PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GERAÇÃO PROCEDURAL DE AMBIENTES VIRTUAIS SEMÂNTICOS

Fernando Pinho Marson

Tese apresentada como requisito à obtenção do grau de Doutor em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientadora: Profª Drª Soraia Raupp Musse

Porto Alegre
2012

Ficha Catalográfica

M373g Marson, Fernando Pinho

Geração Procedural de Ambientes Virtuais Semânticos / Fernando Pinho Marson . – 2012.

72 f.

Tese (Doutorado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientadora: Profa. Dra. Soraia Raupp Musse.

1. Ambientes Virtuais. 2. Geração Procedural de Conteúdo. 3. Semântica. I. Musse, Soraia Raupp. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE TESE DE DOUTORADO

Tese intitulada "Geração Procedural de Ambientes Virtuais Semânticos", apresentada por Fernando Pinho Marson, como parte dos requisitos para obtenção do grau de Doutor em Ciência da Computação, Sistemas Interativos e de Visualização, aprovada em 29/08/2012 pela Comissão Examinadora:

Profa. Dra. Soraia Raupp Musse – PPGCC/PUCRS

Prof. Dr. Márcio Sarroglia Pinho - PPGCC/PUCRS

Prof. Dr. Cláudio Rosito Jung - UFRGS

Prof. Dr. Alessandro de Lima Bicho – FURG

Homologada em...../...../....., conforme Ata No. ..... pela Comissão Coordenadora.

Prof. Dr. Paulo Henrique Lemelle Fernandes
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P. 32 – sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320–3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

AGRADECIMENTOS

Meu primeiro agradecimento é para a minha orientadora Soraia, pela paciência e tempo dedicados à mim; sem ela esta tese não existiria.

Aos meus pais, Lourdes e Antonio, ao meu irmão Claudio e aos meus outros familiares por entenderem os momentos de ausência.

Ao Luciano Alfonso pela parceria e força durante todos os momentos críticos.

À Rossana Queiroz, ao Vinícius Cassol e ao Daniel Camozzato pelos bate-papos, risadas e companheirismo, sempre me incentivando e ajudando em tudo o que foi preciso.

Aos meus colegas do VHLab, que em alguns momentos foram mais que colegas (em ordem alfabética para evitar mágoas): Adriana, Anderson, Cristina, Henry, Hocevar, Júlio, Laura, Leandro, Luiz e à todos que já passaram por lá.

Ao professor Rafael Bidarra da TUDelft, pelas ideias e conversas.

Aos colegas da TUDelft, que se tornaram parte viva desta tese: Tim, Jassin, Nick, Ricardo, Ruben, Joel e Xin.

Aos colegas da Unisinos, principalmente ao Gonzaga, pela força e incentivo.

Aos professores e funcionários da PUCRS, que de alguma forma contribuíram para a minha formação.

Às agências de fomento (CAPES e CNPq) e à PUCRS, pelo apoio financeiro para a realização do doutorado e de viagens acadêmicas.

Aos não citados...

GERAÇÃO PROCEDURAL DE AMBIENTES VIRTUAIS SEMÂNTICOS

RESUMO

Este trabalho descreve um modelo computacional para a geração procedural de ambientes virtuais semânticos, os quais podem ser utilizados em jogos e na realização de simulações comportamentais com grupos e multidões de agentes virtuais autônomos. O processo de criação dos ambientes se dá a partir de informações semânticas fornecidas pelo usuário que tem como base um modelo mental abstrato. Este modelo mental serve como ponto de partida para a especificação das relações existentes entre os diversos espaços que devem ser criados no ambiente virtual. O modelo computacional proposto trabalha em duas linhas distintas: criação de ambientes residenciais (casas e apartamentos) e geração de ambientes amplos. Ambientes virtuais amplos podem ser entendidos como construções virtuais mais complexas, como *shopping centers*, aeroportos e galerias comerciais. Para cada um dos tipos de ambiente é utilizado um método distinto, melhor adequado para tratar os problemas existentes em cada situação. Após a realização das subdivisões dos espaços, a sequência de etapas volta a confluir, mantendo uma arquitetura única para a geração do ambiente virtual. Como saída são obtidos arquivos de geometria e de descrição semântica que podem ser utilizadas em diferentes aplicações nas áreas de jogos, computação gráfica e de simulação.

Palavras-chave: Ambientes Virtuais; Geração Procedural de Conteúdo; Semântica.

PROCEDURAL GENERATION OF SEMANTIC VIRTUAL ENVIRONMENTS

ABSTRACT

This work describes a computational model to generate semantic virtual environments procedurally, which can be used in games and behavioral simulations of crowds and groups of autonomous virtual agents. The process of creating environments, starts with a set of semantic information provided by the user. Such information are based on an abstract mental model, which describes relationships among the spaces that should be created. The proposed approach works in two different ways: building residential environments (i.e. houses and apartments) and also large environments. Large environments can be understood as complex virtual buildings, e.g., shopping malls, airports or train stations. For each type of environment is used a distinct method, more suitable to treat the specific environmental issues. After subdivide all spaces, the flow turns to become one again. As output we have geometry files and semantic description files that can be used in different applications as games and behavioral crowd simulations.

Keywords: Virtual Environments; Procedural Content Generation; Semantics.

LISTA DE FIGURAS

Figura 2.1 Modelo detalhado de construção. [6] 24
Figura 2.2 Detalhamento de uma geometria em um *smart building*. (a) Construção Sólida. (b) Divisão por andares. (c) Refinamento da geometria. (d) Fachadas texturizadas. [15] 24
Figura 2.3 Funcionamento esquemático do módulo de visualização. [24] 25
Figura 2.4 Planta 2D fornecida como entrada e o resultado da reconstrução tridimensional [29]. 28
Figura 2.5 Processo de geração de um *layout* interativo proposto por [21]. 29
Figura 2.6 Modelo de geração de fachada proposto por Wonka [62]. 29
Figura 2.7 Dois modelos gerados a partir da gramática *CGA Shape* [47]. 30
Figura 2.8 Processo de geração da fachada a partir de uma foto [48]. 30
Figura 2.9 Diferentes tipos de telhados gerados por Finkenzeller [19]. 31
Figura 2.10 Construção gerada pelo *framework* de Finkenzeller [19]. 31
Figura 2.11 Processo de geração do modelo proposto por Larive [38]. 32
Figura 2.12 Cidade gerada pelo projeto "*A Different Manhattan Project*" [63]. 33
Figura 2.13 Cidade gerada pelo CityEngine [53]. 34
Figura 2.14 Modificação da malha viária feita em função do crescimento da cidade [28]. . 34
Figura 2.15 *Layout* da cidade em três momentos distintos do tempo. Os níveis de cinza indicam o tipo de ocupação da construção. [28] 35
Figura 2.16 Definição das sementes de construção (esq.). Cidade gerada a partir das sementes de construção (dir.). [42] 35
Figura 2.17 Ambiente gerado pelo modelo proposto por Silveira Jr. e Musse. [12] 35
Figura 3.1 Visão geral da arquitetura proposta. 38
Figura 3.2 Visão geral do conceito de entidades apresentada em [36]. 39
Figura 3.3 Hierarquia espacial 40
Figura 3.4 Estrutura hierárquica organizada em árvore (a) e seu respectivo *treemap* (b). 42
Figura 3.5 Exemplo de um *treemap* onde os retângulos gerados no processo de subdivisão são muito diferentes de 1. 43
Figura 3.6 Passo a passo do processo de construção de um *squarified treemap* (adaptado de [8]). 43
Figura 3.7 Divisão da área total da residência em três áreas principais. 45
Figura 3.8 Grafo de conectividade gerado para uma planta baixa. 46
Figura 3.9 Passos para a criação da residência. 47
Figura 3.10 Visualização tridimensional da planta baixa representada na Figura 3.9e. . . 48
Figura 3.11 Criação do *Straight Skeleton* de um polígono. 49
Figura 3.12 Etapas de criação do corredor principal. 49

Figura 4.1 Em (A), representa-se o grafo de conectividade considerado para representação de um ambiente residencial com respectivas características e, em (B), ilustra-se o mesmo grafo após adequações para ser utilizado na produção de *pathplanning* para a simulação de agentes virtuais na residência. 53
Figura 4.2 Em (A), o grafo de conectividade e em (B) a casa utilizada para simulação de ambientes internos [41]. Em (C), apresenta-se a sua visualização no ambiente de simulação. 54
Figura 4.3 Em (A), apresenta-se o grafo ligando os cômodos da casa e em (B) a locomoção dos agentes em relação ao grafo. Em (C) e (D), apresentam-se agentes que se deslocam pela casa para realizar ações. 55
Figura 4.4 Planta baixa gerada pelo modelo e reconstruída por Tutenel [61]. 56
Figura 4.5 Planta baixa de um ambiente amplo utilizado como estudo de caso por Kraayenbrink [37]. 57
Figura 4.6 *Heap map* de uma simulação de agentes virtuais realizada por Kraayenbrink [37] utilizando o mesmo ambiente ilustrado anteriormente. 58
Figura 4.7 Captura de tela da representação tridimensional do ambiente amplo apresentado na Figura 4.5 58
Figura 4.8 Marcadores de um ambiente amplo em uma simulação de agentes virtuais especificada por Hocevar [27]. 59
Figura 4.9 Visualização 2D da simulação de agentes virtuais especificada por Hocevar utilizando o *framework* CrowdVis [7]. 59
Figura 4.10 Captura de tela da visualização tridimensional dos agentes realizando a simulação. 60

LISTA DE TABELAS

Tabela 3.1 Conexões possíveis (propostas pelo autor) entre as peças de uma residência. 46
Tabela 4.1 Ações e locais de realização definidos para serem considerados na simulação de agentes em ambientes residenciais. 54

LISTA DE SIGLAS

CAD	*Computer Aided Design*
IFC	*Industry Foundation Classes*
BIM	*Building Information Model*
GIS	*Geographic Information Systems*
UEM	*Urban Environment Model*
VUL	*Virtual Urban Life*

SUMÁRIO

Lista de Figuras 13

Lista de Tabelas 15

Lista de Siglas 17

1. INTRODUÇÃO 21
   - 1.1 Objetivo Geral . . . 22
   - 1.2 Objetivos Específicos . . . 22
   - 1.3 Estrutura da Tese . . . 22

2. FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS 23
   - 2.1 Semântica . . . 23
   - 2.2 Geração Procedural de Conteúdo . . . 26
     - 2.2.1 Plantas Baixas . . . 27
     - 2.2.2 Construções . . . 28
     - 2.2.3 Ambientes Virtuais . . . 32

3. MODELO PROPOSTO 37
   - 3.1 Visão Geral da Arquitetura . . . 37
   - 3.2 Informações Semânticas . . . 37
     - 3.2.1 Objetos Tangíveis . . . 39
     - 3.2.2 Espaços . . . 39
     - 3.2.3 *Template* e Customização . . . 40
     - 3.2.4 Instâncias de Objetos . . . 41
     - 3.2.5 Instâncias de Espaços . . . 41
   - 3.3 Módulo de Subdivisão Espacial . . . 41
     - 3.3.1 Subdivisão Espacial de Ambientes Residenciais . . . 42
     - 3.3.2 Subdivisão Espacial de Ambientes Amplos . . . 48
   - 3.4 Módulo para Geração de Informações para População Virtual . . . 50

4. RESULTADOS 53
   - 4.1 Ambientes Residenciais . . . 53
   - 4.2 Ambientes Amplos . . . 55

5. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS 61

Referências Bibliográficas 63

A. Publicações 69
A.1 Resumo em Anais de Conferência 69
A.2 Artigos em Anais de Conferências 69
A.3 Artigo em Periódico 69

B. Listagem dos Arquivos de Informações Semânticas 71

1. INTRODUÇÃO

Para que qualquer história ou estória possa ser representada é necessária a existência de um contexto, um cenário onde a mesma esteja inserida e se desenrole. Isto é válido tanto para a vida real, como para a indústria de entretenimento. Filmes e jogos digitais utilizam massivamente ambientes virtuais a fim de representar as mais lindas estórias de amor ou os mais terríveis futuros onde acontecem a dominação da humanidade por máquinas ou por zumbis. Independente do contexto, o ato de criação de um ambiente virtual é complexo, pois demanda a escolha correta dos objetos e do posicionamento dos mesmos para que exista uma sensação de coerência entre o que é representado e o que se deseja representar.

Neste sentido, a realização de animações comportamentais é ainda mais exigente, pois implica na necessidade de interação por parte dos agentes com o ambiente virtual. Os componentes do ambiente não podem apenas parecer, eles precisam ser, ou pelo menos, mimetizar os atributos e comportamentos desejados. O simples ato de caminhada de um agente requer um conhecimento sobre as áreas por onde é permitido caminhar e os obstáculos dos quais deve desviar durante a sua ação. A extração desse tipo de conhecimento deve ser realizada previamente, pois, caso contrário, acarreta no aumento da complexidade dos agentes, o que torna inviável a realização em tempo real da simulação de multidões com uma grande quantidade de agentes.

O processo de transferência de parte da inteligência dos agentes para o ambiente cria os chamados Ambientes Virtuais Inteligentes ou *Intelligent Virtual Environment* (IVE), onde o ambiente fornece aos agentes as informações por eles solicitadas. O acréscimo de informações semânticas ao modelo geométrico possibilita, entre outras coisas, que os agentes diferenciem as funcionalidades e serviços providos por um determinado espaço, reconhecendo uma construção como sendo, por exemplo, um restaurante e outra como um banco.

O problema a ser tratado nesta tese é como permitir, através de um modelo computacional, a tradução de uma ideia abstrata de um ambiente, definida em *templates*, para um modelo geométrico semântico que atenda os requisitos estabelecidos nesta ideia e que permita o seu uso em jogos e simulações com humanos virtuais autônomos. Os principais fatores agravantes deste problema são:

- a complexidade da modelagem interna e externa de construções virtuais;
- a falta de informações semânticas sobre o uso e sobre as funcionalidades dos diversos componentes das construções virtuais (portas, janelas, escadas, mobiliário, etc);
- a possível falta de coerência entre partes do ambiente geradas separadamente (interior e exterior);
- as interfaces gráficas ou textuais, que devem ser flexíveis e robustas a ponto de permitir que os ambientes possam ser criados de maneira customizada.

Para solucionar esses problemas é proposta a criação de um modelo computacional que permita a geração automática da geometria e da semântica de ambientes virtuais, através da especificação textual do ambiente feita pelo usuário. O modelo será descrito em detalhes nesta tese de doutorado.

1.1 Objetivo Geral

O objetivo principal deste trabalho é o de automatizar o processo de geração de ambientes virtuais que contenham informações semânticas que possibilitem o seu uso por agentes virtuais autônomos. Para isto foi desenvolvido um modelo computacional capaz de criar proceduralmente, ambientes internos (casas e apartamentos) e ambientes amplos (shoppings, galerias, aeroportos, dentre outros) a partir de uma definição hierárquica baseada em *templates* textuais semânticos. Esta informação semântica especifica os usos e funcionalidades de cada espaço e de cada objeto presente no ambiente virtual.

1.2 Objetivos Específicos

Especificamente, tem-se como objetivos para este trabalho:

- Estudo de técnicas de geração procedural aplicadas à criação de ambientes virtuais;
- Estudo de trabalhos relacionados ao uso de semântica em ambientes virtuais;
- Criação de uma camada semântica para representação de conhecimento, explicitando os usos e atributos dos elementos presentes no ambiente virtual;
- Desenvolvimento de um modelo computacional capaz de integrar as camadas de geometria e de semântica;
- Implementação de um protótipo para geração procedural de ambientes virtuais semânticos;
- Geração e análise de resultados.

1.3 Estrutura da Tese

A estrutura desta tese está organizada da seguinte forma: o Capítulo 2 é composto pela fundamentação teórica e pelos trabalhos relacionados relativos à criação de modelos mentais utilizados para especificar quais são as informações necessárias para modelar o ambiente virtual desejado. São apresentados os fundamentos do uso de semântica e de geração procedural, finalizando com a contextualização da pesquisa no estado da arte. O modelo proposto é apresentado no Capítulo 3 e tem seus resultados apresentados e discutidos no Capítulo 4. Considerações finais e possibilidades de trabalhos futuros são elencados no Capítulo 5.

2. FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

2.1 Semântica

Uma representação semântica, de acordo com Rishe [57], oferece uma maneira simples, natural, flexível e não redundante de especificar informações. O conceito de modelagem semântica é originário da área de Banco de Dados e de Sistemas de Informação, mas pode ser utilizado em diferentes áreas da Computação para prover uma camada de informações adicionais sobre um conceito que está sendo definido. Na área de Computação Gráfica, por exemplo, é possível adicionar uma camada semântica sobre um objeto geométrico, informando características como massa, partes que o compõem, material de cada parte, funcionalidades do objeto e assim por diante. A seguir são apresentados trabalhos que utilizam semântica em objetos, ambientes e agentes virtuais.

Kallmann [34] [32] em 1999 cria o conceito de *Smart Objects* através da adição de semântica em objetos virtuais. Estes objetos informam ao agente virtual as possibilidades de interação e como as mesmas devem ser executadas. As interações incluem ações de agarrar, puxar e rotacionar objetos. Ao se posicionar na frente de um elevador, por exemplo, um agente virtual recebe uma lista com as interações possíveis. Ao decidir apertar o botão do elevador, o agente recebe informações de como posicionar a mão para executar a ação. Com esta técnica foi possível retirar parte da inteligência dos agentes diminuindo a sobrecarga da inclusão de novos agentes no ambiente virtual.

Em [55] Peters et al. apresentam um *framework* estendido para a modelagem de interações entre agentes e objetos baseado no conceito de *Smart Objects* apresentado anteriormente por Kallmann [34]. As informações de interação são pré-programadas e utilizadas na geração de animações. Como diferencial, o modelo fornece informações que possibilitam o direcionamento do olhar do agente virtual (*gaze*) durante a animação. Este tipo de interação também pode ser aplicado a múltiplos agentes que interagem com um único objeto.

Benner et al. [6] apresentam um modelo focado na definição de características de construções para o desenvolvimento urbano. A semântica é utilizada na definição das características das partes que compõem as construções: compartimentos, passagens, aberturas e detalhes dos prédios, modelando diferentes níveis de detalhe e *design*. O modelo também inclui um módulo para geração paramétrica das construções. Visando criar uma ligação entre as ferramentas de e o *framework* apresentado, foi desenvolvido um módulo para comunicação com o formato padrão da indústria, o IFC[1]. O objetivo dessas extensões é facilitar a utilização e manutenção dos modelos gerados.

O trabalho desenvolvido por Döllner et al. [15] apresenta o conceito de prédios inteligentes (*smart buildings*), que podem ser utilizados para criação e definição de construções tridimensionais. Um *smart building* representa não só a geometria da construção, mas também a planta baixa por andar e suas partes. Além disso, o modelo provê também uma semântica associando as funcionalidades de

[1] As *Industry Foundation Classes* (IFC, ISO/PAS 16739) compõem um padrão aberto para representação de modelos de construções que englobam diferentes domínios como arquitetura, serviços e gestão da construção.

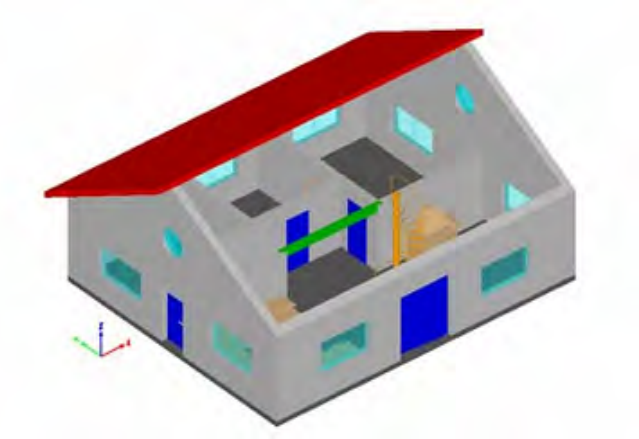


Figura 2.1 – Modelo detalhado de construção. [6]

cada parte da construção, incluindo as fachadas. O uso dos *smart buildings* facilita a modelagem, o refinamento e mesmo a redefinição das formas da construção 3D, permitindo a integração de modelos com diferentes níveis de detalhe dentro de um *framework* uniforme. A partir do conceito apresentado, é possível implementar ferramentas intuitivas para criação e manutenção de cidades virtuais tridimensionais de forma incremental. A Figura 2.2 mostra o detalhamento da geometria na criação de um *smart building*.

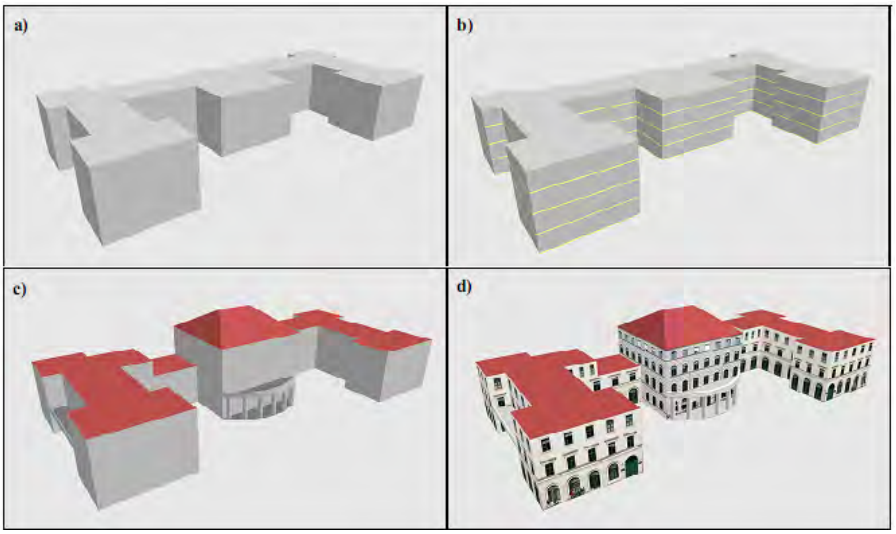


Figura 2.2 – Detalhamento de uma geometria em um *smart building*. (a) Construção Sólida. (b) Divisão por andares. (c) Refinamento da geometria. (d) Fachadas texturizadas. [15]

Hagerdorn e Döllner [24] propõem uma abordagem para visualizar e analisar informações de modelos de construções descritas dentro de cidades virtuais. Os *building information models* (BIM) permitem formalizar e representar informações detalhadas relativas ao ciclo de vida dos prédios, como material utilizado, equipamentos, dependências, usos e manutenção, além de possibilitar a realização do planejamento de resgate em situações de emergência. A proposta integra dados oriundos de GIS e dos BIM de forma distribuída, através de serviços web, integrando-os em um sistema de geovisualização 3D em tempo real. A Figura 2.3 mostra a arquitetura de visualização

implementada.

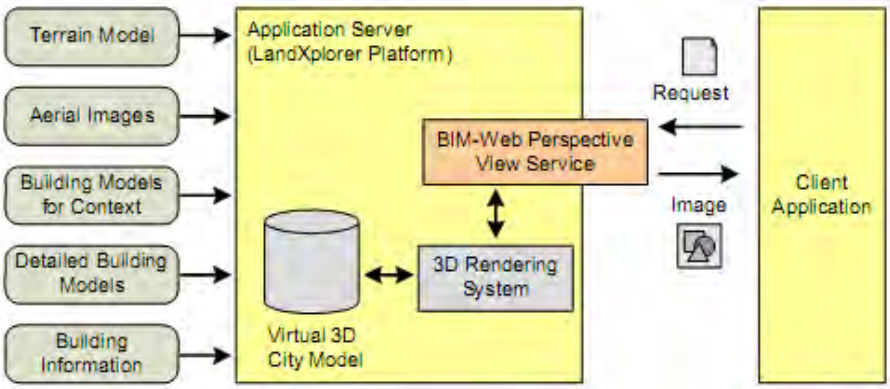


Figura 2.3 – Funcionamento esquemático do módulo de visualização. [24]

Definir apenas a parte geométrica e topológica do Ambiente Virtual não é o suficiente para tornar o mesmo apropriado para a ocupação por humanos virtuais. É necessário que este espaço possua algum tipo de informação que possibilite a interação entre os agentes virtuais e os objetos presentes no mundo. Um Ambiente Virtual que dispõe deste tipo de informação é normalmente referenciado como um Ambiente Virtual Inteligente [40], ou com um Ambiente Informado [18] ou ainda como um Ambiente Virtual Semântico [51].

Thomas e Donikian [58] propõem um modelo de Ambientes Virtuais que utilizam estruturas e informações convenientes para a realização de animações comportamentais. Usando o conhecimento disponibilizado pelo ambiente, os agentes virtuais autônomos podem se comportar como pedestres ou motoristas em um ambiente urbano complexo.

Através do conceito de *Synoptic Objects*, Badawi e Donikian [3] descrevem um ambiente informado usando objetos que contêm um resumo das interações as quais eles próprios podem ser submetidos. Estes objetos descrevem superfícies de interesse no próprio objeto e no espaço afetado pelo objeto durante a interação. Usando um conjunto de sete ações básicas, os objetos descrevem o processo de interação para que qualquer agente possa implementar as mesmas. A descrição do processo de interação é feito através de ações complexas transmitidas do objeto para os agentes. Estas ações complexas são traduzidas para um conjuntos de ações básicas em uma determinada ordem de execução, de forma a permitir que o agente cumpra a ação requisitada.

Gutierrez et al. [23] apresenta um método para representação de objetos em ambientes virtuais utilizando semântica para definir a sua aparência, as suas funcionalidades e formas possíveis de interação. Cada objeto do ambiente é entendido não apenas como uma forma tridimensional, mas como uma entidade que possui diferentes funções e formas de representação. Isto permite a alteração desses objetos, em função do contexto ou do cenário onde o mesmo se encontra.

Um modelo semântico para representação de ambiente complexos com multi-camadas é proposto por Jiang et. al. [30]. Este modelo é composto de três níveis: um nível geométrico, um nível semântico e um nível de aplicação. O nível geométrico contém um modelo tridimensional do ambiente virtual, o qual é utilizado para a visualização e para extrair informações semânticas que

irão alimentar a próxima camada de representação. Esta informação é extraída do modelo de duas formas distintas: através de um processo automático de identificação dos componentes e da leitura de marcações feitas manualmente. A camada semântica compreende um mapa estrutural (partes que compõem o ambiente), um mapa topológico (usado para armazenar e recuperar a conectividade entre diferentes regiões) e um mapa de altura. A camada final (nível de aplicação) é responsável por prover interações eficientes entre pedestres e o ambiente, usando a informação da camada semântica para gerar mapas de alto nível, como mapas de percepção de objetos e mapas de caminhamento individual pelo ambiente.

Outra possibilidade de uso de semântica em ambientes virtuais é proposta por Paiva et al. [52], onde é desenvolvido um modelo chamado *Urban Environment Model* (UEM), uma abordagem na qual é possível povoar espaços urbanos com um grande número de humanos virtuais, de forma semelhante ao que acontece no mundo real. Através desse modelo, a semântica é incluída no espaço virtual simbolizando ações cotidianas de acordo com o perfil de diferentes agentes virtuais distribuídos no tempo e no espaço. Por exemplo, em um mesmo tempo os adultos vão trabalhar enquanto crianças vão para a escola. Vários tipos diferentes de atividades podem ser considerados, como lazer, compras ou trabalho. O trabalho apresenta ainda a integração do modelo UEM com um simulador de multidões visando prover comportamentos realistas e coerentes em um ambiente urbano.

Usando o conceito de ontologia, Grimaldo et al. [22] propõem um *framework* baseado em semântica para a simulação de grupos de agentes virtuais inteligentes. Estes agentes se utilizam de informações fornecidas pelo ambiente para incrementar as relações entre agentes e objetos, bem como interações entre agentes. Uma taxonomia é utilizada para classificar os objetos interativos e suas propriedades. Outro uso de uma ontologia é no sentido de definir relações sociais entre os agentes virtuais de forma a representar decisões aceitáveis socialmente.

Kraayenbrink [37] propõe uma abordagem de multidões semânticas (semantic crowds) para que multidões de agentes virtuais projetadas para um determinado ambiente possam ser utilizadas em outro cenário sem a modificação de valores de parâmetros. Para que os agentes possam interagir de forma autônoma com qualquer cenário, as informações de uso e funcionalidade são armazenadas nos espaços e nos objetos virtuais. Desta forma, os agentes podem recuperar estas informações e planejar as suas ações de acordo com as suas metas. para facilitar a criação destas multidões, foi desenvolvido um editor interativo que trabalha com parâmetros de alto-nível, permitindo a geração de *templates* de multidões utilizáveis em diferentes cenários.

2.2 Geração Procedural de Conteúdo

Segundo Ebert [17], técnicas procedurais podem ser definidas como sendo segmentos de código ou algoritmos que especificam alguma característica de um modelo ou efeito gerado por computador. Tais técnicas têm sido utilizadas desde o início da Computação Gráfica para a geração de modelos geométricos, criação de texturas e animação de objetos e personagens. A principal característica

presente nos métodos procedurais é a abstração. Ao invés de armazenar todos os detalhes de uma determinada geometria ou animação, o conceito destas é abstraído e codificado sob a forma de um algoritmo. Cada vez que o modelo abstraído se faz necessário, o mesmo é recriado através de um procedimento computacional.

Uma vez codificado o modelo, é possível realizar a parametrização de certos elementos do mesmo, a fim de permitir a criação de objetos ou efeitos diferentes do original, mas que compartilhem características comuns. Como exemplo, podemos citar um algoritmo que codifique o conceito genérico de cadeira. Não existe apenas um modelo possível de cadeira, mas infinitas combinações de diferentes características presentes nas cadeiras: altura, material, tipos de assento e de encosto, entre outros.

A parametrização dessas características permite que se gere, a partir de um mesmo código, vários objetos que representem o conceito de cadeira. Em função da abstração e do controle paramétrico, surge a flexibilidade como característica emergente dos métodos procedurais. Tal característica facilita o trabalho dos *designers*, permitindo que os desenvolvedores de jogos digitais criem conteúdos variados em função da necessidade do momento.

Existem duas formas de pensar a geração procedural segundo Dörner et al. [16, p. 118]: o método teleológico ou o ontogenético. No modelo de geração teleológica, o objeto ou efeito a ser criado surge a partir das características e restrições definidas pelo ambiente e pelo processo de criação ao qual está sujeito. Neste caso, a criação de um determinado objeto só pode ocorrer em um dado ambiente. No caso de alteração deste ambiente, o objeto gerado refletirá as modificações do mesmo. Já na abordagem ontogenética, toda a informação necessária para a criação do objeto está codificada em uma única função geradora, por isso o ambiente não influencia a geração atual do mesmo.

Existem diversas possibilidades de classificação dos métodos procedurais, pois trata-se de um termo muito genérico. A maior parte das abordagens que produzem algum tipo de resultado de forma automática, a partir de um modelo conceitual codificado, pode ser classificada como sendo uma técnica procedural.

A seguir serão apresentados alguns trabalhos envolvendo geração procedural de plantas baixas, de construções e de ambientes virtuais completos.

2.2.1 Plantas Baixas

Para que uma construção possa ser explorada em simulações e jogos digitais por agentes virtuais ou personagens controlados pelos jogadores, é necessária a existência não só do exterior, mas também do interior da construção.

Martin [44] faz uma análise dos métodos de geração procedural aplicados à modelagem de construções e propõe um algoritmo para a geração de plantas arquitetônicas de casas residenciais. O passo inicial da abordagem é a criação de um grafo de conectividade entre as peças a serem criadas. Após estabelecer a conexão entre as peças, é feito o posicionamento usando regras de crescimento e pesos. Ao final é gerada a geometria tridimensional.

Uma abordagem para geração de interiores de prédios em tempo real é apresentada por Harn et al. em [25]. Os interiores são criados a partir de onze regras que atuam como diretrizes no processo de geração. Os prédios gerados por esta técnica são divididos em regiões conectadas por portais. O método usa uma técnica de geração tardia, na qual apenas a parte do ambiente visível no momento é gerado. Quando a região deixa de ser visível, a estrutura é removida da memória. Essa abordagem evita o uso de processamento e memória desnecessários. Uma das características do método é que o mesmo gera um ambiente persistente: todas as modificações feitas em uma dada região são armazenadas e podem ser acessadas quando necessário.

Em [29] Horna et al. propõem um método para a reconstrução da geometria e topologia de construções tridimensionais a partir de plantas arquitetônicas em duas dimensões. Informações adicionais podem ser agregadas ao modelo bidimensional visando auxiliar na reconstrução tridimensional. É possível a construção de vários andares através da repetição da mesma planta arquitetônica. A Figura 2.4 mostra uma planta baixa 2D (esquerda) reconstruída em 3D (direita).

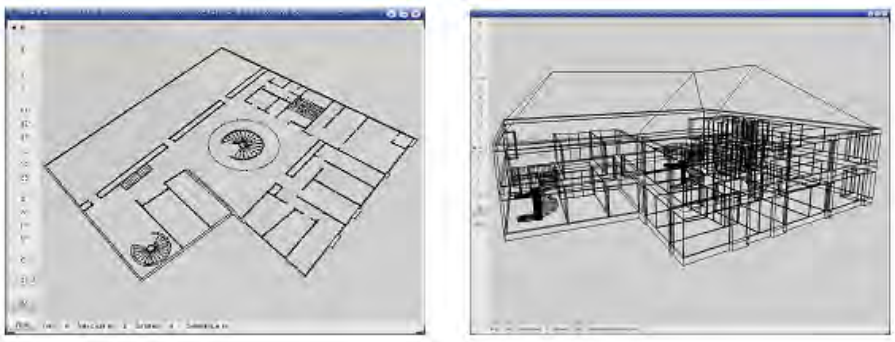


Figura 2.4 – Planta 2D fornecida como entrada e o resultado da reconstrução tridimensional [29].

Gaildrat et al. [21] criam uma abordagem que envolve o uso de restrições e informações semânticas para a criação de uma cena. Na fase de descrição da cena o *designer* expressa como a cena deve parecer. Essa descrição é convertida em restrições que são passadas ao resolvedor de *layout*. O processo é interativo, assim, o usuário pode avaliar o resultado gerado e propor novas modificações que são levadas em conta até que a cena gerada esteja ao seu contento. A Figura 2.5 ilustra o processo.

De forma similar a Martin [44], Lopes et al. [39] também propõem um método de criação de interiores baseado na expansão das peças que irão compor o ambiente. As peças são posicionadas em um gride geométrico por um algoritmo resolvedor de restrições que considera a questão das adjacências de cada peça, suas conexões e zonas funcionais. Após serem acomodadas no gride, as peças passam por um processo de expansão até ocuparem totalmente a área destinada para a construção.

2.2.2 Construções

Wonka [62] apresenta um método para geração automática de fachadas através da derivação de um modelo usando uma gramática de divisão, que é um método paramétrico baseado no conceito

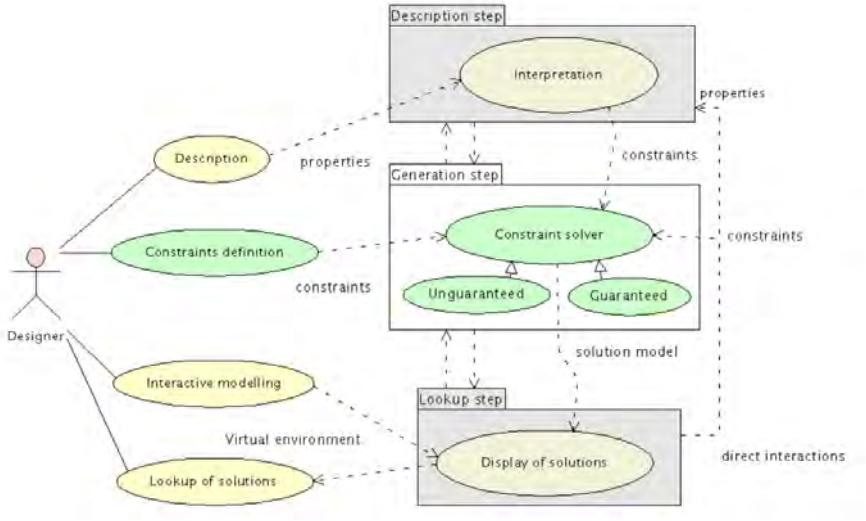


Figura 2.5 – Processo de geração de um *layout* interativo proposto por [21].

de formas. É desenvolvido também um sistema de comparação de formas em conjunto com uma gramática de controle, o que permite uma flexibilidade no momento de gerar diferentes estilos arquitetônicos e *designs*. A Figura 2.6 mostra o modelo esquemático proposto por Wonka para geração das fachadas.

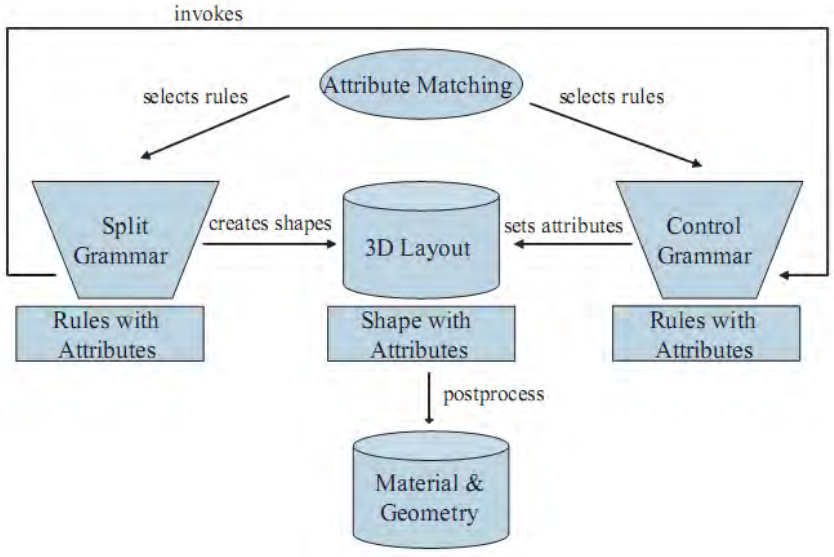


Figura 2.6 – Modelo de geração de fachada proposto por Wonka [62].

Müller [47] propõe uma gramática denominada *CGA Shapes*, voltada para a geração de construções com alta qualidade visual e detalhes geométricos. Utilizando regras de formação, o usuário pode descrever formas geométricas simples, agrupá-las e especificar as relações entre as mesmas, com o objetivo de criar um objeto geometricamente complexo. Uma das principais contribuições do trabalho de Müller é a criação de um sistema de massa que mantém a coerência entre as formas volumétricas criadas. Além disso, ao usuário é permitido interagir dinamicamente em todas as eta-

pas do processo de criação. A Figura 2.7 mostra duas construções geradas pelo modelo de Müller, exemplificando a flexibilidade de criação proporcionada pela gramática *CGA Shape*.

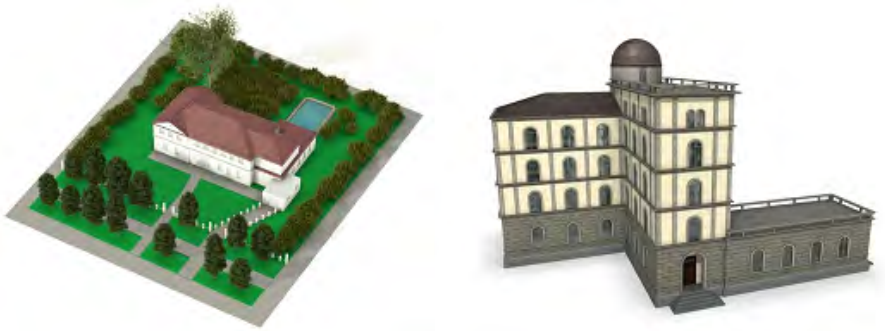


Figura 2.7 – Dois modelos gerados a partir da gramática *CGA Shape* [47].

Em 2007, Müller [48] expande o modelo original proposto por Wonka utilizando as ideias apresentadas em [47], com o objetivo de gerar fachadas sintéticas através de fotografias de fachadas reais de prédios. O princípio de geração segue sendo o mesmo, mas agora é feita uma busca na imagem através da fotogrametria, por padrões previamente armazenados em uma base de dados. Após reconhecidos, os padrões hierárquicos são utilizados como entrada na gramática de divisão para iniciar o processo de reconstrução da fachada. A fotografia da fachada do prédio é utilizada como textura na geometria gerada, aumentando ainda mais o realismo do resultado. Além do realismo e facilidade de geração, uma das contribuições do modelo proposto por Müller é a vinculação de semântica com a geometria, permitindo especificar funções e usos que determinada geometria possui. A Figura 2.8 mostra da esquerda para a direita: foto da fachada de um prédio, reconhecimento do padrão e dos elementos da fachada, criação da geometria baseada no padrão reconhecido e a fachada resultante, com a geometria gerada e a textura da foto.

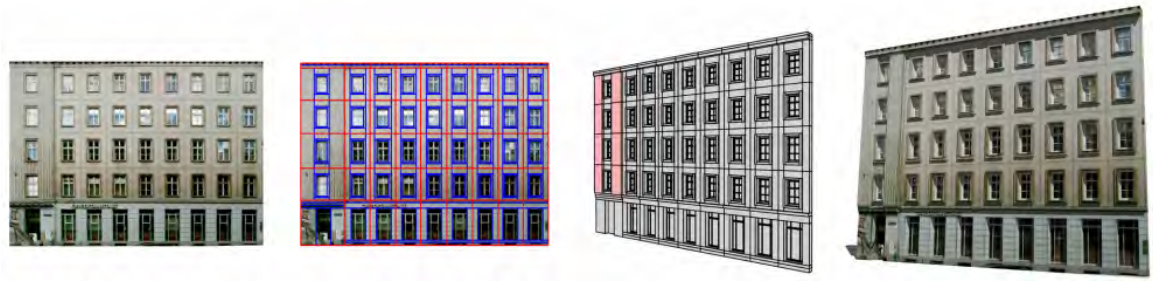


Figura 2.8 – Processo de geração da fachada a partir de uma foto [48].

Finkenzeller [20] [19] apresenta uma técnica para modelagem rápida não só de fachadas, mas do exterior dos prédios através da definição de algumas informações básicas como o contorno do prédio, o seu tipo e estilo e, a partir disso, o algoritmo trata de gerar a estrutura tridimensional. Outros componentes estruturais dos prédios são adaptados automaticamente, como diferentes tipos de texturas celulares para as paredes e contornos de janelas. O *framework* permite a intervenção do usuário em todas as etapas do processo, permitindo mudanças de estilo, material, texturas e

geometria. Cada elemento estruturante da construção, como telhado, portas e janelas, está sob a responsabilidade de um módulo específico, que possibilita definir a aparência que o mesmo irá possuir. A Figura 2.9 exemplifica os tipos diferentes de telhados gerados pelo modelo e a Figura 2.10 mostra uma casa inteiramente criada pelo *framework*.

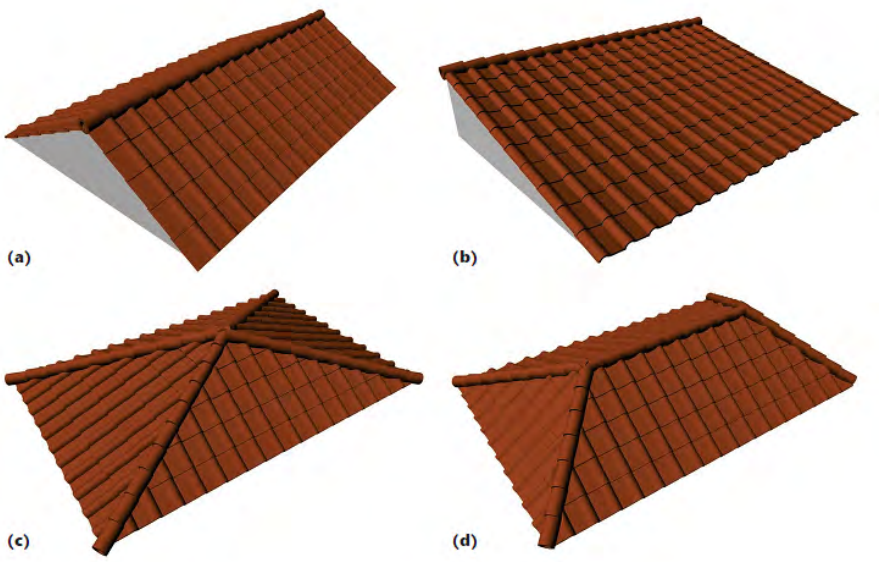


Figura 2.9 – Diferentes tipos de telhados gerados por Finkenzeller [19].

Figura 2.10 – Construção gerada pelo *framework* de Finkenzeller [19].

De forma semelhante a Finkenzeller, Larive [38] descreve um modelo para geração de construções 3D baseado em gramática, partindo de informações simples como contorno do prédio, altura e altura do telhado. Esses dados são aplicados então à modelos pré-definidos que são reconfigurados, gerando

uma geometria tridimensional da construção. O objetivo principal do trabalho é fornecer uma forma de parametrizar a geração de construções de forma rápida a partir de grandes bases de dados e não contempla a criação de construções visualmente sofisticadas como as geradas por Müller e Finkenzeller. O processo de geração pode ser acompanhado na Figura 2.11 da esquerda para a direita: definição dos parâmetros iniciais (contorno do prédio), estrutura básica a partir de uma primeira derivação, definição das extrusões e o prédio gerado.

Figura 2.11 – Processo de geração do modelo proposto por Larive [38].

2.2.3 Ambientes Virtuais

A criação de um ambiente é feita através da combinação de diferentes elementos básicos. Tais elementos são dependentes do contexto em que serão utilizados, podendo variar a sua forma, aparência ou quantidade a fim de criar ambientes únicos. Para geração de uma floresta, por exemplo, é necessária a utilização de um relevo texturizado, além de objetos característicos como plantas e pedras. A modificação de um único elemento, como variações na topologia ou textura do terreno, no tipo, tamanho ou quantidade de árvores, irão gerar ambientes completamente distintos.

Atualmente diferentes técnicas são utilizadas no processo de geração de terrenos. Normalmente a definição da topologia é feita através algoritmos de geração de mapas de altura ou elevação, baseados em *noise* [54], fractais [46] [49] [5] [4], autômatos celulares [13] e algoritmos genéticos [50]. Tais técnicas permitem um alto nível de detalhamento e flexibilidade, criando planícies, montanhas e vales com a simples alteração de poucos parâmetros.

Nesta seção são descritos métodos para geração dos diversos componentes que se relacionam especificamente às cidades virtuais. São apresentados trabalhos para geração de texturas, vegetação, ruas, plantas baixas, fachadas, prédios e mesmo de cidades inteiras. Além disso, são relacionados trabalhos que acrescentam, de alguma forma, informações semânticas ao ambiente virtual.

Em um dos primeiros trabalhos na área, Yap et al. [63] apresentam, como resultado do projeto "*A Different Manhattan Project*", um método para a geração automática de modelos geométricos de cidades baseado em parâmetros estatísticos. A partir de um mapa, a cidade é dividida em bairros. Para cada bairro são atribuídos certos dados estatísticos que irão auxiliar na criação de uma identidade visual para o mesmo. Construções características da cidade, tais como monumentos, são criados manualmente, mas também fazem parte dos dados de entrada. Com base nessas entradas é feita a geração de um modelo tridimensional completo da cidade, com texturas e detalhes urbanísticos. O modelo é composto por cinco módulos: gerador de vizinhança, gerador de blocos,

gerador de ruas, gerador de texturas e o módulo integrador. A Figura 2.12 mostra uma cidade criada pelo sistema.

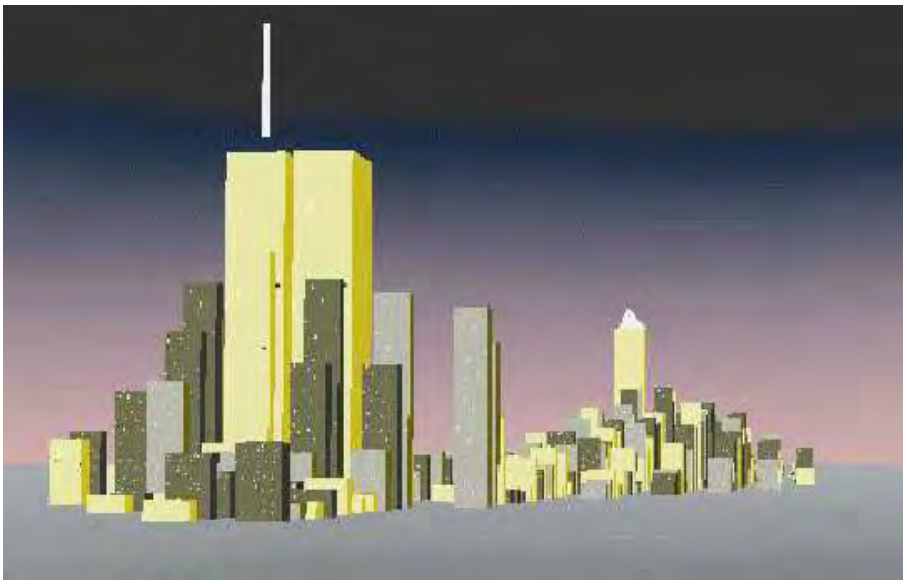


Figura 2.12 – Cidade gerada pelo projeto "*A Different Manhattan Project*" [63].

Parish e Müller [53] propõem um modelo paramétrico, capaz de gerar uma cidade tridimensional a partir de dados sócio-estatísticos e geográficos. O *CityEngine* constrói a malha viária, através de um método L-System [56] estendido usando como entrada mapas de altura, hidrográficos e estatísticos. Após a definição das ruas, o sistema extrai as informações sobre as quadras, as quais passam por um processo de subdivisão a fim de criar os terrenos. Em cada terreno é erguida uma construção, gerada por outro sistema baseado em L-System. De posse dessas informações o sistema gera o modelo geométrico tridimensional da cidade, ao qual são adicionadas texturas visando proporcionar maior realismo ao modelo final. Uma cidade gerada pelo sistema pode ser vista na Figura 2.13.

Honda et al. [28] propõe um método para gerar cidades virtuais que variam dinamicamente durante o tempo, alterando alguns elementos que as compõem, como construções e malha viária. Inicialmente é definido um "vetor ambiental", no qual são agrupadas características refletindo atividades econômicas e funcionais desenvolvidas pelos residentes de cada quadra. A modificação da aparência da cidade é feita modificando os valores desses vetores ambientais através do tempo, modificando em consequência disso o *layout* das constuções e o das própria malha viária (Figura 2.14). A Figura 2.15 mostra três momentos distintos do crescimento de uma mesma cidade.

Marson [42,43] apresenta um modelo paramétrico de descrição de cidades virtuais (VCity), cujas quadras podem ser extraídas automaticamente através de métodos de Processamento de Imagens (PI) ou definidas interativamente. O modelo proposto possibilita entre outras coisas a geração de calçadas, o uso de sementes na criação de construções parametrizáveis, geração de um grafo de conectividade associado às ruas e calçadas da cidade, além da adição de informações semânticas ao modelo geométrico. Uma dos diferenciais da abordagem proposta por Marson é a evitar que ocorram "ilhas", terrenos sem ligação com as calçadas, durante a fase de geração de terrenos. A

Figura 2.13 – Cidade gerada pelo CityEngine [53].

Figura 2.14 – Modificação da malha viária feita em função do crescimento da cidade [28].

Figura 2.16 mostra na esquerda a definição das sementes de construção, que são informações sobre qual tipo de construção deve ocupar aquele determinado terreno e na direita uma imagem das casas geradas a partir dessas informações.

Silveira e Musse [12] apresentam um método para geração de cidades virtuais em tempo real. A principal meta do trabalho é prover um *framework* genérico que suporte a criação semi-automática, gerenciamento e visualização de ambientes urbanos complexos para simulação de humanos virtuais, chamado *Virtual Urban Life*. O objetivo é reduzir o esforço dos *designers* na modelagem de ambientes amplos e complexos. Um modelo de dados multi-nível foi desenvolvido para suportar o gerenciamento de dados e permitir uma visualização mais eficiente. Além disso, também é proposto um algoritmo para resolver a questão do loteamento das quadras, através de parâmetros e restrições. Na Figura 2.17 é possível ver um exemplo de ambiente gerado pelo modelo proposto.

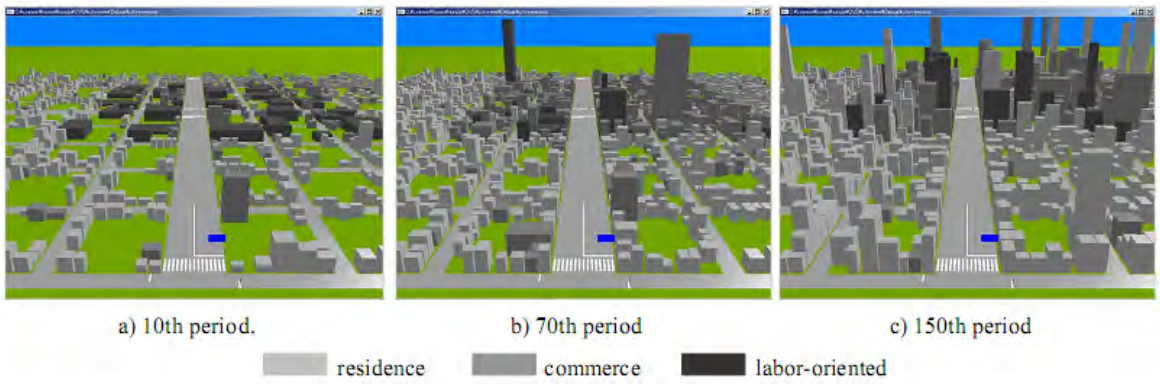


Figura 2.15 – *Layout* da cidade em três momentos distintos do tempo. Os níveis de cinza indicam o tipo de ocupação da construção. [28]

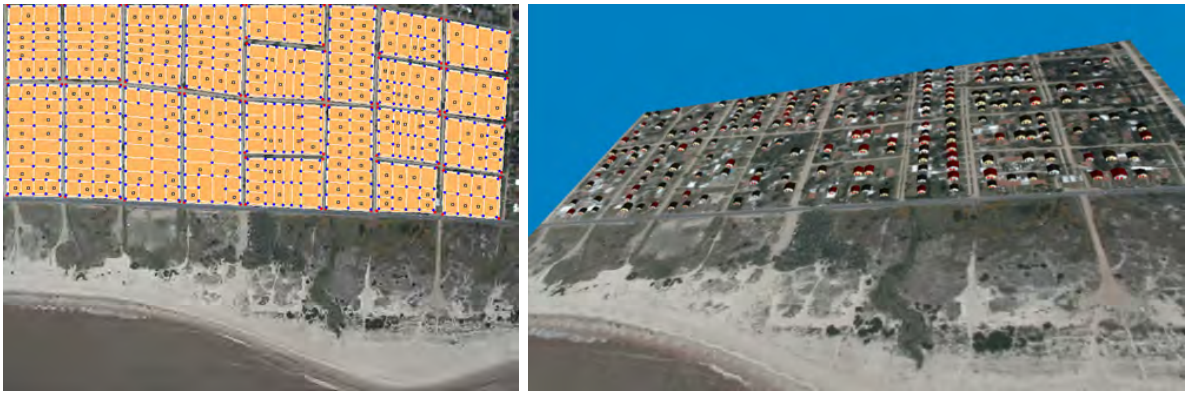


Figura 2.16 – Definição das sementes de construção (esq.). Cidade gerada a partir das sementes de construção (dir.). [42]

Figura 2.17 – Ambiente gerado pelo modelo proposto por Silveira Jr. e Musse. [12]

3. MODELO PROPOSTO

Este capítulo apresenta o modelo desenvolvido neste trabalho para a geração procedural de ambientes residenciais (casas e apartamentos) e de ambientes amplos (como *shopping centers*). Nas seções a seguir são apresentadas a arquitetura geral do modelo que inclui as definições semânticas propostas, o modelo para subdivisão espacial de ambientes e o modelo de geração de informações para a população virtual.

3.1 Visão Geral da Arquitetura

A descrição da arquitetura do modelo faz referência à Figura 3.1. A partir de um modelo mental criado sobre o ambiente que deseja gerar, o usuário define um *template* composto de uma descrição semântica (**a**) do ambiente virtual a ser gerado. Este *template* pode ser customizado para representar diferentes situações onde certas particularidades estão presentes. Essas informações semânticas alimentam o módulo de subdivisão, o qual é composto por dois métodos distintos: o primeiro algoritmo (**b**) baseado em *squarified treemaps* é utilizado para ambientes residênciais como casas e apartamentos. Já o segundo método (**c**) é mais adequado para ambientes amplos como centros comerciais, aeroportos e estações de trem. Ambos os processos (**b**) e (**c**) serão descritos em detalhes nas próximas seções. De posse do ambiente subdividido, o próximo passo é agregar semântica nos ambientes virtuais, o que é feito no módulo (**d**). Como resultado (**e**) é obtido um conjunto de marcadores que descreve a funcionalidade de cada uma das áreas criadas e que, em conjunto com o grafo de conectividade, pode ser utilizado como entrada para simuladores de multidões como apresentado em [27]. Por fim, podem ser geradas também informações da própria planta baixa em 2D (**g**) ou em 3D (**f**).

3.2 Informações Semânticas

Ambientes virtuais geralmente são definidos através da inclusão de uma coleção de objetos geométricos e suas respectivas texturas. Esta abordagem não é a mais interessante quando se trata de um ambiente que será utilizado para a realização de simulações com agentes virtuais, pois os mesmos precisam de informações sobre como interagir com o mundo virtual e seus componentes. Esta questão pode ser resolvida a partir da inclusão de uma camada que agregue informações semânticas sobre o ambiente. Para a criação desta camada, é necessário estabelecer o modelo de relações semânticas a ser utilizado. Tutenel *et al.* [59] propõem um modelo de mapeamento semântico dividido em três níveis distintos: semântica dos objetos, das relações e do ambiente.

O primeiro nível trata das informações relativas ao próprio objeto, incorporando características e atributos que comuniquem mais do que a aparência, como o seu peso, material do qual é feito ou mesmo suas funcionalidades e usos. O trabalho de [33] é um exemplo de pesquisa que trata

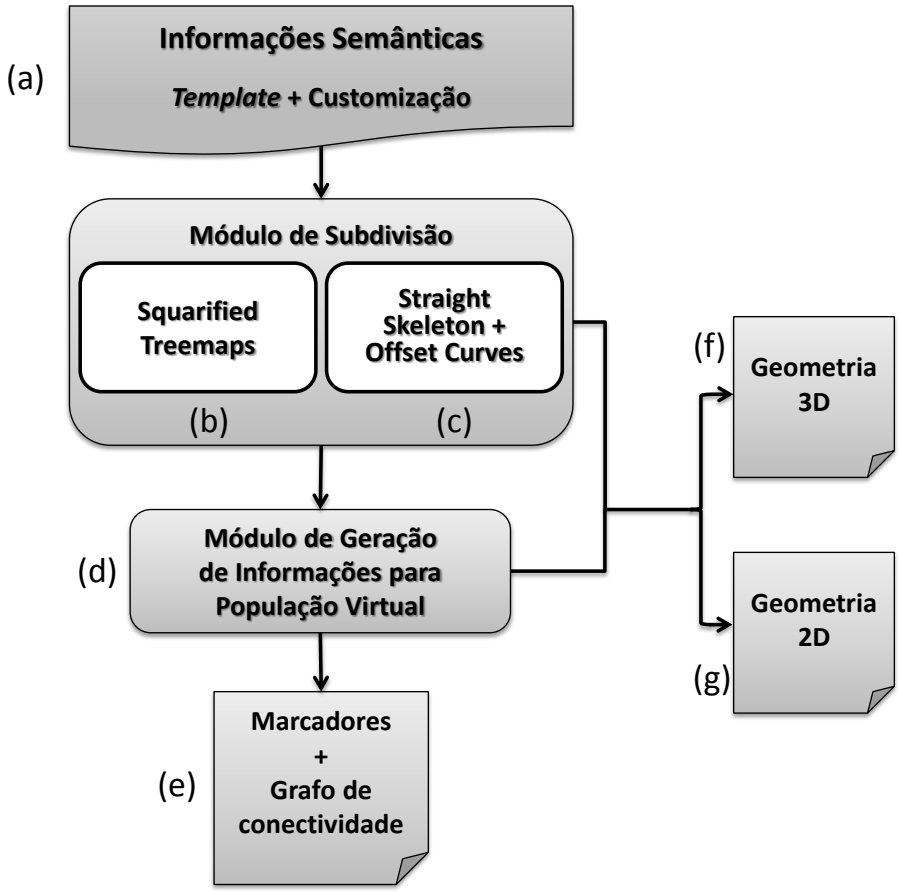


Figura 3.1 – Visão geral da arquitetura proposta.

deste tipo de nível de especificação semântica. O segundo nível de semântica aborda as relações que dois elementos de um ambiente virtual guardam entre si como, por exemplo, uma relação de hierarquia ou de pertinência. Desta forma, pode-se definir que um determinado agente possui um determinado objeto ou que um dado espaço é composto por outros mais simples. O último nível de semântica versa sobre as características do próprio ambiente, permitindo uma compreensão global sobre as características e partes do todo, estabelecendo uma conexão entre todos os elementos que o compõe.

Aplicando como fundamento estes três níveis de semântica, Kessing *et al.* [36] propõem um modelo para a criação de *Semantic Game Worlds*. Este modelo utiliza o conceito de entidades compostas por matéria (*matter*) que podem se relacionar entre si e prover serviços. Toda a camada semântica empregada nesta tese é baseada no *framework* desenvolvido pelo grupo de *Game Technology*[1] da Universidade de Delft, Holanda. Na Figura 3.2 é apresentada uma visão geral do conceito de entidades.

Para fins de definição dos ambientes virtuais neste trabalho, serão utilizados apenas os conceitos de entidades físicas, ou seja, objetos tangíveis (*tangibleObjects*) e espaços (*Spaces*). Estes conceitos

[1] `http://graphics.tudelft.nl/Game_Technology`

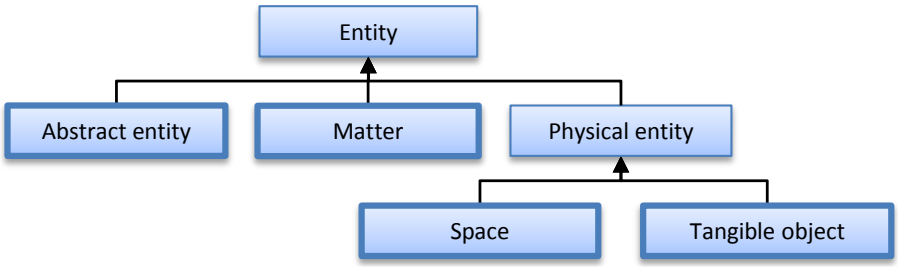


Figura 3.2 – Visão geral do conceito de entidades apresentada em [36].

serão apresentados a seguir.

3.2.1 Objetos Tangíveis

Objetos tangíveis, como o próprio nome diz, são entidades utilizadas na representação de objetos físicos em um ambiente virtual. Este tipo de entidade possui uma ou mais representações geométricas associadas. Estas representações podem ser dadas por arquivos (OBJ, FBX e 3DS, entre outros) ou geradas proceduralmente utilizando *shape grammars*, como CGA [47], ou primitivas geométricas simples.

Os objetos tangíveis possuem uma posição espacial e podem prover serviços para ser requeridos e utilizados por agentes virtuais que necessitem realizar uma certa tarefa dentro de um ambiente virtual. Uma máquina de refrigerantes, por exemplo, pode prover um serviço chamado "'BEBIDA GELADA"', assim como uma máquina de café pode prover o serviço "'BEBIDA QUENTE"'. Se um agente virtual necessitar de uma "'BEBIDA GELADA"', requisitando essa informação ao ambiente, ele saberá todos os objetos que fornecem esse serviço e suas posições no espaço, podendo se deslocar até o objeto o mais próximo de onde se encontra.

3.2.2 Espaços

Os espaços são regiões definidas por uma forma geométrica tridimensional que podem conter outras entidades físicas (espaços e objetos tangíveis). Desta forma, um ambiente virtual pode ser desmembrado em diversos espaços que contém outros espaços e assim sucessivamente, como ilustrado na Figura 3.3. Tanto ambientes residenciais como ambientes amplos podem ser facilmente representados por esse tipo de estrutura hierárquica.

Os espaços não possuem uma representação física geométrica direta, apenas a dos objetos tangíveis representados dentro dos mesmos. Para delimitar os espaços fisicamente, é necessária a criação de elementos físicos como paredes, teto, chão e portas. Os espaços podem prover um determinado tipo de serviço a fim de explicitar o seu uso. Assim, se um espaço A for composto por dois outros espaços ($A1$ e $A2$) que forneçam dois serviços distintos ($S1$ e $S2$), estes serviços serão repassados para o espaço A, fazendo que este espaço indique a disponibilidade dos mesmos dentro de si. Neste sentido, objetos tangíveis presentes em um espaço e que disponibilizem um dado serviço, transferem o mesmo ao espaço que o contém.

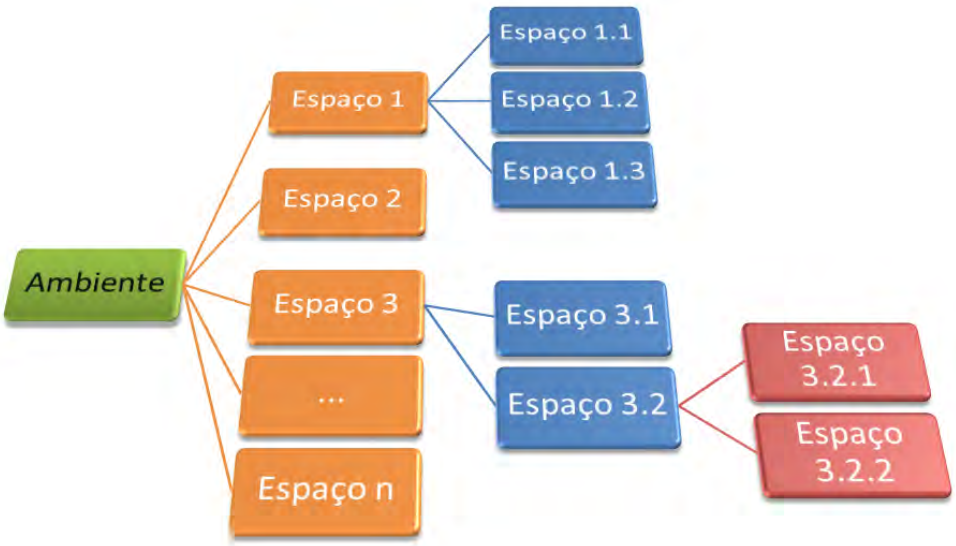


Figura 3.3 – Hierarquia espacial

Nas seções a seguir serão descritas as informações semânticas utilizadas na criação dos ambientes. No Apêndice B estão listados alguns exemplos dos arquivos de informações semânticas. As características necessárias utilizadas como parâmetros para a geração dos ambientes são:

- o tipo de ambiente a ser gerado: residence (residencial) ou complexInfra (amplo);
- a base de dados que irá conter as informações semânticas e geométricas do ambiente virtual;
- a definição do *template* que descreve as entidades e seus atributos;
- a definição de uma possível customização do ambiente a ser aplicado sobre o *template*;
- a definição das instâncias dos espaços a serem criados;
- a definição das instâncias dos objetos a serem distribuídos dentro do ambiente;
- a largura do corredor a ser criado e unidade métrica;
- a espessura e altura das paredes a serem criadas e unidade métrica;
- a altura das portas a serem criadas e unidade métrica.

3.2.3 *Template* e Customização

O *template* é o responsável por definir toda a hierarquia de entidades presentes no ambiente virtual. As entidades basicamente se dividem em objetos tangíveis e em espaços e podem apresentar um número variado de atributos. Os atributos podem ser do tipo textual (AS), numérico (AN) ou lógico (AB). Os mesmos conceitos se aplicam às definições de customização, que servem para

especificar um ambiente derivado de outro mais genérico, previamente definido. Na Listagem 3.1 é apresentado a sintaxe utilizada na definição das informações de *template* e de customização.

Listing 3.1 – Sintaxe utilizada nas informações de *template* e de customização.

```
tangibleObject NAME
        as provides STRING, STRING, ....

space NAME <PARENT>
        as quantity NONE | UNIQUE | FEW | SEVERAL
        an nquantity VALUE | VALUE minValue VALUE maxValue VALUE | minValue VALUE maxValue VALUE
        ab splittable FALSE | TRUE
        ab growable FALSE | TRUE
        an footage VALUE <UNIT> minValue VALUE <UNIT> maxValue VALUE <UNIT>
        an percentage VALUE <%> minValue VALUE <%> maxValue VALUE <%>
        an aspectRatio VALUE
        as spaceType CONCEPTUAL | FIXED | FLOATING | CORRIDOR
        as provides STRING, STRING, ....
```

3.2.4 Instâncias de Objetos

Nesta etapa define-se as informações sobre os objetos a serem instanciados no ambiente virtual. Este tipo de arquivo pode ser gerado manualmente ou através de algum método de disposição automática de objetos como os apresentados por [60] [45] [64]. Basicamente devem ser definidos o tipo do objeto (especificado previamente no *template*), nome do objeto, nome e caminho do objeto em disco, em qual espaço ele está inserido, sua posição espacial e sua orientação no eixo Y.

3.2.5 Instâncias de Espaços

Da mesma forma que definido para os objetos, é possível instanciar manualmente os espaços a serem criados. Esta funcionalidade foi acrescida para permitir a prototipagem rápida de ambientes virtuais com um *layout* pré-definido. Três categorias de espaços podem ser descritas: *main spaces* (que deve ser único por ambiente), *open spaces* (sem paredes) e *closed spaces* que apresentam paredes ao seu redor. Além da categoria e do tipo do espaço (definido no *template* ou na customização), é informado também o nome desejado para o espaço, o número de pontos e os pontos que compõem o contorno do polígonos e, por fim, para os espaços fechados, o número de entradas e janelas e as suas respectivas posições 2D.

3.3 Módulo de Subdivisão Espacial

O módulo de subdivisão é responsável por selecionar, a partir das informações semânticas fornecidas pelo usuário, o método a ser utilizado para a divisão do espaço. O modelo contempla atualmente dois métodos de divisão: *Squarified Treemaps* e o *Straight Skeleton*. O primeiro método é mais aplicável aos espaços internos, como casas e apartamentos, e o segundo mais adequado para ambientes amplos. A seguir são descritos e analisados ambos os métodos.

3.3.1 Subdivisão Espacial de Ambientes Residenciais

Treemap é um método para visualização de informações hierárquicas estruturadas na forma de uma árvore, proposto por Johnson e Shneiderman [31]. A Figura 3.4 mostra um exemplo de uma árvore composta por onze nodos. Cada nodo na árvore tem associado um valor que indica o quanto esse nodo ocupa, em área, isoladamente na estrutura. Como o nodo a é o nodo raiz da árvore mostrada na Figura 3.4-a, o seu valor reflete o valor total da mesma (20). Na mesma árvore, o nodo f tem um valor de 5 unidades. O *treemap* (Figura 3.4-b) é construído usando um processo de subdivisão recursiva de um retângulo inicial, onde a área reflete o valor total representado pelo nodo raiz. A direção da subdivisão (vertical ou horizontal) é alternada por nível caminhado na árvore. Consequentemente, o retângulo inicial é subdividido em retângulos menores, com uma área equivalente ao seu valor associado. Maiores detalhes sobre o funcionamento dos *treemaps* pode ser encontrado em [31].

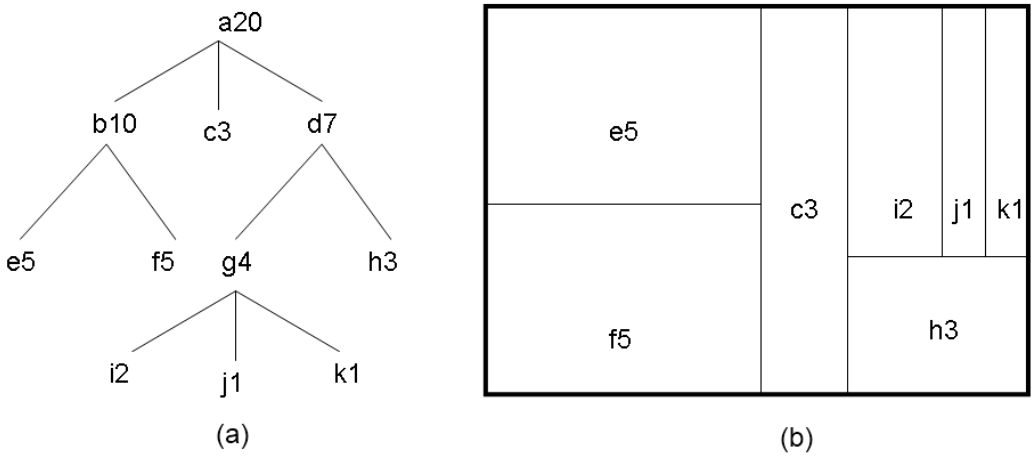


Figura 3.4 – Estrutura hierárquica organizada em árvore (a) e seu respectivo *treemap* (b).

Este método pode originar subdivisões como a ilustrada na Figura 3.5. Neste caso, é possível perceber que as proporções (*aspect ratio*), definidas como $max(\frac{altura}{largura}, \frac{largura}{altura})$, dos retângulos gerados são muito diferentes de 1. Consequentemente, esta abordagem não é adequada para resolver o problema de geração de plantas baixas, pois geralmente as peças representadas em uma planta baixa se aproximam de 1.

Squarified treemaps foi proposto por Bruls et al. [8] e o seu principal diferencial em relação ao algoritmo original é a manutenção do *aspect ratio* o mais próximo possível de 1. Este método recursivo gera retângulos baseado em uma lista ordenada decrescente de áreas desejadas. Durante o processo de subdivisão, a orientação da mesma é definida, não pelo nível caminhado na árvore como no algoritmo original, mas sim pelo *aspect ratio* da parte a ser dividida. A cada passo p, a razão da área gerada é testada com a razão obtida no passo anterior $p-1$. Caso o *aspect ratio* gerado no passo $p-1$ seja mais próximo de 1 do que o do passo atual, a operação de subdivisão atual é abortada, a direção de divisão é invertida novamente e a operação é refeita somente para o nodo atual. Para melhor compreensão do processo de geração de um *squarified treemap*, será

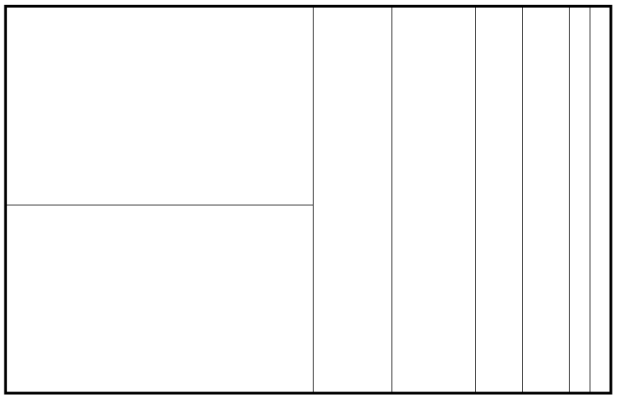


Figura 3.5 – Exemplo de um *treemap* onde os retângulos gerados no processo de subdivisão são muito diferentes de 1.

apresentado, passo a passo, o exemplo ilustrado na Figura 3.6.

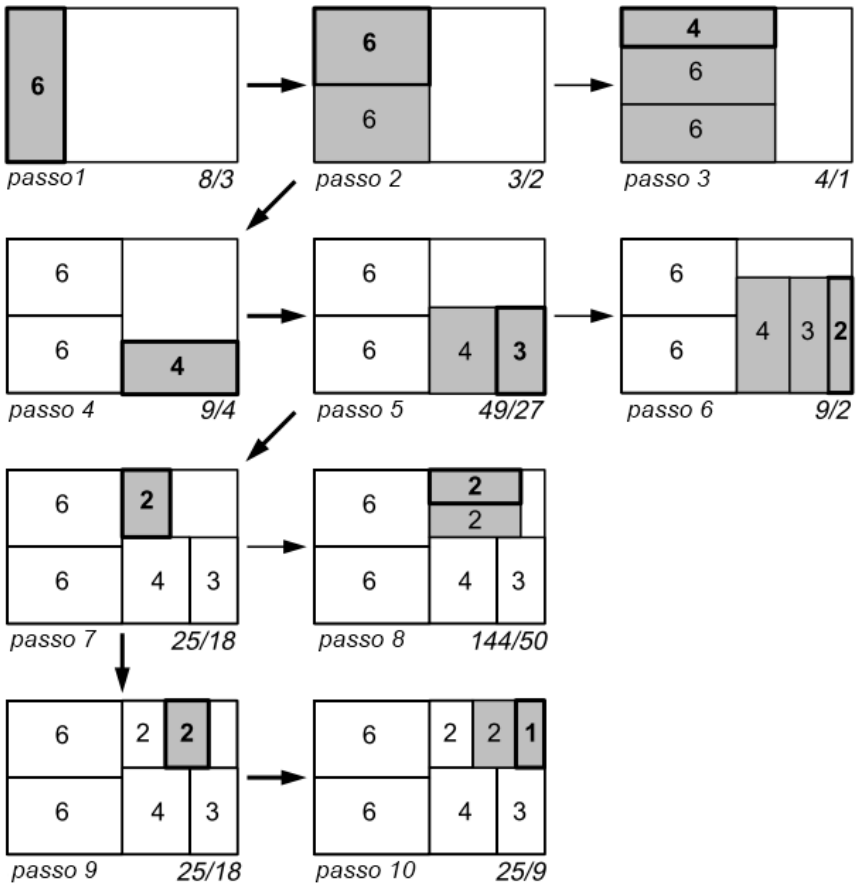


Figura 3.6 – Passo a passo do processo de construção de um *squarified treemap* (adaptado de [8]).

Para que o processo tenha início, é necessário organizar as áreas dos nodos de forma decrescente, a fim de utilizá-las como entrada para o algoritmo. Neste exemplo, os valores adotados para cada subárea são 6, 6, 4, 3, 2, 2, e 1, totalizando uma área de 24 unidades. O retângulo que representa

a área a ser subdividida mede 6 unidades de largura por 4 unidades de comprimento. No $passo\,1$, o método gera um retângulo com *aspect ratio* $= \frac{8}{3}$ em uma divisão vertical. No segundo passo é testada a subdivisão no sentido horizontal, gerando dois retângulos com *aspect ratio* $= \frac{3}{2}$, o qual é mais próximo de 1 do que a divisão anterior. No $passo\,3$, o próximo retângulo é gerado, apresentando um *aspect ratio* $= \frac{4}{1}$. Este passo é ignorado e o $passo\,4$ é computado com base nos retângulos obtidos no $passo\,2$. O procedimento continua até que todas as áreas da lista sejam geradas. Considera-se que para este trabalho, esta versão do *Squarified Treemap* é mais interessante quando aplicado na geração de plantas baixas do que o algoritmo original, em função do fato de que peças presentes em construções residenciais também tendem a apresentar razões próximas de 1.

Porém, antes de aplicar o algoritmo de subdivisão, é necessário realizar o agrupamento das peças que fazem parte da residência a partir da funcionalidade indicada nas informações semânticas fornecidas pelo usuário. Esta funcionalidade indica como uma determinada área da residência será utilizada. Existem três possibilidades distintas: área social, área de serviço e área privativa. A área social é composta pela sala de estar, pela sala de jantar e pelo *toilet*. Na área de serviço é possível existir a cozinha, a copa e a lavanderia. Por fim, a a área privativa abrange os quartos, suítes, banheiro e possíveis peças secundárias que podem ser utilizadas de diversas maneiras como, por exemplo, biblioteca ou escritório. Esta lista não é fixa e pode ser customizada pelo usuário.

A divisão da área da residência ocorre em duas etapas diferentes. Inicialmente as peças são agrupadas por funcionalidade nos três tipos de áreas descritas anteriormente (social, serviço e privativa). Este processo geral um *layout* inicial contendo até três retângulos, onde cada um representa um agrupamento distinto (Figura 3.7a).

A segunda etapa da divisão é iniciada após a obtenção das posições dos polígonos que representam cada agrupamento por funcionalidade. Cada retângulo irá servir como entrada para uma nova subdivisão através do algoritmo de *squarified treemap* usado anteriormente, desta vez, gerando a geometria de cada peça que compõe o grupo. Após a geração das peças é necessário a criação de conexões entre as mesmas (portas, por exemplo). Estas conexões são criadas usando como critério, a funcionalidade de cada peça. Todas as conexões possíveis que serão utilizadas neste trabalho estão representadas na Tabela 3.1, a qual foi criada a partir da análise prévia de plantas baixas comerciais de casas e apartamentos. Nesta tabela, foram consideradas como entradas possíveis para a residência apenas a cozinha e a sala de estar. Da mesma forma que as peças, essa tabela também pode ser adaptada para refletir outras configurações.

As portas são criadas nas arestas compartilhadas por duas peças que mantém uma possível conexão. O tamanho da porta e a sua posição preferencial na parede (lado esquerda, centro ou lado direito) são pré-definidos pelo usuário. Um processo similar acontece com as janelas, com a diferença de que janelas somente são criadas em arestas que fazem fronteira com o exterior da residência. A Figura 3.7b ilustra uma planta baixa gerada pelo método, contendo janelas (retângulos cinzas) e portas (retângulos brancos).

Após processamento das conexões, um grafo de conectividade é automaticamente gerado (Fi-

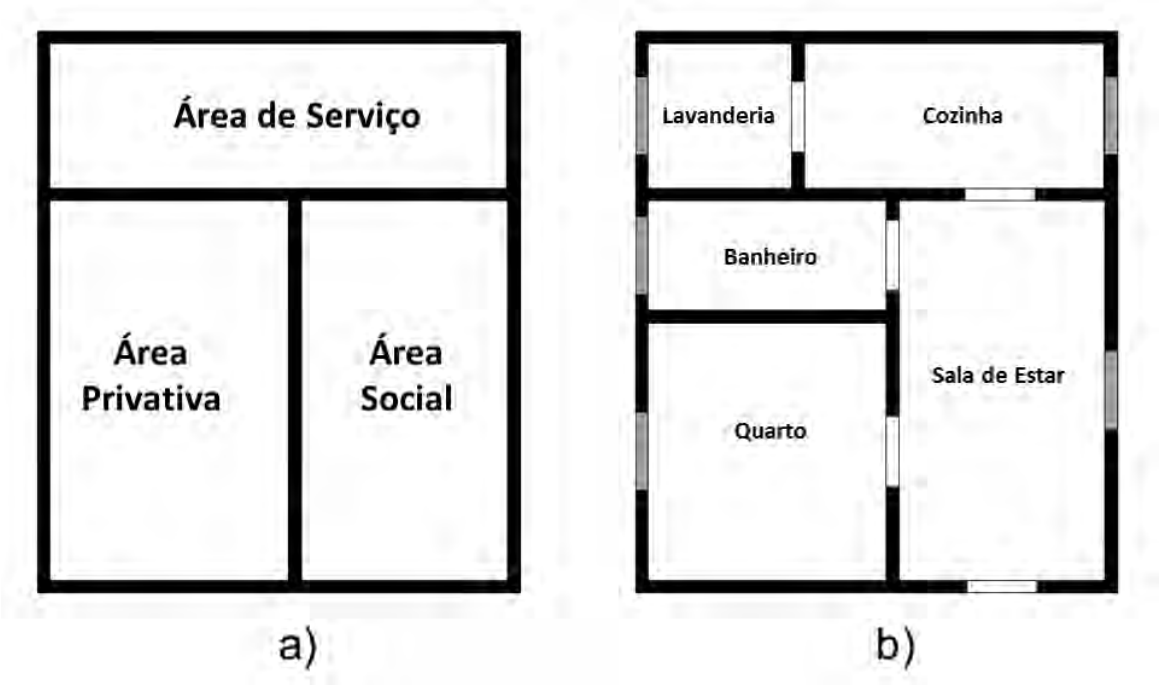


Figura 3.7 – Divisão da área total da residência em três áreas principais (a): privativa, social e serviço. Exemplo de planta baixa gerada pelo modelo (b).

gura 3.8), representando as ligações entre as peças. Este grafo permite verificar a existência de peças não conectadas à nenhuma outra peça, significando que a mesma é inacessível. O nodo inicial do grafo sempre se localiza na porta que faz a conexão com o exterior (sala de estar). Se alguma peça não estiver presente no grafo, se faz necessária a inclusão de corredores.

A geração de corredores é necessária para manter a coerência do ambiente criado. A ideia principal do processo é encontrar todas as arestas compartilhadas possíveis, entre as peças não conectadas (marcadas com um X na Figura 3.9a) e usar o menor caminho que as liguem com a sala de estar (marcada com um L na Figura 3.9a). A solução proposta utiliza as paredes internas da construção para criar uma espécie de "'espinha-dorsal"' de circulação no espaço.

O algoritmo é simples e composto por três partes principais. Primeiramente, todas as paredes externas são removidas, evitando assim a criação de corredores nas extremidades da mesma (Figura 3.9b). Logo após, todos os segmentos internos (paredes) que pertencem à sala de estar são removidos. Os segmentos remanescentes (descritos pelos seus vértices) são utilizados como entrada para a criação de um grafo de conectividade. Os vértices representam os nodos e os segmentos descrevem as arestas no grafo (Figura 3.9c). A última parte do processo é a extração do menor caminho que conecte todas as peças sem acesso. Isso é alcançado através do algoritmo A* [26], que é um algoritmo bem conhecido para o cálculo de caminhos. O algoritmo é utilizado no sentido de explorar o grafo a fim de encontrar as arestas que conectem todos as peças sem acesso anteriormente e a sala de estar. Uma peça é considerada conectada se o caminho passa por pelo menos uma de suas arestas. Ao final do processo, um conjunto de caminhos possíveis é gerado. O caminho que apresentar o menor custo (distância entre todas as peças) é escolhido para representar o corredor

	exterior	cozinha	copa	lavanderia	sala de estar	sala de jantar	toilet	dormitório	suíte	banheiro	peça secundária
exterior		X			X						
cozinha	X		X	X	X	X					
copa		X		X							
lavanderia		X	X								
sala de estar	X	X				X	X	X	X	X	X
sala de jantar		X			X		X	X	X	X	X
toilet					X	X					
dormitório					X	X				X	
suíte					X	X				X	
banheiro					X	X		X	X		X
peça secundária					X	X				X	

Tabela 3.1 – Conexões possíveis (propostas pelo autor) entre as peças de uma residência.

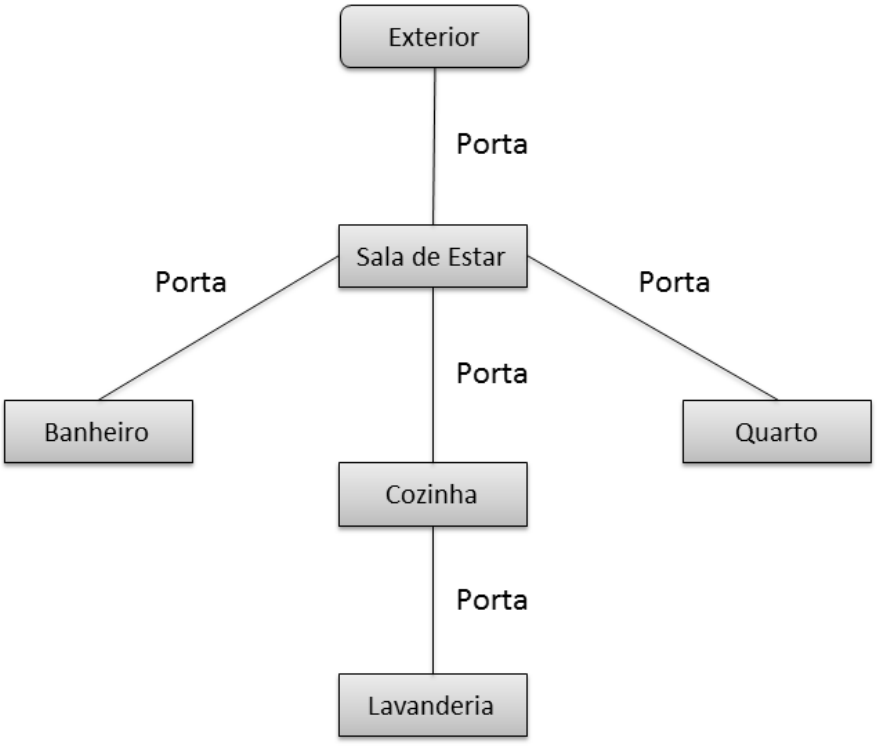


Figura 3.8 – Grafo de conectividade gerado para uma planta baixa.

(Figura 3.9d).

Após a obtenção do menor caminho, é necessária a geração dos polígonos que irão representar o corredor na planta baixa. O conjunto de arestas passa por um processo de extrusão 2D, a fim

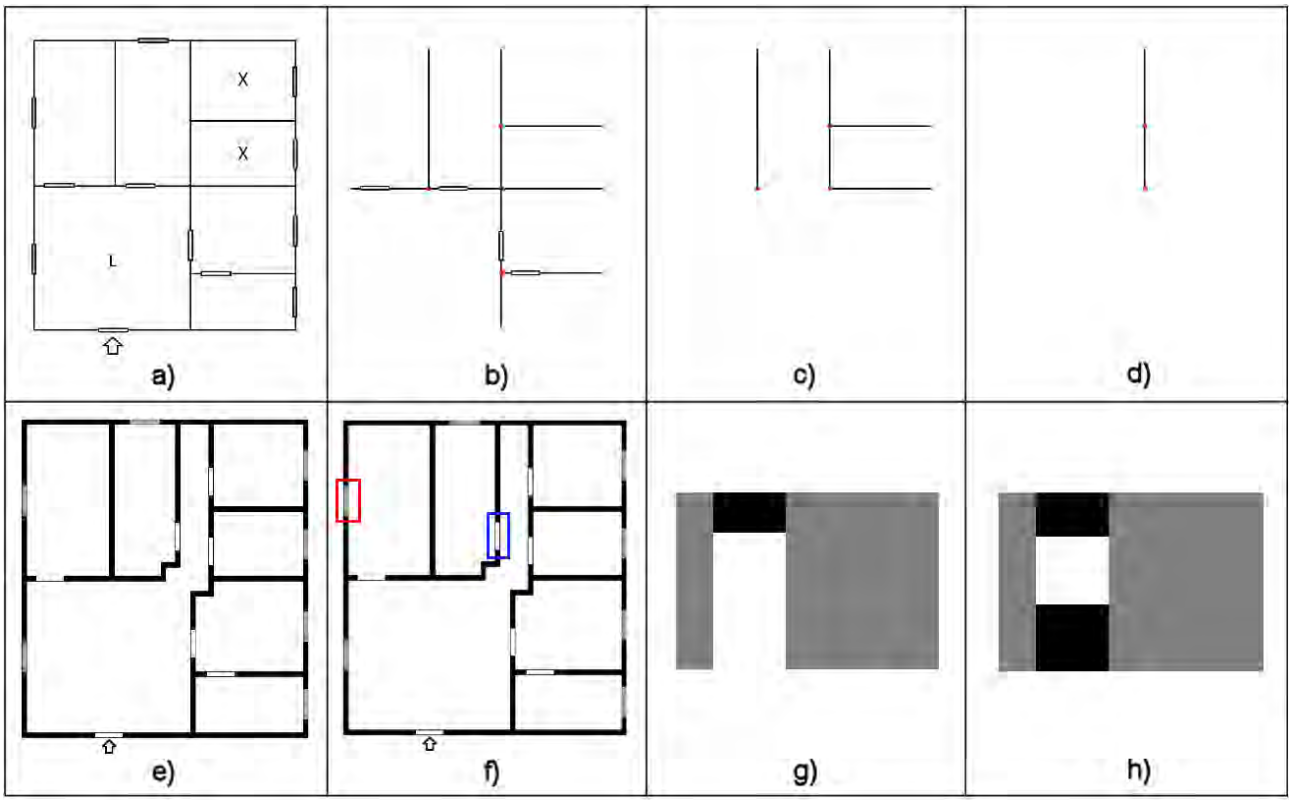


Figura 3.9 – Passos para a criação da residência. a) Planta baixa original gerada pelo modelo. As peças sem conexão (portas) estão marcadas com um X e a sala de estar com um L. b) Planta baixa após a remoção das paredes externas. c) Paredes internas que não pertencem à sala de estar. d) Menor caminho conectando todas as peças inacessíveis à sala de estar. e) Planta baixa final. f) Os retângulos vermelho e azul destacam, respectivamente, a representação de uma janela e de uma porta na planta baixa final. Representação do processo de criação de janelas (g) e portas (h).

de gerar retângulos que serão unidos em um único polígono, que tornará o ambiente próprio para ser explorado por agentes virtuais. As partes das peças onde acontecem sobreposições destas e do corredor são removidas dos polígonos que representam as peças e repassadas ao polígono do corredor. Se o tamanho de alguma peça for menor do que o mínimo permitido, a planta baixa inteira passa por um processo de reajuste onde a área total da casa é aumentada até que a área da peça fique de acordo com o esperado.

De posse da planta baixa que representa a construção em 2D (Figura 3.9e) é possível gerar uma representação 3D através de outro processo de extrusão, desta vez em relação ao eixo Y. Cada parede é extrudada uma dada altura H, definida pelo usuário. As paredes podem ter portas (retângulo azul na Figura 3.9f) e janelas (retângulo vermelho na Figura 3.9f), que devem ser geradas apropriadamente, o que é feito através do recorte do espaço necessário na parede criada (figuras 3.9g e 3.9h). Uma visualização tridimensional da planta baixa representada na Figura 3.9e é apresentada na Figura 3.10.

O algoritmo para divisão do espaço pelo *straight skeleton* [1] parte da premissa de que as áreas dos espaços a serem criados já são conhecidas, o que no caso dos ambientes amplos não é uma verdade. A alocação das áreas é feita dinamicamente em função da área existente e do *template*

Figura 3.10 – Visualização tridimensional da planta baixa representada na Figura 3.9e.

e da customização definida para a construção. Outra questão fundamental diz respeito ao formato dos corredores. Em ambientes amplos normalmente são observados corredores longos e retos ou com leves curvas. Os corredores gerados nos ambientes residenciais buscam o menor caminho de ligação entre as peças, ocasionando, geralmente, corredores mais curtos e angulados.

3.3.2 Subdivisão Espacial de Ambientes Amplos

Straight Skeleton é uma forma de representação de polígonos através de esqueletos topológicos criada por Aichholzer *et al.* [2]. Ele é composto por segmentos de linhas retas que particionam o interior de um polígono. Ele é definido por um processo de encolhimento do polígono, onde as arestas são movidas paralelamente em direção ao seu interior em uma velocidade constante. A Figura 3.11 ilustra o processo de criação do *Straight Skeleton* através do processo de encolhimento do mesmo (Figura 3.11b). Uma aplicação comum em Computação Gráfica do algoritmo de *Straight Skeleton* é a geração de telhados, a partir de um polígono formado pelas paredes externas da construção [35].

No caso da geração de ambientes amplos, a aplicação do algoritmo de *Straight Skeleton* é feita na criação dos corredores de circulação do ambiente virtual e na consequente subdivisão do mesmo. Este processo de subdivisão do espaço é realizado nos passos que serão listados e descritos abaixo.

A primeira etapa consiste no mapeamento do contorno e das saídas da construção a ser gerada. A ilustração desta etapa do processo pode ser vista na Figura 3.12. Esse mapeamento é feito com base nas informações semânticas fornecidas pelo usuário. Para que se garanta a existência de uma conexão entre o interior e as entradas e saídas da construção, é feita a marcação das posições das mesmas no contorno do ambiente (Figura 3.12a) e logo após são agregadas protuberâncias (Figura 3.12b) que irão atrair as arestas do *Straight Skeleton* até o ponto definido como conexão do ambiente com o exterior (Figura 3.12c). Após a criação do *Straight Skeleton* todas as arestas que compartilham um vértice com o contorno são removidas (Figura 3.12d). As arestas remanescentes são submetidas

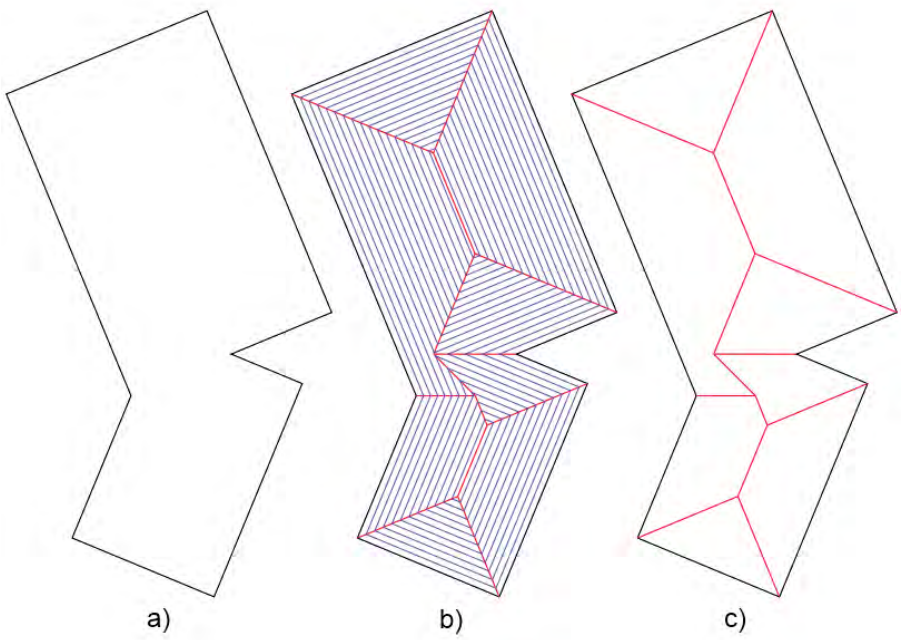
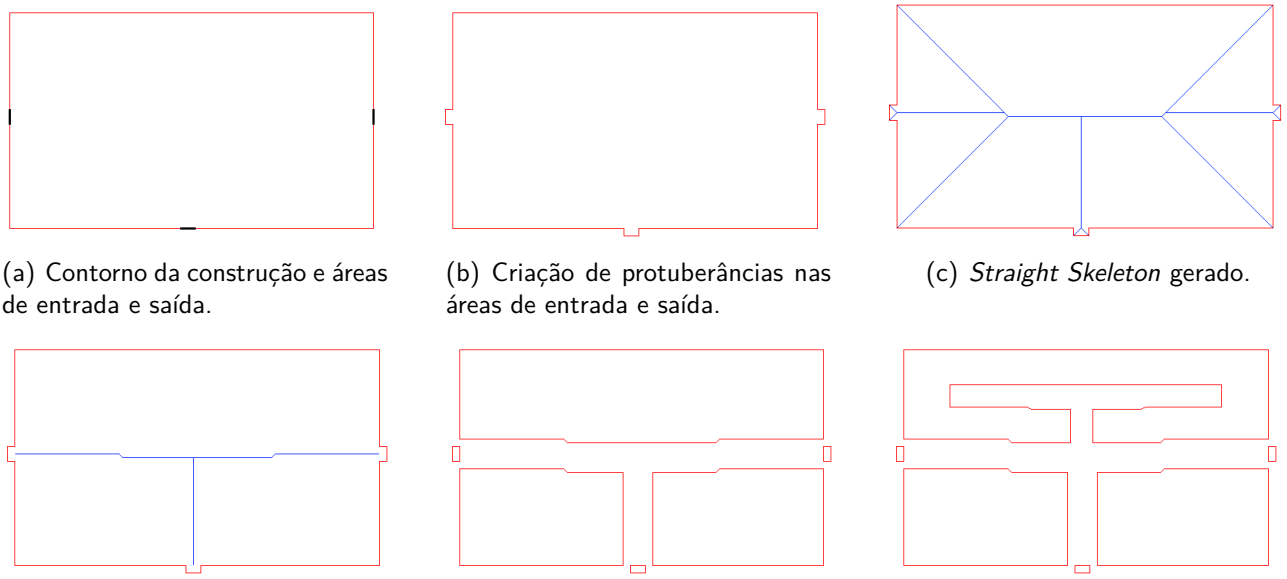


Figura 3.11 – Criação do *Straight Skeleton* de um polígono. a) Polígono original. b) Processo de encolhimento do polígono em azul. c) *Straight Skeleton* resultante representado em vermelho. Ilustração inspirada em [9].

à um processo de dilatação, para obter a forma final do corredor principal (Figura 3.12e). O valor da dilatação segue os limiares pré-definidos para a largura do corredor na Seção 3.2.



(a) Contorno da construção e áreas de entrada e saída.

(b) Criação de protuberâncias nas áreas de entrada e saída.

(c) *Straight Skeleton* gerado.

(d) Remoção das arestas do *Straight Skeleton* que possuem vértices ligados ao contorno da construção.

(e) Geração do corredor principal através de um processo de dilatação das arestas.

(f) Geração de um corredor secundário através da reaplicação do algoritmo de *Straight Skeleton*.

Figura 3.12 – Etapas de criação do corredor principal.

Após a realização da subdivisão inicial do espaço e criação dos corredores principais, é importante

verificar a hierarquia desejada pelo usuário, a qual foi expressa através do *template* e de uma possível customização. Esta estrutura auxilia na escolha dos locais mais adquados para a criação de cada espaço e que se transformarão, no caso de uma *galeria comercial*, em lojas. Esta segunda etapa permite verificar se existe a necessidade de gerar corredores secundários.

De posse das áreas (polígonos) remanescentes após a geração do corredor principal (Figura 3.12e), é feita uma alocação lógica de cada espaço da hierarquia nestas áreas. Este processo verifica qual área é mais adequada para receber cada tipo de espaço e quantos espaços de cada tipo serão criados. A hierarquia presente no *template* é apenas um guia para a criação destes espaços, assim, um mesmo *template* aplicado à duas áreas diferentes, poderá produzir dois resultados distintos. Supondo, por exemplo, que a hierarquia de espaços deste ambiente amplo definisse a necessidade de criação de quarenta instâncias de pequenas lojas (uma galeria popular), é natural imaginar pela área total que dez lojas ficariam em cada região inferior e vinte lojas na região superior. Como resultado de uma subdivisão do espaço através do *squarified treemaps* (apresentado na Seção 3.3.1), algumas lojas ficariam sem saída para o corredor principal gerado pelo algoritmo de *Straight Skeleton*. Assim, definiu-se que as regiões que possuirem espaços sem acesso ao corredor principal devem sofrer novamente um processo de subdivisão utilizando o próprio *Straight Skeleton*. Até que todos os espaços sejam acessíveis pelos corredores secundários ou principais. A Figura 3.12f mostra a subdivisão final deste processo.

É importante salientar que este processo iterativo acontece para a geração automática de ambientes. No entanto, também é possível definir estes ambientes através de informações definidas pelo usuário, como é o caso dos ambientes gerados na Seção 4.2.

3.4 Módulo para Geração de Informações para População Virtual

Após a etapa de geração espacial e geométrica do ambiente virtual, é necessário agregar algumas informações que possibilitem a utilização dos ambientes por agentes virtuais. Isto implica em gerar informações para o planejamento de caminhos, além de inserir os dados de funcionalidade dos espaços, que foram definidos na fase definição do semântica do ambiente.

É possível a inclusão de informações para caminhamento nos ambientes gerados através da definição de marcadores [14]. Os marcadores são pontos que discretizam um determinado espaço (2D ou 3D) e que no caso de alguns modelos de simulação [14] [10], são usados para prover movimento movimento aos agentes virtuais. Estes pontos servem para descrever movimentos sem colisão de multidões [14], bem como podem possuir outras características como, por exemplo, se representam regiões caminháveis ou não caminháveis.

Além dos marcadores, os espaços gerados também são identificados conforme definido na semântica dos espaços. Por exemplo, no caso da residência, os espaços são identificados com nomes como *toilet*, suíte, cozinha e etc. Estas informações são de extrema importância para a sequência de simulações, assim como para posicionamento automático de móveis e movimentação dos agentes virtuais. Alguns resultados neste sentido são apresentados no próximo capítulo.

Além dos marcadores, é gerado um grafo de conectividade no ambiente virtual que pode ser utilizado como entrada para algoritmos de caminhamento como o A*. Estes grafos podem auxiliar na escolha de rotas globais mais eficientes para o deslocamento de agentes virtuais. Eles são construídos a partir das informações contidas nos marcadores que indicam se determinada região é caminhável (livre de obstáculos), preferencialmente não caminhável (um gramado, por exemplo) ou não caminhável (região ocupada por um obstáculo). Ambas as estruturas utilizam uma granularidade informada pelo usuário (densidade de marcadores por metro quadrado), assim é possível determinar a quantidade e distribuição de marcadores no ambiente e o número de nodos do grafo.

4. RESULTADOS

Nas próximas seções serão apresentados os resultados obtidos pelo modelo proposto e aplicados em trabalhos de outros pesquisadores, nas áreas de Simulação de Humanos Virtuais e de disposição automatizada de objetos. As seções agrupam os trabalhos em ambientes residenciais e ambientes amplos.

4.1 Ambientes Residenciais

Cassol [10] utiliza como entrada os ambientes residenciais para a realização de simulações de grupos de humanos virtuais. Os ambientes residenciais, são representados através de um grafo de conectividade ligando as peças que compõem a casa, conforme definido no Capítulo 3. Ilustrado na Figura 4.1 (A), o grafo tem os ambientes representados por nodos e as portas por arestas. Para a execução do *path planning*, conforme objetivo de Cassol, o grafo é ajustado para que as portas sejam consideradas como nodos, igualmente aos cômodos, conforme ilustrado na Figura 4.1 (B). Para a movimentação do agente, é considerada uma série de ações pré-definidas que, por sua vez, são realizadas em determinado ambiente da casa. Atualmente, são consideradas as ações e os locais representados na Tabela 4.1.

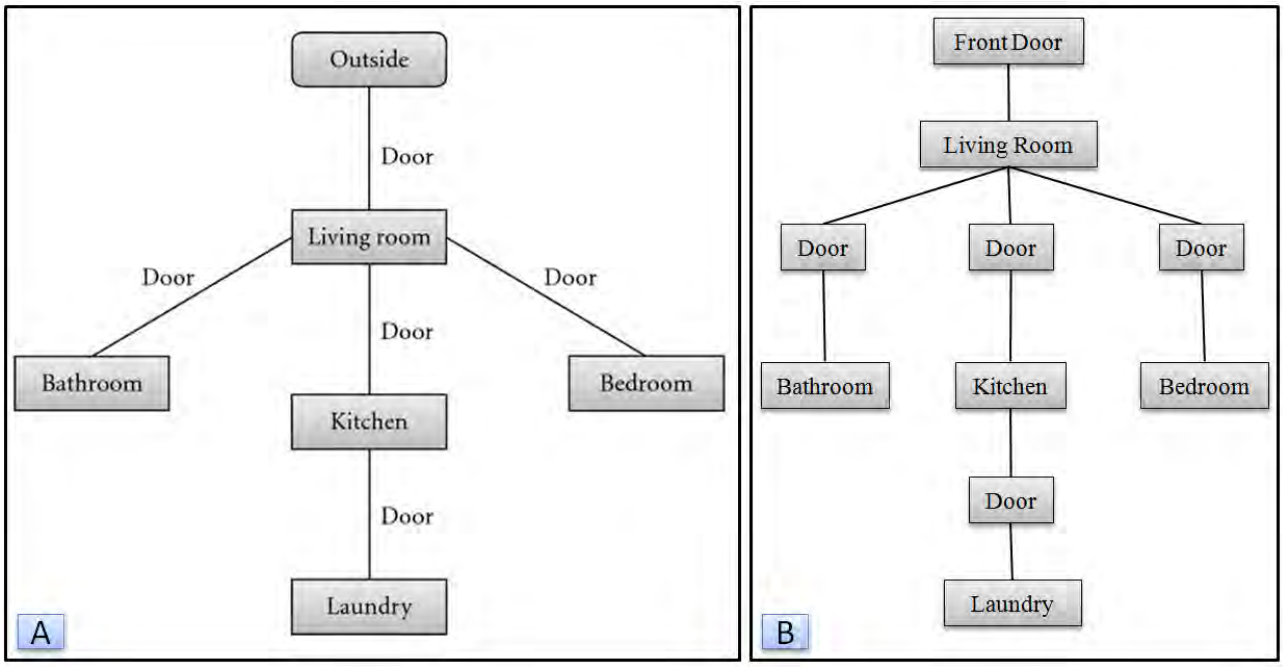


Figura 4.1 – Em (A), representa-se o grafo de conectividade considerado para representação de um ambiente residencial com respectivas características e, em (B), ilustra-se o mesmo grafo após adequações para ser utilizado na produção de *pathplanning* para a simulação de agentes virtuais na residência.

Cômodos	Ação
Living Room	read, play
Bedroom	sleep
Bathroom	shower, wash hands, bath
Kitchen	eat, drinnk, wash the dishes
Laundry	wash clothes

Tabela 4.1 – Ações e locais de realização definidos para serem considerados na simulação de agentes em ambientes residenciais.

De posse de uma ação que deve ser realizada, o algoritmo de planejamento de caminho é executado para gerar o caminho partindo da posição do agente (que também é considerada como um nodo - origem) até a porta do cômodo onde a ação deve ser realizada (nodo destino). Com isso, tem-se a lista de sub-objetivos a serem contemplados na movimentação do agente. Todavia, considerando a porta do ambiente como o ultimo ponto do caminho gerado, define-se uma posição dentro do cômodo para que o agente se desloque até ela e execute a ação. Esta definição é possível visto que neste momento, os marcadores possuem associados a si a semântica do espaço em que estão alocados, através do atributo local (l_k) que agora passa a definir a peça que o marcador está representando. Atualmente, a movimentação do agente é feita considerando somente o local em que marcador se encontra, porém, destaca-se que ao considerar ambientes mobiliados, os marcadores também podem levar em conta esta informação e utiliza-la na definição do movimento do agente, para definir obstáculos ou objetos com quais os agentes devam interagir, em busca de resultados coerentes.

Ao chegar no local determinado para a execução da ação, o agente aguarda determinado intervalo de tempo pré-definido no local das ações, a fim de simular que a atividade está sendo realizada no local determinado. Ao finalizar a ação, escolhe-se randomicamente uma nova ação a ser realizada e o agente se desloca até o respectivo local, conforme processo descrito anteriormente.

A Figura 4.2 apresenta a casa gerada pelo modelo proposto (B), juntamente com o grafo de conectividade utilizado em sua construção (A). Em (C), é apresentada a visualização da mesma no ambiente de simulação, onde os marcadores também podem ser visualizadas.

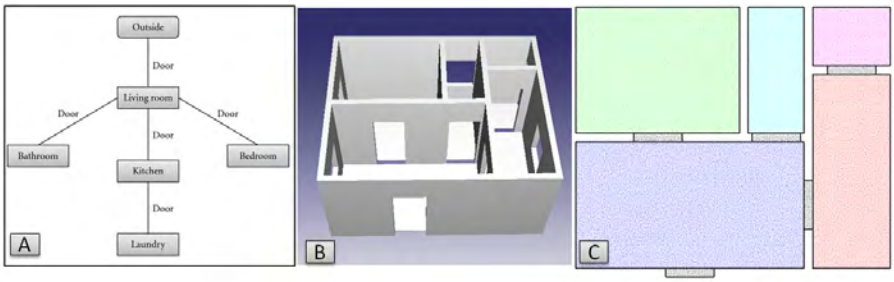


Figura 4.2 – Em (A), o grafo de conectividade e em (B) a casa utilizada para simulação de ambientes internos [41]. Em (C), apresenta-se a sua visualização no ambiente de simulação.

Considerando as informações do grafo de conectividade, é possível adequar as informações para definir o grafo a ser utilizado pelo algoritmo A* para definir o caminho a ser seguido pelas agentes.

Atualmente, os agentes executam diferentes ações nos ambientes da casa. Graças ao *path planning*, após definida a ação a ser executada, o agente se desloca até o local desejado. Após transcorrido o tempo de execução da ação, uma nova ação é definida e o agente se desloca até o local de execução da nova ação. Este ciclo é ilustrado na Figura 4.3: em (A), apresenta-se o grafo ligando os cômodos da casa e em (B) a locomoção dos agentes neste local. Em (C) e (D), apresentam-se agentes que se deslocam pela casa para realizar ações. A cor dos agentes está relacionada ao local que o agente deve realizar a ação.

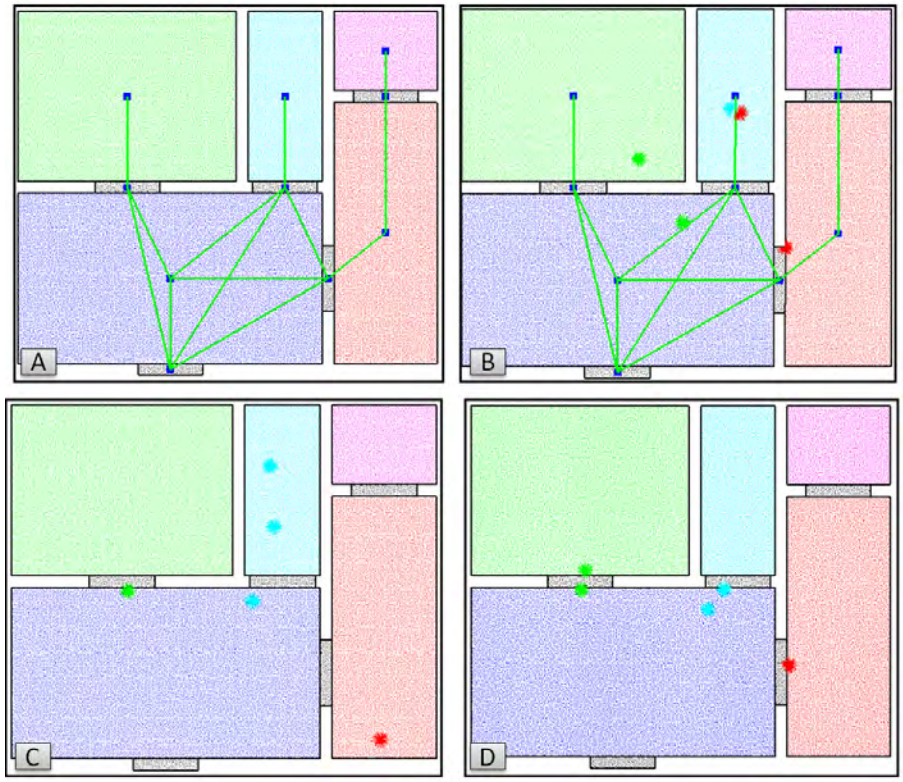


Figura 4.3 – Em (A), apresenta-se o grafo ligando os cômodos da casa e em (B) a locomoção dos agentes em relação ao grafo. Em (C) e (D), apresentam-se agentes que se deslocam pela casa para realizar ações.

Visando verificar a flexibilidade do *framework* proposto em [61], Tutenel *et al.* realizam a integração do mesmo com o gerador de ambientes residenciais proposto neste trabalho. A planta baixa fornecida pelo modelo é utilizada para gerar a geometria da casa e dispor os móveis de acordo com o uso previsto para cada peça. Toda a geometria foi criada automaticamente utilizando a gramática *CGA Shapes* proposta por [47]. A Figura 4.4a apresenta a visão do exterior da construção e a Figura 4.4a o seu interior, com os móveis distribuídos automaticamente pela abordagem proposta em [60].

4.2 Ambientes Amplos

Para validar o conceito de multidões reutilizáveis, introduzido na dissertação de mestrado intitulada "'Authoring Reusable Crowds using Semantics"' [37], Kraayenbrink utiliza como estudos de

(a) Vista externa da residência.

(b) Vista interna da residência com móveis dispostos pelo *layout solver*.

Figura 4.4 – Planta baixa gerada pelo modelo e renderizada por Tutenel.

caso cenários gerados através da definição de *templates* contendo a hierarquia dos espaços a serem criados e os serviços oferecidos pelos objetos presentes em cada ambiente amplo. A Figura 4.5 mostra o ambiente virtual utilizado em um dos estudos de caso de simulação de agentes. Já a Figura 4.6 mostra um *heat map* gerado após a simulação, o qual representa a movimentação de agentes durante a simulação. A representação tridimensional do ambiente pode ser vista nas figuras 4.7a e 4.7b.

Outro trabalho que utiliza as informações semânticas aplicadas à simulação de multidões é o de Hocevar [27], intitulado *Simulation of Group Behavior using Semantic Virtual Environments*. As informações semânticas do ambiente amplo podem ser utilizadas para criar automaticamente populações específicas, denominadas de *Population Class* (PC), como proposto em [27]. A partir dessas informações, os agentes da simulação tendem a ficar próximos de objetos que fornecem algum

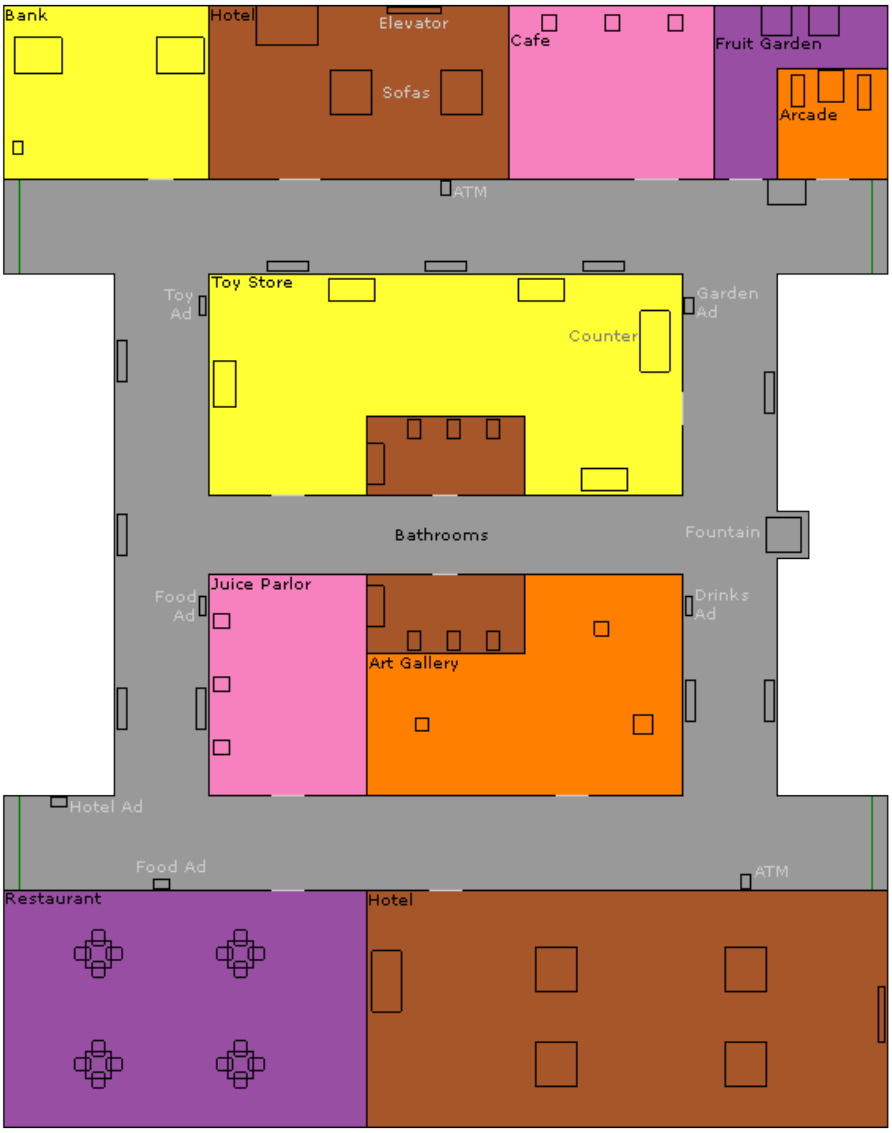


Figura 4.5 – Planta baixa de um ambiente amplo utilizado como estudo de caso por Kraayenbrink [37].

serviço específico, de acordo com o interesse de cada agente. Para a movimentação dos agentes, livres de colisão, utiliza-se o modelo apresentado por Bicho [14]. Um exemplo de simulação gerado por Hocevar pode ser visto na Figura 4.8. A mesma simulação de humanos virtuais é representada nas Figuras 4.9 e 4.10 utilizando o *framework* CrowdVis proposto por Braun [7].

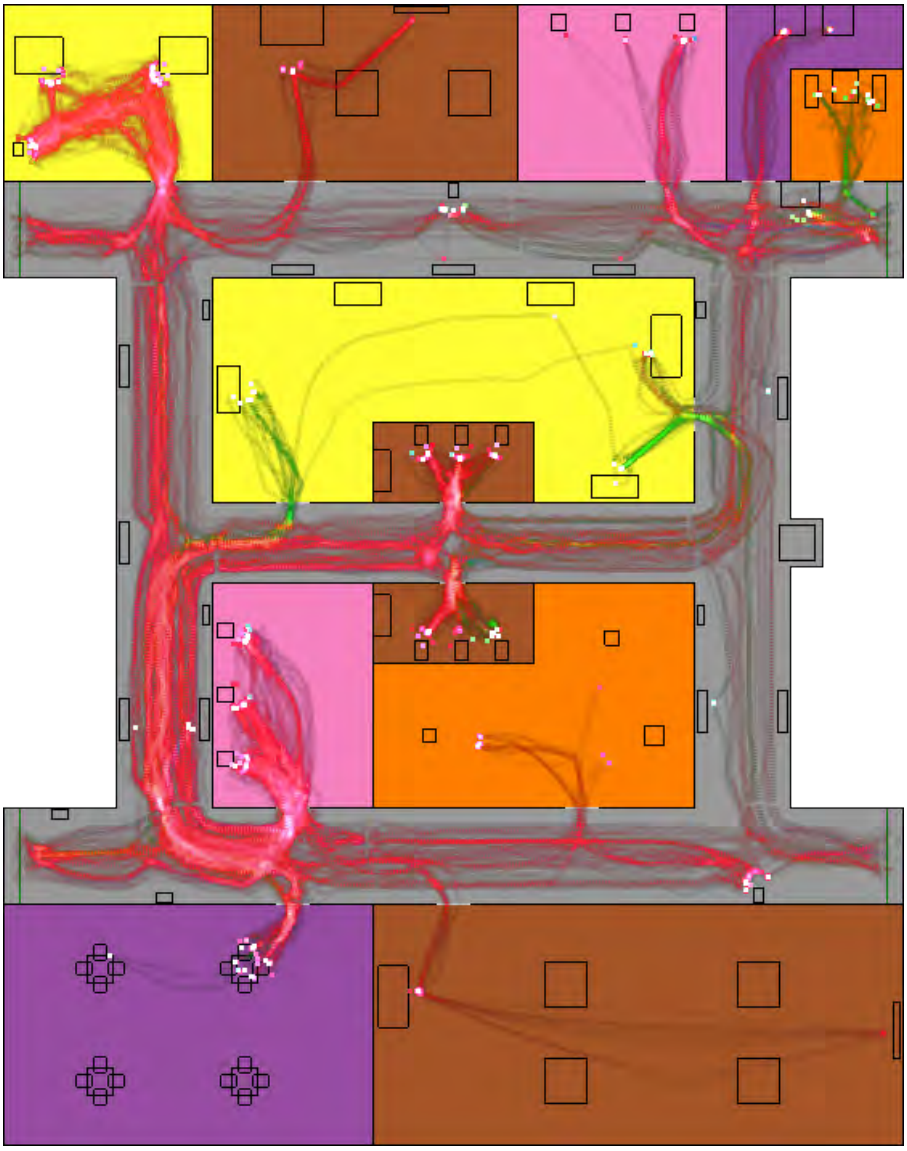
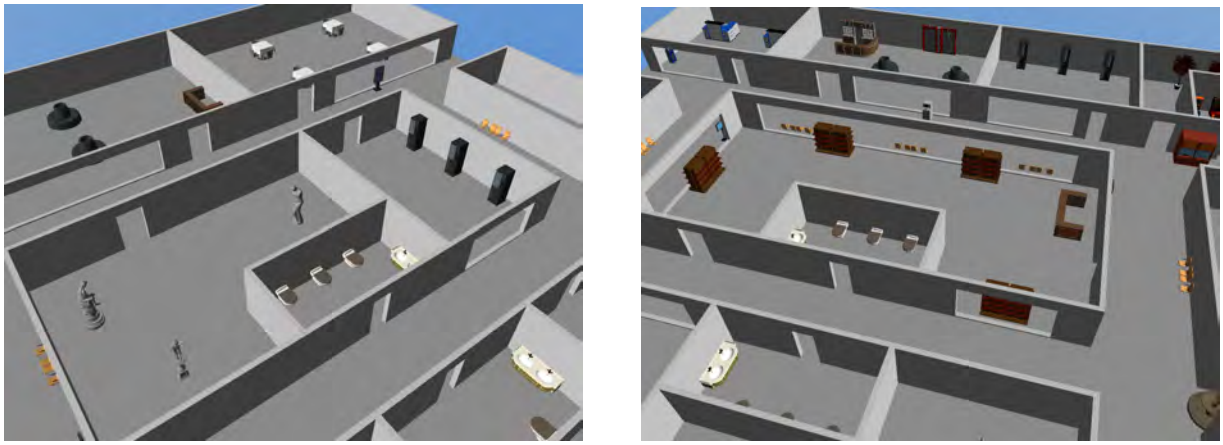


Figura 4.6 – *Heap map* de uma simulação de agentes virtuais realizada por Kraayenbrink [37] utilizando o mesmo ambiente ilustrado anteriormente.



(a) (b)

Figura 4.7 – Captura de tela da representação tridimensional do ambiente amplo apresentado na Figura 4.5.

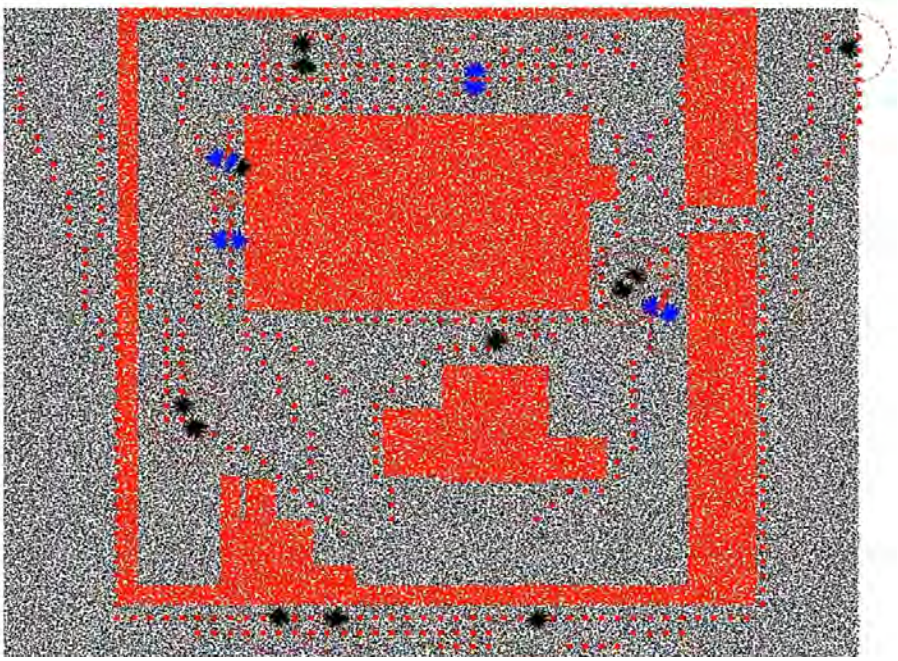


Figura 4.8 – Marcadores de um ambiente amplo em uma simulação de agentes virtuais especificada por Hocevar [27].

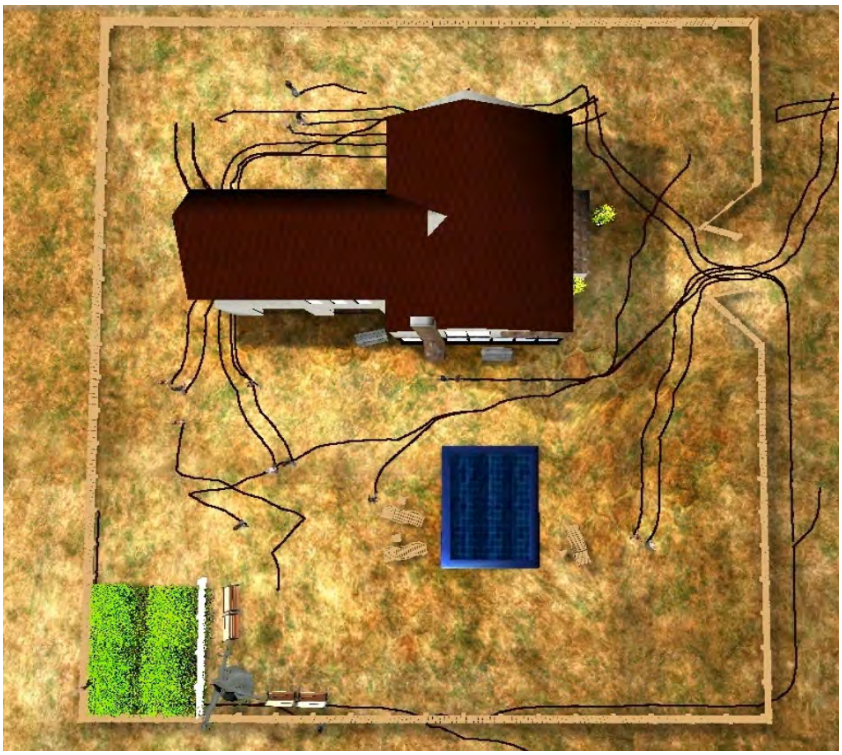


Figura 4.9 – Visualização 2D da simulação de agentes virtuais especificada por Hocevar utilizando o *framework* CrowdVis [7].

Figura 4.10 – Captura de tela da visualização tridimensional dos agentes realizando a simulação.

5. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Este trabalho apresentou um modelo computacional para a geração procedural de ambientes virtuais semânticos. O principal objetivo do trabalho foi o de automatizar a geração de ambientes com a inclusão de informações semânticas, permitindo que os mesmos possam ser utilizados em simulações de agentes virtuais autônomos e jogos. Atualmente, ainda existem poucos trabalhos na literatura que lidam com a questão de geração de ambientes virtuais internos, sendo este o foco do trabalho. A solução proposta lida com ambientes residenciais, como casas e apartamentos, e ambientes amplos (*shopping centers* e aeroportos, por exemplos). Para outros casos é possível a criação de instâncias de forma manual, permitindo que o modelo atenda ambientes externos e configurações internas de maior complexidade.

A entrada inicial do modelo proposto é dada por um conjunto de informações semânticas que sintetizam um modelo mental do ambiente pensado pelo usuário. Este conjunto de informações diz respeito ao tipo de construção a ser criada e como suas partes se relacionam com o todo. Através do conceito de entidades [36], como o de espaços e objetos tangíveis, é criada uma hierarquia espacial, onde o ambiente virtual é decomposto em espaços mais simples. Isto se aplica tanto para ambientes amplos como para ambientes residenciais. Uma casa, por exemplo, pode ser decomposta inicialmente em três espaços distintos: área social, área de serviço e área privativa. Cada uma destas áreas ou espaços pode ser decomposta novamente, criando assim hierarquia de espaços.

Além de espaços, é possível definir também a criação de objetos tangíveis. Estes objetos são utilizados para preencher o ambiente virtual. Da mesma forma que acontece em ambientes reais, estes objetos podem apresentar geometrias distintas, de acordo com o estilo arquitetônico ou ambiente em que são utilizados. Assim, paredes e pisos podem assumir aparências diferentes, dependendo do local onde estão inseridos. O mesmo pode ser dito sobre os objetos de decoração e utilitários, como mesas, bancos e armários.

As informações semânticas também dizem respeito aos usos e funcionalidades dos espaços e dos objetos. Desta forma, é possível definir que um determinado espaço serve como quarto e assim colocar móveis e objetos que tenham relação com um quarto [61]. Agentes virtuais também se beneficiam deste tipo de informação, pois em um dado momento, pode ser necessário a execução de uma tarefa que demande um local específico, como, por exemplo, ir ao banheiro para tomar banho. A camada semântica presente neste modelo permite que agentes obtenham esse tipo de informação através do *framework* semântico integrado [36].

De posse da descrição semântica, o módulo de subdivisão espacial trata de realizar criação dos espaços do ambiente requisitado. Este processo é feito utilizando dois algoritmos já existentes na literatura, mas que foram adaptados para a necessidade do trabalho. O primeiro algoritmo é baseado no *squarified treemaps*. Ele é utilizado para realizar divisões em ambientes internos residenciais, como discutido na Seção 3.3.1. O resultado final é adequado para geração de plantas baixas residenciais, de forma muito semelhante às plantas baixas comerciais disponíveis. Neste algoritmo foi adicionado

uma segunda etapa, que resolve situações onde alguma das peças criadas não possua conexão com nenhuma outra, tornando-a inacessível. Usando a estrutura das paredes, é encontrado um caminho (corredor) que crie essa ligação.

Para realizar a subdivisão do espaço em ambientes amplos, é utilizado o algoritmo de *straight skeleton*. Este algoritmo gera um corredor inicial, o qual dará início ao processo de alocação dos espaços com base na hierarquia espacial contida nas definições semânticas. A diferença principal entre dos dois algoritmos reside no fato de que para o primeiro, os valores das área dos espaços a serem criados já são conhecidas, enquanto que no caso dos ambientes amplos ainda não. Outra questão fundamental diz respeito ao formato dos corredores. Em ambientes amplos normalmente são observados corredores longos e retos ou com leves curvas. Os corredores gerados nos ambientes residenciais buscam o menor caminho de ligação entre as peças, ocasionando corredores mais curtos e angulados.

Após a divisão do espaço o processo volta a ser único para ambos os tipos de ambientes, gerando a inclusão de marcadores e de grafos de conectividade, para que os mesmos possam ser utilizados em simulações de agentes virtuais autônomos. Esta aplicabilidade foi comprovada através dos ambientes ambientes em trabalhos desenvolvidos por Cassol [10] [11], Kraayenbrink [37] e Hocevar *et al.* [27].

Além da planta baixa, também é possível gerar a geometria tridimensional dos ambientes. Este módulo gera geometrias mais simples, adequadas para uma visualização rápida da construção gerada. Para a obtenção de uma geometria mais refinada, é possível usar as informações semânticas e da planta baixa como dados de entrada em um gerador especializado. No Capítulo 4 é possível ver duas imagens (Figuras 4.4a e 4.4b) ilustrando uma construção gerada por Tutenel *et al.* [61] através de uma *shape grammar* [47] que utiliza como base uma planta baixa criada com o modelo proposto.

Apesar dos resultados alcançados, existem várias questões que podem ser trabalhadas para aprimorar o modelo proposto. Entre elas pode-se destacar:

- geração de construções com mais de um andar e as conexões entre os andares;
- geração integrada do interior e com fachada de construções, fazendo com que a presença de elementos influencie na planta baixa gerada;
- aprimoramento do módulo de geração de geometria, a fim de obter resultados visualmente mais interessantes;
- realização de mais testes com o algoritmo de *straight skeleton* para verificar diferentes possibilidades de geração de planta baixa e dos corredores secundários.

Referências Bibliográficas

[1] Aichholzer, O., and Aurenhammer, F. Straight skeletons for general polygonal figures in the plane. In *Voronoi's Impact on Modern Science II*, A. Samoilenko, Ed. National Academy of Sciences of Ukraine, Kyiv, Ukraine, 1998, pp. 7–21.

[2] Aichholzer, O., D.Alberts, Aurenhammer, F., and Gärtner, B. A novel type of skeleton for polygons. *Journal of Universal Computer Science 1*, 12 (December 1995), 752–761. Springer Verlag.

[3] Badawi, M., and Donikian, S. The generic description and management of interaction between autonomous agents and objects in an informed virtual environment. *Computer Animation and Virtual Worlds 18*, 4-5 (2007), 559–569.

[4] Belhadj, F. Terrain modeling: a constrained fractal model. In *AFRIGRAPH '07: Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa* (New York, NY, USA, 2007), ACM, pp. 197–204.

[5] Belhadj, F., and Audibert, P. Modeling landscapes with ridges and rivers: bottom up approach. In *GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia* (New York, NY, USA, 2005), ACM, pp. 447–450.

[6] Benner, J., Geiger, A., and Leinemann, K. Flexible generation of semantic 3d building models. In *1st International Workshop on Next Generation 3D City Models* (Bonn, 2005).

[7] Braun, H. A framework for real time crowd visualization. Master's thesis, Pontifícia Universidade Católica do Rio Grande do Sul, 2012.

[8] Bruls, M., Huizing, K., and van Wijk, J. Squarified treemaps. In *Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization* (2000), Press, pp. 33–42.

[9] Bélanger, D. Designing roofs of buildings, 2000. Disponível em: <http://www.sable.mcgill.ca/ dbelan2/roofs/roofs.html>. Acessado em 20 de agosto de 2016.

[10] Cassol, V. Simulação de grupos de agentes virtuais utilizando raciocínio de terrenos. Master's thesis, Pontifícia Universidade Católica do Rio Grande do Sul, 2011.

[11] Cassol, V. J., Marson, F. P., Vendramini, M., Paravisi, M., Bicho, A. L., Jung, C. R., and Musse, S. R. Simulation of autonomous agents using terrain reasoning. In *Proc. the Twelfth IASTED International Conference on Computer Graphics and Imaging (CGIM 2011)* (Innsbruck, Austria, 2011), IASTED/ACTA Press.

[12] da Silveira, L. G., and Musse, S. R. Real-time generation of populated virtual cities. In *VRST '06: Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2006), ACM, pp. 155–164.

[13] DAmbrosio, D., Gregorio, S., Gabriele, S., and Gaudio, R. A cellular automata model for soil erosion by water. *Physics and Chemistry of the Earth, Part B: Hydrology, Oceans and Atmosphere 26* (2001), 33–39(7).

[14] de Lima Bicho, A. *Da modelagem de plantas à dinâmica de multidões: um modelo de animação comportamental bio-inspirado*. PhD thesis, Universidade Estadual de Campinas - UNICAMP, Campinas, 2009.

[15] Dollner, J., Buchholz, H., Brodersen, F., Glander, T., Jutterschenke, S., and Klimetschek, A. Smart buildings: A concept for ad-hoc creation and refinement of 3d building models. In *1st International Workshop on Next Generation 3D City Models* (Bonn, 2005).

[16] Dörner, R., Göbel, S., Effelsberg, W., and Wiemeyer, J. *Serious Games: Foundations, Concepts and Practice*. Springer International Publishing, 2016.

[17] Ebert, D. S., Musgrave, F. K., Peachey, D., Perlin, K., and Worley, S. *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[18] Farenc, N., Boulic, R., and Thalmann, D. An informed environment dedicated to the simulation of virtual humans in urban context. *Computer Graphics Forum 18*, 3 (1999), 309–318.

[19] Finkenzeller, D. Detailed building facades. *IEEE Computer Graphics and Applications 28*, 3 (2008), 58–66.

[20] Finkenzeller, D., and Schmitt, A. Rapid modeling of complex building façades. In *Short paper proceedings of Eurographics 2006* (2006), D. Fellner and C. Hansen, Eds., pp. 95–98.

[21] Gaildrat, V. Declarative modelling of virtual environments, overview of issues and applications. In *Proceedings of the* 10^{th} *International Conference on Computer Graphics and Artificial Intelligence (3IA 2007)* (Athens, Greece, May 2007), pp. 5–15.

[22] Grimaldo, F., Lozano, M., Barber, F., and Vigueras, G. Simulating socially intelligent agents in semantic virtual environments. *The Knowledge Engineering Review 23*, 04 (2008), 369–388.

[23] Gutierrez, M., Vexo, F., and Thalmann, D. Semantics-based representation of virtual environments. *International Journal of Computer Applications in Technology 23*, 2-4 (2005), 229–38. Virtual Reality Lab., Ecole Polytech. Fed. de Lausanne, Switzerland.

[24] Hagedorn, B., and Dollner, J. High-level web service for 3d building information visualization and analysis. In *GIS '07: Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems* (New York, NY, USA, 2007), ACM, pp. 1–8.

[25] Hahn, E., Bose, P., and Whitehead, A. Persistent realtime building interior generation. In *Sandbox '06: Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames* (New York, NY, USA, 2006), ACM, pp. 179–186.

[26] Hart, P., Nilsson, N., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on 4*, 2 (july 1968), 100 –107.

[27] HOCEVAR, R., MARSON, F., CASSOL, V., BRAUN, H., BIDARRA, R., and MUSSE, S. From their environment to their behavior: a procedural approach to model groups of virtual agents. TBP in Proccedings of IVA 2012 - 12th International Conference on Intelligent Virtual Agents, 12-14 September, Santa Cruz, CA, 2012.

[28] Honda, M., Mizuno, K., Fukui, Y., and Nishihara, S. Generating autonomous time-varying virtual cities. *Cyberworlds, 2004 International Conference on* (Nov. 2004), 45–52.

[29] Horna, S., Damiand, G., Meneveaux, D., and Bertrand, Y. Building 3d indoor scenes topology from 2d architectural plans. In *GRAPP (GM/R)* (2007), J. Braz, P.-P. Vázquez, and J. M. Pereira, Eds., INSTICC - Institute for Systems and Technologies of Information, Control and Communication, pp. 37–44.

[30] Jiang, H., Xu, W., Mao, T., Li, C., Xia, S., and Wang, Z. A semantic environment model for crowd simulation in multilayered complex environment. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2009), VRST '09, ACM, pp. 191–198.

[31] Johnson, B., and Shneiderman, B. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *VIS '91: Proceedings of the 2nd conference on Visualization '91* (Los Alamitos, CA, USA, 1991), IEEE Computer Society Press, pp. 284–291.

[32] Kallmann, M. *Object interaction in real-time virtual environments*. PhD thesis, Infoscience | Ecole Polytechnique Federale de Lausanne [http://infoscience.epfl.ch/oai2d.py] (Switzerland), 2001.

[33] Kallmann, M., and Thalmann, D. Modeling objects for interaction tasks. In *Proceedings of the Eurographics Workshop on Animation and Simulation* (Lisbon, Portugal, August-September 1998), pp. 73–86.

[34] Kallmann, M., and Thalmann, D. Direct 3D interaction with smart objects. In *Proceedings of the ACM symposium on Virtual reality software and technology* (London, UK, December 1999), VRST '99, ACM, pp. 124–130.

[35] Kelly, T., and Wonka, P. Interactive architectural modeling with procedural extrusions. *ACM Trans. Graph. 30*, 2 (Apr. 2011), 14:1–14:15.

[36] Kessing, J., Tutenel, T., and Bidarra, R. Designing semantic game worlds. In *Proceedings of the third workshop on Procedural Content Generation in Games (PCG 2012)* (Raleigh, NC, USA, May 2012).

[37] Kraayenbrink, N. Authoring reusable crowds using semantics. Master's thesis, Delft University of Technology, 2011.

[38] Larive, M., and Gaildrat, V. Wall grammar for building generation. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia* (New York, NY, USA, 2006), ACM, pp. 429–437.

[39] Lopes, R., Tutenel, T., Smelik, R. M., de Kraker, K. J., and Bidarra, R. A constrained growth method for procedural floor plan generation. In *Proceedings of GAME-ON 2010, the* 11^{th} *International Conference on Intelligent Games and Simulation* (Leicester, UK, November 2010), EUROSIS.

[40] Luck, M., and Aylett, R. Applying artificial intelligence to virtual reality: Intelligent virtual environments. *Applied Artificial Intelligence 14*, 1 (2000), 3–32.

[41] Marson, F., and Musse, S. R. Automatic generation of floor plans based on squarified treemaps algorithm. *IJCGT International Journal on Computers Games Technology 2010* (January 2010), 1–10.

[42] Marson, F. P. Modelo paramétrico de cidades virtuais populáveis. Master's thesis, Universidade do Vale do Rio dos Sinos, 2004.

[43] Marson, F. P., Musse, S., and Jung, C. R. Modelagem procedural de cidades virtuais. In *VI Symposium on Virtual Reality* (Ribeirão Preto, 2003), Sociedade Brasileira de Computação, COC.

[44] Martin, J. Procedural house generation: A method for dynamically generating floor plans. In *Symposium on Interactive 3D Graphics and Games* (2006).

[45] Merrell, P., Schkufza, E., Li, Z., Agrawala, M., and Koltun, V. Interactive furniture layout using interior design guidelines. In *SIGGRAPH '11: Proceedings of the* 38^{th} *Annual Conference on Computer Graphics and Interactive Techniques* (Vancouver, Canada, August 2011), ACM, pp. 87:1–87:10.

[46] Miller, G. S. P. The definition and rendering of terrain maps. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1986), ACM, pp. 39–48.

[47] Muller, P., Wonka, P., Haegler, S., Ulmer, A., and Gool, L. V. Procedural modeling of buildings. *ACM Trans. Graph. 25*, 3 (2006), 614–623.

[48] Muller, P., Zeng, G., Wonka, P., and Gool, L. V. Image-based procedural modeling of facades. *ACM Trans. Graph. 26*, 3 (2007), 85.

[49] Musgrave, F. K., Kolb, C. E., and Mace, R. S. The synthesis and rendering of eroded fractal terrains. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1989), ACM, pp. 41–50.

[50] Ong, T. J., Saunders, R., Keyser, J., and Leggett, J. J. Terrain generation using genetic algorithms. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation* (New York, NY, USA, 2005), ACM, pp. 1463–1470.

[51] Otto, K., and Berlin, F. U. Towards semantic virtual environments. In *In Workshop Towards Semantic Virtual Environments* (2005), pp. 47–56.

[52] Paiva, D., Vieira, R., and Musse, S. Ontology-based crowd simulation for normal life situations. *Computer Graphics International 2005* (June 2005), 221–226.

[53] Parish, Y. I. H., and Muller, P. Procedural modeling of cities. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 301–308.

[54] Perlin, K. An image synthesizer. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1985), ACM, pp. 287–296.

[55] Peters, C., Dobbyn, S., MacNamee, B., and O'Sullivan, C. Smart objects for attentive agents. In *Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG '03)* (Plzen - Bory, Czech Republic, February 2003), pp. 1–8.

[56] Prusinkiewicz, P., and Lindenmayer, A. *The algorithmic beauty of plants*. Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[57] Rishe, N. *Database design: the semantic modeling approach*. McGraw-Hill, Inc., New York, NY, USA, 1992.

[58] Thomas, G., and Donikian, S. Modelling virtual cities dedicated to behavioural animation. *Computer Graphics Forum 19*, 3 (2000), 71–80.

[59] Tutenel, T., Bidarra, R., Smelik, R. M., and de Kraker, K. J. The role of semantics in games and simulations. *ACM Computers in Entertainment 6* (2008), 1–35.

[60] Tutenel, T., Smelik, R. M., Bidarra, R., and de Kraker, K. J. A semantic scene description language for procedural layout solving problems. In *AIIDE* (2010), G. M. Youngblood and V. Bulitko, Eds., The AAAI Press.

[61] Tutenel, T., Smelik, R. M., Lopes, R., de Kraker, K. J., and Bidarra, R. Generating consistent buildings: a semantic approach for integrating procedural techniques. *IEEE Transactions on Computational Intelligence and AI in Games 3*, 3 (2011), 274–288.

[62] Wonka, P., Wimmer, M., Sillion, F., and Ribarsky, W. Instant architecture. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), ACM, pp. 669–677.

[63] Yap, C., H.Biermann, Hertzman, A., Li, C., Meyer, J., Pao, H., and Paxia, T. A different manhattan project: Automatic statistical model generation. In *Proc. 14th Ann. Symp., Electronic Imaging 2002* (San Jose, California, USA, 2002), SPIE, pp. 259–268.

[64] Yu, L.-F., Yeung, S. K., Tang, C.-K., Terzopoulos, D., Chan, T. F., and Osher, S. Make it home: Automatic optimization of furniture arrangement. In *SIGGRAPH '11: Proceedings of the* 38^{th} *Annual Conference on Computer Graphics and Interactive Techniques* (Vancouver, Canada, August 2011), ACM, pp. 86:1–86:12.

A. Publicações

Este apêndice apresenta as publicações obtidas e submetidas durante o decorrer do doutorado.

A.1 Resumo em Anais de Conferência

CASSOL, Vinícius; MARSON, Fernando; VENDRAMINI, Mateus; PARAVISI, Marcelo; BICHO, Alessandro; JUNG, Cláudio and MUSSE, Soraia. *Computing Agents Motion using Terrain Reasoning*. In: 10th International Conference on Intelligent Virtual Agents, 2010, Philadelphia. Booklet of short papers of 10th International Conference on Intelligent Virtual Agents, 2010. p. 10-11.

A.2 Artigos em Anais de Conferências

CASSOL, Vinícius; MARSON, Fernando and MUSSE, Soraia. *Procedural Hair Generation*. In: SBGames: Simpósio Brasileiro de Jogos e Entretenimento Digital, 2009, Rio de Janeiro. Anais do SBGames, 2009. p. 185-190.

CASSOL, Vinícius; MARSON, Fernando; VENDRAMINI, Mateus; PARAVISI, Marcelo; BICHO, Alessandro; JUNG, Cláudio and MUSSE, Soraia. *Simulation of Autonomous Agents Using Terrain Reasoning*. In: Computer Graphics and Imaging - CGIM 2011, Innsbruck - Austria.

HOCEVAR, Rafael; MARSON, Fernando; CASSOL, Vinícius; BRAUN, Henry; BIDARRA, Rafael and MUSSE, Soraia. *From their environment to their behavior: a procedural approach to model groups of virtual agents*. Proccedings of IVA 2012 - 12th International Conference on Intelligent Virtual Agents, 12-14 September, Santa Cruz, CA

KRAAYENBRINK, Nick; KESSING, Jassin; TUTENEL, Tim; HAAN, Gerwin; MARSON, Fernando; MUSSE, Soraia Raupp; BIDARRA, Rafael. *Semantic crowds: reusable population for virtual worlds*. In: VS-Games, 2012, Genova. Proceedings of VS-Games 2012, 2012.

FLACH, Laura; MARSON, Fernando; CASSOL, Vinícius; MUSSE, Soraia Raupp. *A Procedural Approach to Simulate Virtual Agents Behaviours in Indoor Environments*. In: Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGAMES), 2013, São Paulo. Proceedings of SBGames 2013, 2013. v. 1.

BRAUN, Henry; MARSON, Fernando; HOCEVAR, Rafael; CASSOL, Vinícius; MUSSE, Soraia Raupp . *CrowdVis: A Framework For Real Time Crowd Visualization*. In: ACM Symposium on Applied Computing (SAC), 2013, Coimbra. Proceedings of SAC, 2013. v. 1.

A.3 Artigo em Periódico

MARSON, Fernando and MUSSE, Soraia. *Automatic Real-Time Generation of Floor Plans Based on Squarified Treemaps Algorithm*, International Journal of Computer Games Technology, vol. 2010, Article ID 624817, 10 pages, 2010. doi:10.1155/2010/624817

B. Listagem dos Arquivos de Informações Semânticas

Listing B.1 – Exemplo de um arquivo configuração.

```
complexInfra
database case2
template shopping.template
customization case2.custom
objects case2.objects
corridorWidth 16 m
wallThickness 0.3 m
wallHeight 4.2 m
doorHeight 3 m
```

Listing B.2 – Exemplo de um arquivo de objetos.

```
bench bench1 ..\complexInfra\3DModels\benches\bench.obj shopping 10 0 17.5 3.14159274
bench bench2 ..\complexInfra\3DModels\benches\bench.obj shopping 20 0 17.5 3.14159274
bench bench3 ..\complexInfra\3DModels\benches\bench.obj shopping 29 0 17.5 3.14159274
bench bench4 ..\complexInfra\3DModels\benches\bench.obj shopping 40 0 17.5 3.14159274
bench bench5 ..\complexInfra\3DModels\benches\bench.obj shopping 50 0 17.5 3.14159274

atm atm_h1 ..\complexInfra\3DModels\atm\atm3.obj shopping 29 0 12.7 0

arcade arcade1 ..\complexInfra\3DModels\arcade\arcade1a.obj arcade 51.3 0 6 0
arcade arcade2 ..\complexInfra\3DModels\arcade\arcade2a.obj arcade 53.4 0 6 0
arcade arcade3 ..\complexInfra\3DModels\arcade\arcade3a.obj arcade 55.5 0 6.5 0

coffeemachine coffeemachine1 ..\complexInfra\3DModels\coffee\coffee.obj coffee 35.5 0 2 0
coffeemachine coffeemachine2 ..\complexInfra\3DModels\coffee\coffee.obj coffee 39.5 0 2 0
coffeemachine coffeemachine3 ..\complexInfra\3DModels\coffee\coffee.obj coffee 43.5 0 2 0

reception reception_h1 ..\complexInfra\3DModels\reception\reception1.obj hotel 18 0 2.6 0
elevator elevator_h1 ..\complexInfra\3DModels\elevator\elevator.obj hotel 27 0 1.3 0
sofa roundsofa1_h1 ..\complexInfra\3DModels\sofa\roundsofa.obj hotel 23 0 6.5 0
sofa roundsofa2_h1 ..\complexInfra\3DModels\sofa\roundsofa.obj hotel 30 0 6.5 0

counterbank leftcounter ..\complexInfra\3DModels\bank\counter.obj bank 4 0 4 0
counterbank rightcounter ..\complexInfra\3DModels\bank\counter.obj bank 11 0 4 0
```

Listing B.3 – Exemplo de um arquivo de *template* ou de customização.

```
unit m2
unit m
space shopping
        space cinema shopping
                space screen cinema
                        as provides fun movie
                        an footage 400 m2 minValue 350 m2 maxValue 500 m2
                        as quantity few
                space boxOffice cinema
                        as provides movieTicket
                        an footage 80 m2 minValue 50 m2 maxValue 100 m2

        space foodCourt shopping
                space tables foodCourt
                        as spaceType flexible
```

```
                an percentage 45 %
        space snackstore foodCourt
                as provides food beverage pizza hotdog
                an footage 40 m2 minValue 30 m2 maxValue 60 m2
                as quantity few
        space coffeeshop foodCourt
                as provides snacks coffee
                as quantity few
                an footage 30 m2 minValue 20 m2 maxValue 50 m2
        space restroom foodCourt
                as provides toilet
                an footage 60 m2 minValue 50 m2 maxValue 120 m2

space service shopping
        space drugstore service
                as provides medicine health
                an footage 65 m2 minValue 50 m2 maxValue 90 m2
        space grocery service
                as provides cookie fruit vegetable meat milk
                an footage 120 m2 minValue 90 m2 maxValue 200 m2
        space bank service
                as provides cash
                an footage 90 m2 minValue 50 m2 maxValue 160 m2
                as quantity few

space bigstores shopping
        ab splittable true
        space Department_Store bigstores
        as provides cloth shoes gift furniture bathCloth bedCloth
        an footage minValue 1000 m2 maxValue 2000 m2
        as quantity few

space smallstores shopping
        ab stripArea true
        ab splittable true
        ab growable false
        space toystore smallstores
                as provides toy gift
                an footage 80 m2 minValue 70 m2 maxValue 130 m2

        space clothstore smallstores
                as quantity several
                an footage 80 m2 minValue 60 m2 maxValue 150 m2
                as provides cloth shoe gift

        space jewelry smallstores
                as provides glasses jewel watch gift
                an footage 50 m2 minValue 30 m2 maxValue 100 m2

        space bookstore smallstores
                as provides book magazine culture information gift
                an footage 140 m2 minValue 100 m2 maxValue 200 m2
```