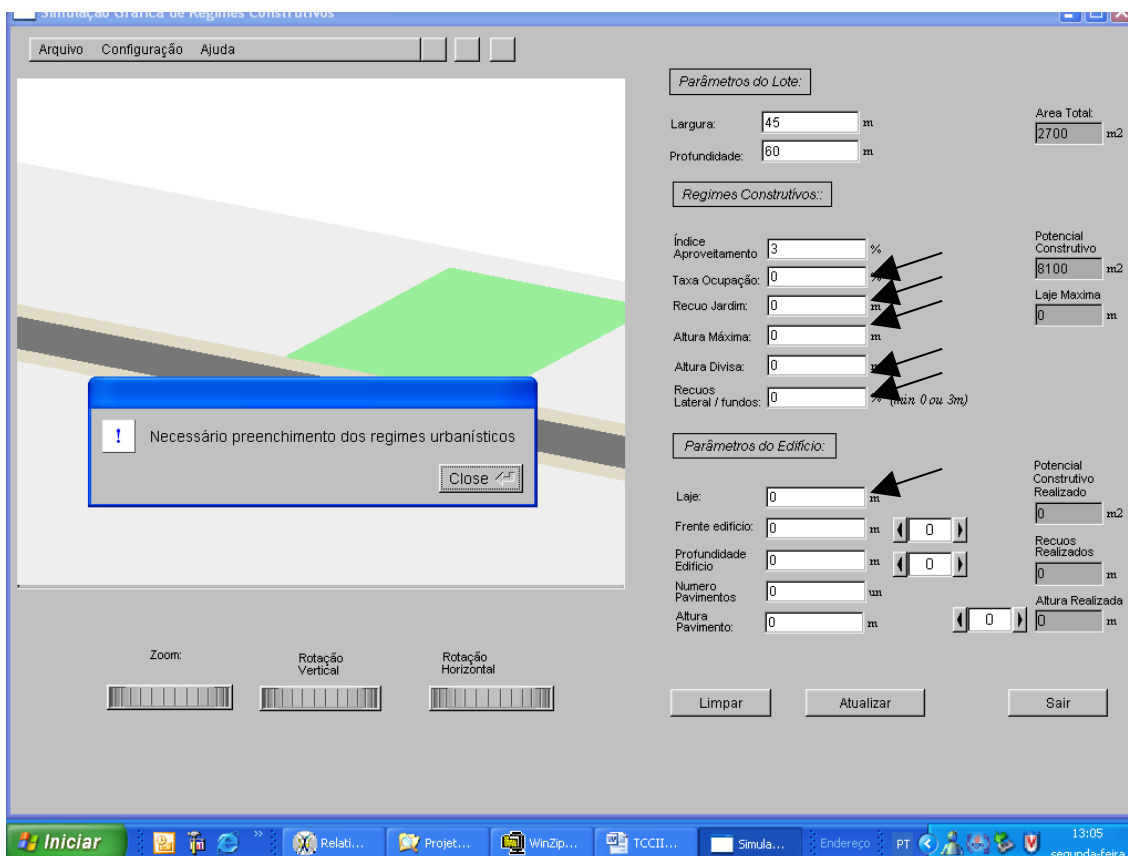


Figura 34. Mensagem de atenção para o preenchimento dos campos referentes aos “Regimes Construtivos”.

Após o preenchimento dos parâmetros referentes aos regimes construtivos, o usuário deverá preencher pelo menos dois campos referentes aos parâmetros do edifício. Com base nestes valores, os demais serão calculados.

O cálculo da “Laje”, caso não tenha sido preenchido, pode ser obtido pela multiplicação da “Profundidade Edifício” pela “Frente do Edifício” ou, caso um desses não tenha sido fornecido, poderemos obter também pela divisão do “Potencial Construtivo Realizado” pelo “Numero Pavimentos”. O valor da “Frente do edifício” pode ser obtida pela divisão da “Laje” pela “Profundidade Edifício” ou pelo dobro do valor “Recuos Realizados” subtraído da “Largura” (do lote). O valor da “Profundidade Edifício” pode ser calculada dividindo-se a “Laje” pela “Frente Edifício” ou pela soma dos valores de “Recuo Jardim” com o

valor dos “Recuos Realizados” e posterior subtração de tal soma pela “Profundidade” (lote).

Já o “Numero de Pavimentos” é obtido pela divisão da “Altura Realizada” pela “Altura do Pavimento”. A “Altura Realizada” é calculada pela multiplicação do “Numero Pavimentos” pela “Altura Pavimento” ou, caso não se tenha um desses dois valores, pela divisão do “Potencial Construtivo” pela “Laje Máxima” ou “Potencial Construtivo Realizado” pela “Laje”. Caso não tenha sido informado o valor da “Altura Pavimento”, o sistema assume o padrão de 3m.

Os “Recuos Realizados”, se não fornecidos, são calculados com base na “Altura Realizada” ou pelo valor da “Frente edifício”. Porém, existem ressalvas estipuladas pelo “Recuos lateral/fundos” e pela “Altura da Divisa”, que determinam o mínimo para os “Recuos Realizados” - pode ser 0 ou 3 metros e nunca entre estes valores.

Todos estes cálculos só são executados mediante o evento gerado pelo acionamento do botão “Atualizar” da Interface do sistema desenvolvido (ver Figura 35).

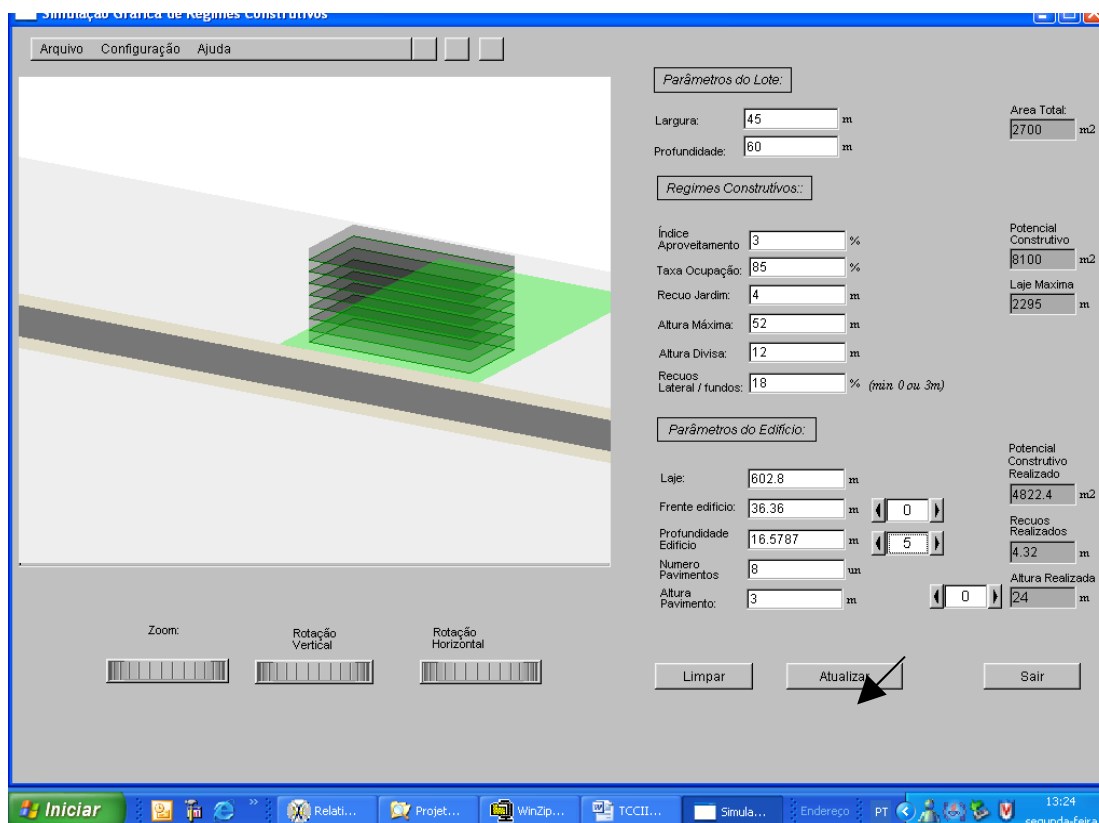


Figura 35. Imagem do acionamento do botão “Atualizar” da Interface do sistema desenvolvido.

O botão “Atualizar” é tratado na classe Controle e realiza a verificação do preenchimento de todos os parâmetros relativos aos “Regimes Construtivos” e aos “Parâmetros do Lote”. Também é verificado, na Classe Controle, quais valores dos “Parâmetros do Edifício” foram oferecidos e solicitado, para a Classe JanelaOpenGL, a realização dos cálculos.

Além destes, na classe Controle também é verificado se os parâmetros não ultrapassaram os limites permitidos como recuos laterais e de profundidade e, caso tenha acontecido, envia mensagem de advertência. Além disso, verifica se o “Potencial Construtivo Realizado” não ultrapassou o “Potencial Construtivo” e, caso tenha acontecido, o valor do “Potencial Construtivo Realizado” será mostrado em vermelho, a fim de chamar a atenção do usuário para o erro (ver Figura 36). Por fim, além de mostrar o “Potencial Construtivo Realizado” mostra os resultados dos campos “Recuos Realizados” e “Altura Realizada”. (Veja Código no anexo B).

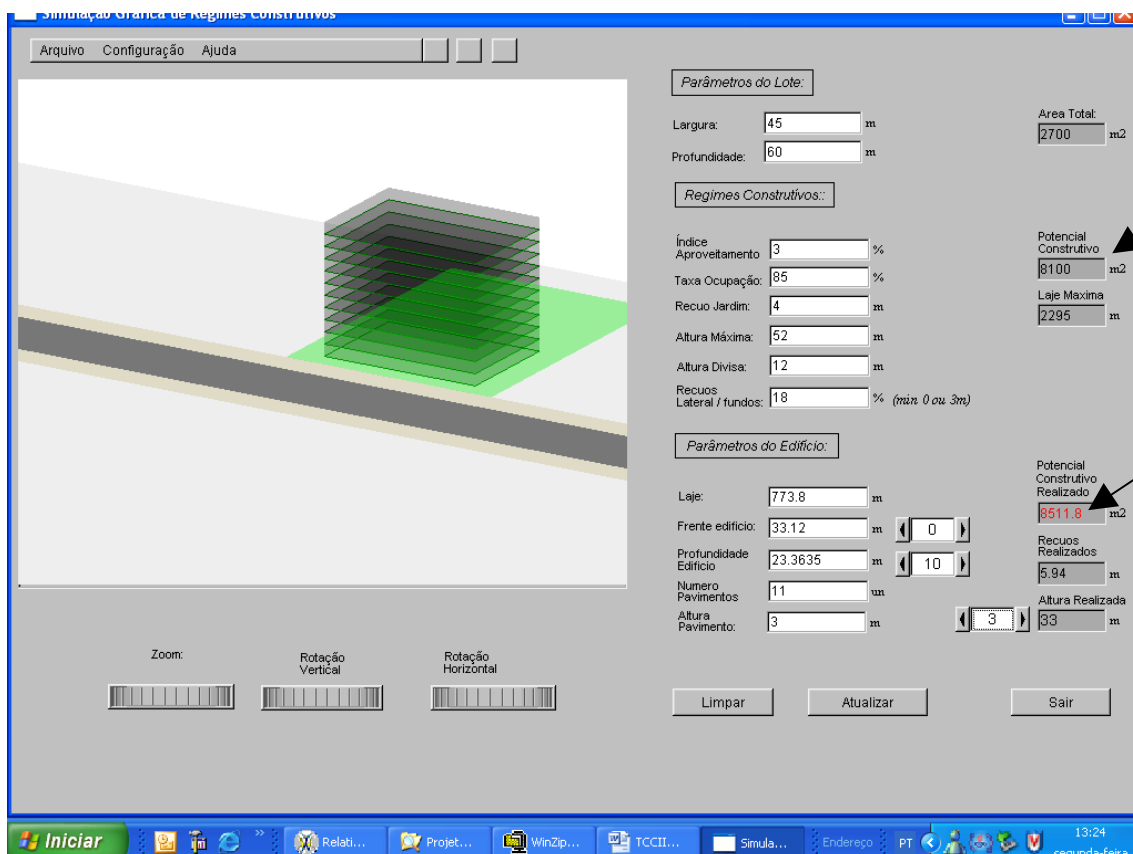


Figura 36. Imagem do “Potencial Construtivo Realizado” maior que o “Potencial Construtivo”.

A ferramenta também oferece para os usuários botões de rolagem onde os valores “Frente edifício”, “Profundidade edifício” e “Altura Realizada” podem ser rapidamente modificados. Para tanto, basta selecionar os componentes contadores - ao lado de cada um destes campos - que as alterações nas dimensões da imagem serão geradas permitindo melhor iteração do usuário com esta.

Além disso, a ferramenta também oferece como iteração do usuário com a imagem os botões de “Zoom”, “Rotação Vertical” e “Rotação Horizontal” (Ver Figuras 37,38,39).

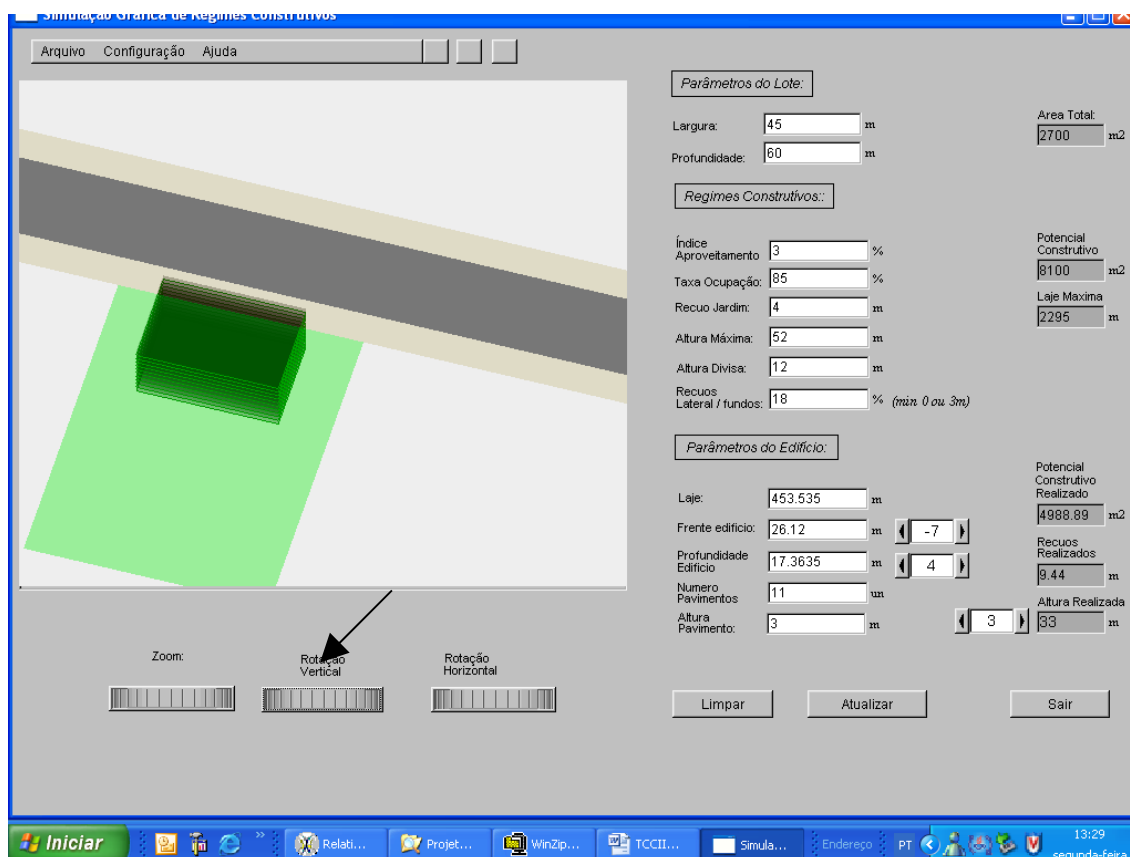


Figura 37. Imagem rotacionada com o botão “Rotação Vertical”.

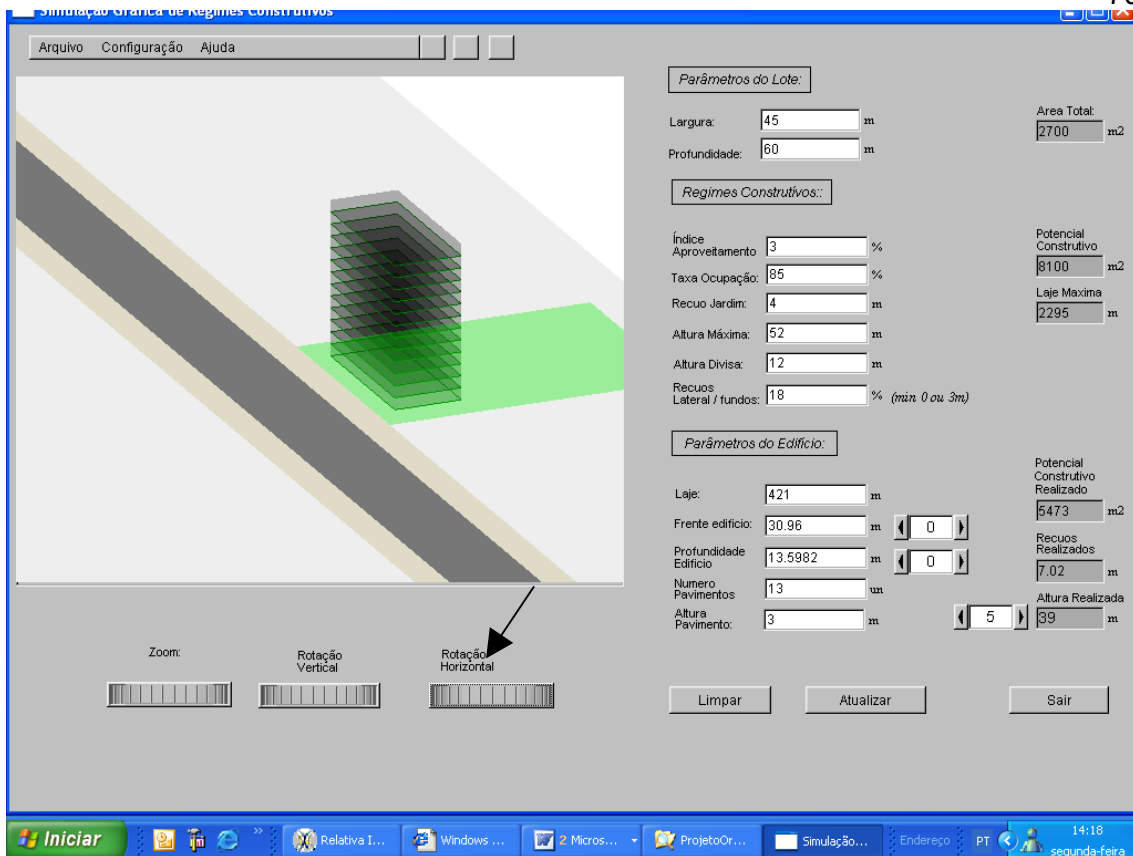


Figura 38. Imagem rotacionada com o botão “Rotação Horizontal”.

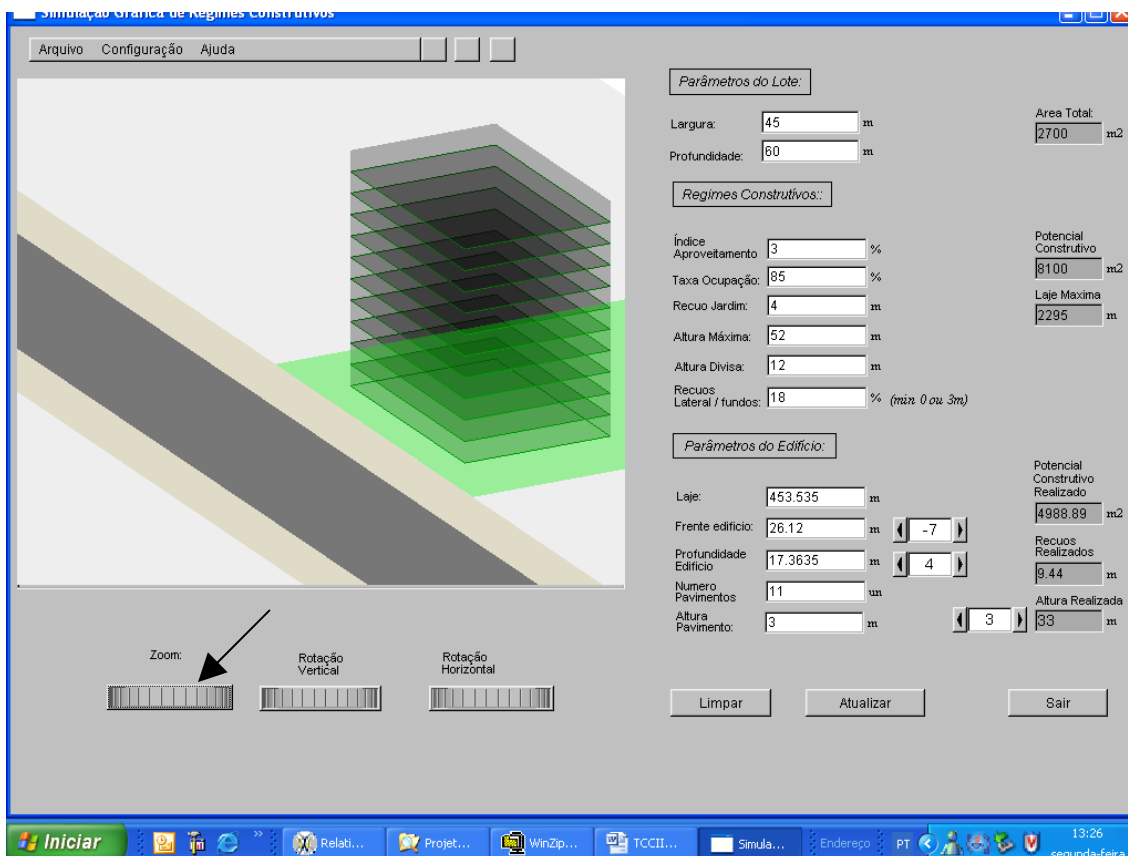


Figura 39. Imagem aumentada com o botão “Zoom”.

4.4 Validação da Ferramenta

A forma aplicada para avaliação e validação da ferramenta desenvolvida visa mencionar o aspecto acadêmico da ferramenta. Para tanto foi realizado por um grupo de 6 alunos do Curso de Arquitetura e Urbanismo do Uniritter, os quais encontram-se cursando o 9º ou o 10º semestres.

A ferramenta foi apresentada aos sujeitos avaliadores e solicitou-se que estes utilizassem-na para algumas tarefas relativas aos regimes contrutivos. Após a sessão de uso, os sujeitos preencheram um questionário foi a através do emprego de um questionário (Ver Anexo C). A aplicação foi coordenada pelo Professor e Mestre Julio Celso Vargas e as respostas coletadas são apresentadas na Tabela 8.

TABELA 8: Respostas obtidas junto ao questionário de validação.

PERGUNTAS	Sim	Em parte	Não
De um modo geral, você considera que a ferramenta atendeu o propósito para o qual foi criada?	6	0	0
Você considera que a interface gráfica é clara e de fácil entendimento?	5	1	0
Em relação aos cálculos realizados, você considera que os resultados foram consistentes e de acordo com os regimes construtivos oferecidos como entrada de dados?	6	0	0

Além destas 3 questões, foi apresentada uma quarta pergunta: “Você sentiu necessidade de algum tipo de interação não oferecida pela ferramenta?”. Em relação a esta, observa-se que grande parte do grupo sugeriu a funcionalidade ligada à rotação vertical da cena. A partir de então, tal funcionalidade foi agregada à ferramenta, procurando ajustá-la à necessidade dos sujeitos.

Foi sugerido, também pelos sujeitos, a possibilidade do salvamento e da exportação dos resultados obtidos com a ferramenta de modo que estes pudessem ser utilizados em outro programa específico para projetos arquitetônicos, como uma ferramenta CAD por exemplo.

Nota-se que, de um modo geral, a ferramenta foi bem aceita e atendeu os objetivos a que foi proposta.

CONCLUSÃO

Neste trabalho, foi realizado o estudo de conceitos das áreas da Computação Gráfica, da Interação Humano-Computador, do Projeto Arquitetônico bem como das normas de construção e ocupação do solo presentes no Plano Diretor Municipal e outros assuntos relevantes ao desenvolvimento do sistema proposto.

Considerando o foco principal do sistema ser o projeto arquitetônico, um assunto não tratado no curso de Sistemas de Informação, fez-se necessária a busca da informação através de pesquisa em livros da área da Arquitetura e Urbanismo. Baseado nestes estudos, foi desenvolvida a ferramenta de simulação gráfica de regimes construtivos que foi testada e avaliada por um grupo de alunos do curso de Arquitetura e Urbanismo do Centro Universitário Ritter dos Reis. Estes avaliaram a respectiva ferramenta com relação à interface, ao seu propósito, à facilidade de uso e aos resultados e, também, deixaram sugestão de melhoras.

Analisando as respostas dos sujeitos avaliadores, o trabalho proposto atingiu seu objetivo geral de desenvolver uma ferramenta gráfica voltada à área da Arquitetura e Urbanismo no que tange projetos de edificações. Assim, a ferramenta permite a comunicação entre os indivíduos diretamente atuantes no ciclo de vida da edificação de forma mais eficiente e, também, especulações numéricas e morfológicas dos regimes construtivos de forma rápida e interativa.

Como sugestão para trabalhos futuros, indica-se a implementação de um banco de dados permitindo que a ferramenta ofereça ao usuário os parâmetros urbanísticos de uma determinada zona a critério dele bem como salvamento dos dados e exportação dos mesmos para a plataforma web e para outras ferramentas CAD.

BIBLIOGRAFIA

ALMEIDA, Rodrigo Rebolças de: **Model-View-Controller**. Data da publicação 2005.

Disponível em: <<http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/arqu/mvc/mvc.htm>>. Último acesso: abril de 2007.

ARNAUT, Dagoberto: **Oficina de Programação. Publicado** em: <<http://www.arnaut.eti.br/op/HelpDesk.htm>>. Último acesso: maio de 2007.

ARQUITETURA. Disponível em: <<http://www.arq.ufsc.br/graduacao.php>>. Último acesso: abril de 2007.

AUTODESK. Disponível em:<www.autodesk.com>. Último acesso: maio de 2007.

AZEVEDO, E; CONCI, A. **Computação Gráfica: Teoria e Prática**. Editora Campos. 2003.

BALDEVI, Fabianne: **O Potencial Blender**. Disponível em: <http://gufsc.das.ufsc.br/tiki-download_file.php?fileId=10>

BARRETO, Marcos E. **Programação Orientada a Objetos em C++**. Disponível em:<<http://www.inf.lasalle.tche.br/~barreto/cursoC++/index.htm>>. Último acesso: maio de 2007.

BICHO, Alessandro L.; JR, Luiz G, da Silveira,; CRUZ, Adailton J.A da,; RAPOSO, Alberto B,. **Programação Gráfica 3D com OpenGL, Open Inventor e Java 3D.** Disponível em: <http://www.tecgraf.puc-rio.br/publications/artigo_2002_programacao_grafica_3d.pdf>. Último acesso maio de 2007.

BRASIL. **Ministério das Cidades – Planos Diretores.** Disponível em <<http://www.cidades.gov.br>>. Último acesso: maio de 2007.

BRITO, Allan. **Blender 3D – Guia do Usuário.** Porto Alegre, editora: Novatec ,2006.

BURBECK, Steve: **Applications Programming in Smalltalk-80™: How to use Model-View-Controller (MVC).** Disponível em: <<http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html>>. Último acesso: abril de 2007.

CARDOSO, Antônio; SOUSA, Antonio Augusto. **Projeto Cassilde – Design de Iluminação.** Disponível em: <http://www.dei.isep.ipp.pt/~acc/cassilde/>. Último acesso: junho de 2007.

COHEN, Marcelo; MANSSOUR, Isabel Harb: **OpenGL Uma Abordagem Prática e Objetiva.** Porto Alegre, Editora: Novatec. 2006.

COHEN, Marcelo; MANSSOUR, Isabel Harb: **Introdução à Computação Gráfica:** Data de publicação 2006. Disponível em: <<http://www.inf.pucrs.br/~manssour/Publicacoes/TutorialSib2006.pdf>>

COSTA, Antonio: **Tópicos Avançados em Computação Gráfica.** Disponível em < <http://www.dei.isep.ipp.pt/cg/Topicos%20avancados.pdf>>. Último acesso em maio de 2007.

CORBUSIER, Lê: **Precisões sobre um Estado Presente da Arquitetura e do Urbanismo**. Cosac & Naify, 2004.

CORONA, Eduardo; LEMOS, Carlos A.C. **Dicionário da arquitetura brasileira**. Edart. São Paulo, 1972.

COVAS, Nelson, BELK, Abram. **Tecnologia e Qualidade em Sistemas, Revista Qualidade na Construção** - SindusCon/SP - nº 19 - Ano II - 1999.

DATA CAD. Disponível em: <<http://www.datacad.com.br/>>. Último acesso: maio de 2007.

Desenho Arquitetônico. Disponível em: <<http://pt.wikipedia.org/wiki/>>. Último acesso: abril de 2007.

DEV-C++. Disponível em: < <http://pt.wikipedia.org/wiki/Dev-C++>>. Último acesso: abril de 2007.

DEV C++. BloodshedSoftware. Disponível em: < <http://www.bloodshed.net/devcpp.html>>. Último acesso em maio de 2007

DUFFY, Alexandra; COLLINS, Susan. **O Guia do Usuário do VectorWorks**. Nemetschek N.A. Inc., 7150 Riverwoode Drive, Columbia MD, 21046, USA. Data da publicação 2004.

FAGUNDES, Themis; SCHLEMMER, Eliane; MARASCHIN, Clarice: **Cidades Virtuais**. Disponível em: <http://cumincades.scix.net/data/works/att/sigradi2004_057.content.pdf>. Último acesso: abril de 2007.

FERREIRA, Sergio Leal: “**Proposta de ampliação do modelo IFC com a contribuição do IES LM-63: A luminária no ciclo de vida da Edificação.**”

FILHO, Clerton Ribeiro de A; MONTEIRO, Théo Alves; FECHINE, Joseana Macêdo: **Proposta De Um Sistema Para Gerência Informatizada Para O Programa Pré-Vestibular Solidário**. 2005. Disponível em: <http://www.dsc.ufcg.edu.br/~pet/Artigos/ARTIGO_PREVESTIBULAR.pdf>. Último acesso: abril de 2007.

FLTK 1.1.7 Programing Manual. Disponível em:< <http://www.fltk.org/doc-1.1/fltk.pdf>>Último acesso em maio de 2007.

FOLEY, J.D. **Computer Graphic Principles and Praticce**. Addeeson- Wesley, 1990.

FORMZ. Disponível em:<<http://www.formz.com/>>.Último acesso: maio de 2007.

Free-CAD programs. Disponível em:<Free-CAD programs>Último acesso: junho de 2007.

FREITAS, Carla M.D.S. **Programação com OpenGL**. Disponível em: <<http://www.inf.ufrgs.br/~carla/OpenGL/>>. Último acesso maio de 2007.

Fundamentos do UML. Disponível em:< http://docs.kde.org/stable/pt_BR/kdesdk/umbrello/uml-basics.html >. Último acesso em maio de 2007.

GOMES, Jonas; VELHO, Luiz. **Computação Gráfica: Volume 1**, IMPA/SBM, 1998.

GRAPHISOFT. Disponível em:<<http://www.graphisoft.com/>> Último acesso: maio de 2007.

GRAZZIOTIN, Pablo Colossi: **Técnicas de Incorporação de Controle de Acesso à Luz Solar em Modelos Computacionais de Edificações**. Data da publicação: março de 2003.

HENSGEN, Paul: **Manual do Umbrello UML modeller**. Disponível em <http://docs.kde.org/stable/pt_BR/kdesdk/umbrello/>. Último acesso em 2007).

KARTHIK, Ramani: **Programa encontra desenhos CAD a partir de rascunho. Da redação 28/09/2005**. Disponível em:<<http://www.inovacaotecnologica.com.br/noticias/noticia.php?artigo=010170050928>> Último acesso maio de 2007.

KERRY, H. T. (1997). **Planejamento de processo automático para peças paramétricas**. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo. Disponível na biblioteca da EESC - USP.

KÓS, José Ripper, BARKI, José, SEGRE,Roberto, BORDE, Andréa, BOAS, Naylor Vilas: **Investigação Digital dos Projetos** do Mesp – A Busca dos Vestígios do Modernismo Braisleiro. Disponível em: <http://cumincades.scix.net/data/works/att/fa3a.content.pdf>. Último acesso: Maio de 2007.

LARMAN, Craig. **Utilizando UML e padrões: uma introdução a análise e projeto orientados a objetos**. Porto Alegre: Bookamn.2000.

LEDER, Solange Maria; PEREIRA, Fernando O.R; CLARO, Anderson; RAMOS, Marcela G.: **Análise da Legislação Urbanística Através de Critérios de Insolação e Luz Natural**. Disponível em (http://www.fag.edu.br/professores/michele/2007/CONFORTOII2007.1/Fichas/ENTAC2006_0504_513.pdf). último acesso: maio de 2007.

MACIEL, Fabiano: **Iniciando em C++ - O que é C++**. Disponível em:<http://www.dotnetraptors.com.br/start/artigos/artigos_cpp/1294.aspx> ultimo acesso: maio de 2007.

MACORATTI, José Carlos. **Padrões de projeto- O modelo MVC**. Disponível em: <http://www.macoratti.net/vbn_mvc.htm>. Último acesso: abril de 2007.

MANSSOUR, Isabel: **Introdução a OpenGL**. Disponível em: <<http://www.inf.pucrs.br/~manssour/OpenGL/Introducao.htm>>. Último acesso: maio de 2007.

MANSSOUR, Isabel: **Linguagem de Programação C++**. Disponível em: <<http://www.inf.pucrs.br/~manssour/LinguagemC++/Introducao.pdf> >. Último acesso: maio de 2007.

MASON, WOO; SHREINER, DAVE; ANGEL, ED; An **Interactive Introduction to OpenGL Programming**, SIGGRAPH 1999; Los Angeles

MEDEIROS, Givaldo, ANELLI, Renato, BARBATO, Roberto. **Arquitetura, Cidade e Paisagem**. Disponível em: <<http://www.saplei.eesc.usp.br/projeto3/index.html>> Último acesso em abril de 2007.

MONTECLARO, César; JUNIOR, Carlos Eugênio. **O ser arquiteto-urbanista**. 2.ed. São Paulo, 1998.

NETO, Kanji Hara; NADALETE, Lucas G.; GENNARI, Fabio A.; FREITAS, Antonio A. Carneiro de: **Desenvolvimento de Sistemas Web utilizando arquitetura em Três Camadas e Applets**. Disponível em: <<http://inf.unisul.br/~ines/workcomp/cd/pdfs/2905.pdf>>. Último acesso: abril de 2007.

OPENGL. Disponível em: <<http://www.opengl.org>>. Último acesso: março de 2007.

PAPA, Renata Pietra: **Computação Gráfica: arquitetura ganha um aliado.** Data da publicação: 26 de maio de 2006. Disponível em: <<http://www.odebate.com.br>>. Último acesso: Março de 2007.

PENEAU, Jean Pierre: – **Nuevos instrumentos de gestion y de concepcion del espacio urbano. In Nuevas Tecnologias en Urbanismo.** Revista Ciudad y Territorio. No. 84. Madrid. 1990.

Portal dos Cursos de Graduação do Centro Universitário Ritter dos Reis. Disponível em: <<http://www.uniritter.edu.br/w2/arquitetura>>. Último acesso: Abril de 2007.

RAD: **Radiance – Syntetic Imaging System. 1997.** Disponível em: <<http://radsite.lbl.gov/radiance/HOME.html>>. Último acesso: abril de 2007.

REIS, Paula Gonçalves: **Laboratório de Arquitetura e Urbanismo On-Line: Uma Proposta de Ação de Aprendizagem.** Florianópolis, 2002.

RHINO3D. Disponível em:<<http://www.rhino3d.com/>>.Último acesso: maio de 2007.

ROGERS, D.; ADAMS J. **Mathematical Elements for Computer Graphics.** 2.ed. New York, NY: McGraw – Hill, 1990.

SANDRO, Santos Andrade. **Orientação a Objetos com Standart C++.** Disponível em < <http://www.gnu.frb.br/documentacao/tutorial/material-treinamento-standart-c/treinamento-standart-c-pdf>>. Último acesso maio de 2007.

SANTIAGO, Alina G., FEITOSA, Flávia F., BIANCHI, Miguel F., ROSA, Marcelo M.: **A CONSTRUÇÃO DE UM SISTEMA DE INFORMAÇÕES GEOGRÁFICAS: UMA ABORDAGEM DIDÁTICO-PEDAGÓGICA.** Disponível

em: <http://www.dpi.inpe.br/~flavia/pessoal/files/Santiago_enepea.pdf>. Último acesso maio de 2007.

SANTOS, Roberta Nascimento Saint Clair dos. Petrópolis: **Quadro Legal da Ocupação do Solo da Cidade Imperial**. Data da publicação maio. 2004. Disponível em: <<http://www.vitruvius.com.br/minhacidade/mc097/mc097.asp>>. Último acesso: abril de 2007.

SHREINER, Dave; WOO, Mason; NEIDER, Jackie; Davis, Tom: **OpenGL Programming Guide: the official guide to learning OpenGL**, version 1.4. 4th ed. Reading, Massschetts: addison Wesley, 2004.

SILVA, Cristiano Jeferson Cipriano da: **Simulação Termoenergética de um Sistema de Ar Condicionado para Comparação dos Resultados Medidos e Simulados**. Porto Alegre, 2003.

SMP:Secretaria do Planejamento Municipal de Porto Alegre. Disponível em:< <http://www2.portoalegre.rs.gov.br/spm/default.php>>. Último acesso: maio de 2007.

SILVA, Isabel C. Siqueira da. **Oficina de Programação com OpenGL**. Disponível em:< http://www.sis.ritterdosreis.br/atividade_complementar_opengl/>. Último acesso: maio de 2007.

SILVA,Carlos Roberto Filho: **Introdução a Compução Gráfica digitalizada de Mapas**. Disponível em < http://www.ige.unicamp.br/site/aulas/91/AULA_1_2004_PRATICA.ppt#256,1,Sli de 1>

SOBRINHO, Marcionilio Barbosa: **Tutorial de Utilização de OpenGL**. Data de publicação 2003.

Tutorial da linguagem C++. Disponível em:

<<http://www.cplusplus.com/doc/tutorial/>>. Último acesso: abril de 2007.

Tutorial FLTK. Disponível em: <<http://www.fltk.org>>. Último acesso: abril de 2007.

Tutorial FORMZ. Disponível em: < <http://www-personal.umich.edu/~emdanat/Tutorials/FormZ/index.html> > . Último acesso: junho de 2007.

UML - Unified Modeling Language. Disponível em: <http://www.uml.org>. Último acesso: abril de 2007.

UML. Disponível em: <<http://pt.wikipedia.org/wiki/UML>>. Último acesso: abril de 2007.

VECTORPRO. Disponível em: <<http://www.vectorpro.com.br/>>.Último acesso: maio de 2007.,

WRIGHT, Richard S.Jr.; SWEET, Michael. **OpenGL SuperBible**. 2nd ed. Indianápolis, Indiana: Waite Group Press, 2000.

WIKIPÉDIA. Fluid: disponível em:< <http://pt.wikipedia.org/wiki/FLUID>> ultimo acesso: maio de 2007.

ANEXO A – Código que realiza os calculos dos resultados

No quadro abaixo mostrarei algumas linhas do código onde implemento cálculos de todos os resultados oferecidos pela ferramenta usando a linguagem C++. (Ver Quadro N°)

```

//*****Calculo Da Area Do Lote
float Janelaopengl::CalculoAreaTot(){
    areaTot= (largLote*profLote); // AREA
TOTAL DO LOTE
    return areaTot;
}
//*****Calculo Do Potencial
Construtivo*****
float Janelaopengl::CalculoPotConst(){
    po=(areaTot*indAprov);
//POTENCIAL CONSTRUTIVO
    return(po);
}
//*****
**
float Janelaopengl::CalculoTaxaOcupacao(){
    float taxaOcup_aux;
    taxaOcup_aux=((taxaOcup*po)/100); // TAXA DE
OCUPAÇÃO
    return taxaOcup_aux;
}
//*****
**
float Janelaopengl::CalculaLajeMaxima(){
    LajeMax= (areaTot*taxaOcup)/100; // LAJE
MÁXIMA
    return LajeMax;
}
//*****
**
float Janelaopengl::CalculoPotConstRealizado(){
    poRealizado=laje*numpav;
    if (poRealizado==0){poRealizado=po;} //
POTENCIAL REALIZADO
    return (poRealizado);
}

```

```

    }
    //*****
float Janelaopengl::CalculaLaje() {
    if ((laje==0)|| (laje!=0)) {
        if ((frentEdi!=0) && (profEdi!=0)) {
            laje=frentEdi*profEdi; } //
LAJE
            else if(poRealizado!=0) {
                laje=poRealizado/altEdi; } //
LAJE
            else {laje=LajeMax;}
//LAJE
        }
        if (laje<0){laje=0;} //LAJE
        return laje;
    }
//*****Calculo Da Frente Do
Edificio*****
float Janelaopengl::CalculoFrentEdi() {
    if (laje!=0){frentEdi=laje/profEdi;}
    else {frentEdi=largLote-(2*recMetro);}
//FRENTE EDIFICIO
        if(frentEdi>(largLote-
(2*recMetro))){frentEdi=largLote-(2*recMetro);}
        if(frentEdi<0){frentEdi=0;} //FRENTE
EDIFICIO
    return (frentEdi);
}
//*****Calculo Do Lado Do Edificio*****
float Janelaopengl::CalculoProfEdi() {
    if ((frentEdi==0) && (laje==0)){profEdi=0;}
    else profEdi=laje/frentEdi; // PROFUNDIDADE
EDIFICIO
        if(profEdi>(profLote-
(recMetro+recuJardim))){profEdi=profLote-
(recMetro+recuJardim);} //PROFUNDIDADE EDIFICIO
        if(profEdi<0){profEdi=0;}
//PROFUNDIDADE EDIFICIO
    return (profEdi);
}
//*****Calculo Da Altura Do Edificio*****
float Janelaopengl::CalculoAltEdi() {
    if ((numpav!=0)&& (altPav!=0)) {
        altEdi=numpav*altPav; //ALTURA DO
EDIFICIO
    }
    else {
        if(poRealizado!=0) {

```

```

        altEdi=(poRealizado/laje);
//ALTURA DO EDIFICIO
        altEdi=floorf(altEdi/altPav);
        altEdi=altEdi*altPav;} //ALTURA
DO EDIFICIO
        else {
            if (laje!=0){
                (altEdi=po/laje);
                altEdi=floorf(altEdi/altPav)
;
                altEdi=altEdi*altPav;}
//ALTURA DO EDIFICIO
        else {
            (altEdi=po/LajeMax);
            altEdi=floorf(altEdi/altP
av);
            altEdi=altEdi*altPav;} }
//ALTURA DO EDIFICIO
        }
        // if (altEdi>altMax){ altEdi=altMax;}
//ALTURA DO EDIFICIO
        return altEdi;
    }
    /*******
**
    float Janelaopengl::CalculoAndares() {
        if((altEdi!=0)&&(altPav!=0)){
            numpav=altEdi/altPav; // NUMERO DE
PAVIMENTOS
            numpav=floorf(numpav);}
        return numpav;
    }
    /*******calculo do recuo em metros*****
    float Janelaopengl::CalculoRecuoRealizado() {
        if ( altEdi>altDivi){
            recMetro=(recuos*altEdi)/100;
//RECUOS REALIZADOS
            if (recMetro<3.0){recMetro=3.0;}
//CALCULADOS EM METROS
        }
        if (altEdi<altDivi){
            recMetro=0; }
        return(recMetro);
    }

```

ANEXO B – Código realizado pelo evento “Atualizar”

Abaixo mostrarei o código que implementa este evento na classe Controle. A Classe Controle tem, além deste evento, outros 11 eventos implementados.

```
/**
**
void Controle::Liga3() {
    //verifica se ocorreu algum preenchimento
    if (laje->value() !=0) {flags2=true;} //
laje
    if (frenteEd->value() !=0) {flags2=true;} //
frente edificio
    if (profundidadeEd->value() !=0) {flags2=true;} //
profundidade do edificio,
    if (numPav->value() !=0) {flags2=true;} //
numero de pavimentos
    if (altPav->value() !=0) {flags2=true;} //
altura do pavimento
    if (flags1=true) { // caso tenha algum valor
preenchido verifica os outros
        //verifica PARAMENTROS DO LOTE
        ImagemEdificio->VerificaParametros(8);
        //retorna 0 se preenchidos ou 1 se não
        teste=ImagemEdificio->GetResulta();}
    if (flags2=true) {
        // verifica REGIMES CONSTRUTIVOS
        ImagemEdificio->VerificaParametros(9);
        teste=ImagemEdificio->GetResulta();}
        //retorna 0 se preenchido ou 1 se não
        if(teste==1) {

            TrataEventos(eBtlimpaParametros); // caso
tenha retornado um limpa todos os campos que por ventura
tenha sujeira
        }
    else{
        // variaveis auxiliares inicializadas com os
valores de entrada da interface
        // valor de entrada da laje
        float auxlaje1 = laje->value();
    }
}
```

```

//variavel que indica se foi preenchido o campo
laje
float totlaje = 0;
//valor de entrada da frente edificio
float auxfrenteEdificio1 = frenteEd-
>value();
// variavel que indica se foi ou não
preenchido o campo frente
float totfrente = 0;
//valor de entrada da profundidade edificio
float auxprofundidadeEdificio1 =
profundidadeEd->value();
//variavel que indica se foi ou não
preenchido o campo profundidade edificio
float totprofundidade = 0;
//variavel de entrada do numero de
pavimentos
float auxnumpavimentos1 = numPav->value();
//variavel que indica se foi ou não
preenchido o campo numero pavimentos
float totnumpav = 0;
//variavel de entrada d altura de pavimentos
float auxalturapavimentos1 = altPav-
>value();
//variavel que indica se foi ou não
preenchido o campo altura pavimento
float totaltupav = 0;
//variaveis auxiliares
auxlaje = 0; //auxiliar para laje
auxfrenteEdificio = 0; //auxiliar para
frente edificio
auxnumpavimentos = 0; //auxiliar para
numero de pavimentos
auxalturapavimentos = 0; // auxiliar para
altura pavimentos
auxPO = 0; // auxiliara para
Potencial construtivo
if(auxlaje!=0){ // se campo laje foi
preenchido entao
//verifica se laje não
ultrapassou laje máxima
ImagemEdificio->VerificaParametros(1);
teste2=ImagemEdificio->GetResulta2();
if(teste2==1){ totlaje=0;}
else totlaje=1; // se laje for permitida
temos um valor
}
if (auxfrenteEdificio1!=0){ //se campo
frente edificio foi preenchido

```

```

//verifica se campo válido
ImagemEdificio->VerificaParametros(2);
teste2=ImagemEdificio->GetResulta2();
    if(teste2==1){totfrente=0;}
    //se valido temos um valor
    else totfrente=1;
    }
    if (auxprofundidadeEdificio1!=0){// se
campo profundidade foi preenchido
        // verifica se campo válido
ImagemEdificio->VerificaParametros(3);
teste2=ImagemEdificio->GetResulta2();
    if(teste2==1){totprofundidade=0;}
    //se valido temos um valor
    else totprofundidade=1;
    }
    if (auxnumpavimentos1!=0){//se campo
numero pavimentos foi preenchido
        //verifica se campo válido
ImagemEdificio-
>VerificaParametros(6);
        teste2=ImagemEdificio->GetResulta2();
        if(teste2==1){totnumpav=0;}
        //se valido temos um valor
        else totnumpav=1;
        }
        if (auxalturapavimentos1!=0){//se campo
altura do pavimento foi preenchido
            //verifica se campo válido
ImagemEdificio-
>VerificaParametros(6);
            teste2=ImagemEdificio->GetResulta2();
            if(teste2==1){totaltupav=0;}
            //se valido temos um valor
            else totaltupav=1;
            }
            // se campo frente foi preenchido e
profundidade não
                if (( totfrente!=0.0)  &&
(totprofundidade!=1.0) ){
                    // solicita calculo da profundidade
para janelaopengl (ImagemEdificio)
                    //e envia para o campo da interface
profundidadeEd->value(ImagemEdificio-
>CalculoProfEdi());
                }

// se campo frente não foi preenchido mas
a profundidade sim

```

```

    if ((totfrente==0) && (totprofundidade==1)){
        // solicita calculo da frente
para janelaopengl(ImagemEdificio)
        //e envia para o campo da
interface
        frenteEd->value(ImagemEdificio-
>CalculoFrentEdi());
    }
    //se campo frente e profundidade foram
preenchidos
    if ((totfrente==1)&& (totprofundidade==1)){
        // solicita calculo da laje para
janelaopengl(ImagemEdificio)
        //e envia para o campo da
interface
        laje->value(ImagemEdificio->CalculaLaje());
    }
    //se campo numero de pavimentos foi
preenchido
    if (totnumpav==1){
        //altura não foi preenchido
        if (totaltupav==0){
            //solicita para
janelaopengl o valor padrão
            altPav->value(ImagemEdificio->GetAtlPav()); }
            altEdifi->value(ImagemEdificio-
>CalculoAltEdi());
        //solicita o calculo dos
recuosMetro
        ImagemEdificio->CalculoRecuoRealizado();
        //solicita o calculo do limite do
terreno
        resulRec->value(ImagemEdificio->GetLimit());
        //se frente não tenha sido
preenchida solicita calculo
        if (totfrente==0){frenteEd-
>value(ImagemEdificio->CalculoFrentEdi());}
        //se laje não tenha sido preenchida
solicita calculo
        if (totlaje==0){laje->value(ImagemEdificio-
>CalculaLaje());}
        //se profundidade edificio não
tenha sido preenchido solicita calculo
        if (totprofundidade==0){profundidadeEd-
>value(ImagemEdificio->CalculoProfEdi());}
        //calcula o Potencial realizado -
solicita para a Janelaopengl
        Porealizado->value(ImagemEdificio-
>CalculoPotConstRealizado());

```

```

//busca valor do Potencial construtivo preenchido
pelo usuário
    PC->value(ImagemEdificio->GetPo());
    //verifica se potencial realizado
excedeu o potencial construtivo
    if ((POrealizado->value())>(PC->value())){
        cor=1;}
    else cor=0;
    // se excedeu o potencial realizado
aparecerá com a cor vermelha.
    POrrealizado->textcolor(cor);
    ImagemEdificio->Refresh();
    //ImagemEdificio-
>VerificaParametros(2);
    }
    //se numero de pavimentos não foi
preenchido
    if (totnumpav==0){//2
        //verifica se frente e profundidade
tambem não foram preenchidos
        if ((totfrente==0)&&(totprofundidade==0)){
            //se não foram mensagem
deve ser preenchido
            //pelo menos 2 campos
aparecerá
            ImagemEdificio->VerificaParametros(11);}
        else{ // verifica se laje e
frente não foram preenchidos
            if ((totlaje==0) && (totfrente==0))
                //caso não tenham sido
mensagem de pelo menos 2 campo
                //deve ser preenchido
aparecerá
                {ImagemEdificio->VerificaParametros(11);}
                else {//verifica se laje e
profundidade não foram preenchidos
                    if ((totlaje==0) &&(totprofundidade==0))
                        //se não foram mensagem
de pelo menos 2 campos
                        //devem ser preenchidos
aparecerá
                        {
                            ImagemEdificio->VerificaParametros(11);}
                        else{
                            //caso tenham
sido preenchido
                            if(totlaje==0){
                                if ((totfrente!=0)
&&(totprofundidade!=0)){

```

```

        //solicita calculo da laje caso tenha frente e
profundidade
        laje->value (ImagemEdificio->CalculaLaje ());
    }

        //solicita o
valor da altura pavimento dada
        altPav->value (ImagemEdificio->GetAtlPav ());
        //solicita
calcula da altura do edificio
        altEdifi->value (ImagemEdificio-
>CalculoAltEdi ());
        //solicita
calcula do numero de paviemntos
        numPav->value (ImagemEdificio-
>CalculoAndares ());
        //solicita
calcula do potencial construtivo
        Porealizado->value (ImagemEdificio-
>CalculoPotConstRealizado ());
        //auxilia recebe resultado
do potencial realizado da interface
        auxPorealizav
el=Porealizado->value ();
        //auxiliar recebe resultado do
potencial construtivo da interface
        auxPO=PC->value ();
        //compara o
pontencial construtivo com o realizado
        if ((auxPorealizavel)>(auxPO)) {
            //caso o realizado excedeu o construtivo
aparecerá vermelho o realizado
        }
        Pore
alizado->textcolor (1);
        //calcula o
recuo realizado
        ImagemEdificio->CalculoRecuoRealizado ();
        //calcula o
limite para o terreno
        resulRec->value (ImagemEdificio->GetLimit ());
        //se frente não
tenha sido preenchido calcula
        if (totfrente==0) { frenteEd-
>value (ImagemEdificio->CalculoFrentEdi ()); }
        //sa profundiade
não tenha sido preenchido calcula
        if (totprofundidade==0)
        { profundidadeEd-
>value (ImagemEdificio->CalculoProfEdi ()); }

```

```
//se laje não tenha sido preenchida calcula.
    if (totlaje==0){laje->value(ImagemEdificio-
>CalculaLaje());}
    }
    }
} //1
} //2
//solicita imagem atualizada.
ImagemEdificio->Refresh();
}
}
```

ANEXO C – Questionário aplicado aos sujeitos avaliadores

Avaliação –Simulação Gráfica de Regimes Construtivos

Idade:.....

Semestre do curso:.....

Em relação ao uso da ferramenta, deixe sugestões a respeito De cada um dos itens a seguir:

- 1-) De um modo geral, você considera que ela atendeu o propósito para o qual foi desenvolvida?
- 2-)Você considera que a interface gráfica é clara, de fácil entendimento?
- 3-)Você sentiu necessidade de algum tipo de interação não oferecido pelo ferramenta?
- 4-)Em relação aos cálculos realizados, voce considera que os resultados foram consistentes e de acordo com os regimes construtivos oferecidos como entrada de dados?