UNIVERSIDADE DO VALE DO RIO DOS SINOS
CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA INTERDISCIPLINAR DE PÓS-GRADUAÇÃO EM
COMPUTAÇÃO APLICADA

Simulação de Forças-Sociais em Ambientes Tridimensionais

por

OTÁVIO CORRÊA CORDEIRO

Dissertação submetida a avaliação como requisito parcial para a obtenção do grau de Mestre em Computação Aplicada

Orientadora: Prof^a. Dr^a. Soraia Raupp Musse

São Leopoldo, Janeiro de 2007

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Cordeiro, Otávio Corrêa

Simulação de Forças-Sociais em Ambientes Tridimensionais / por Otávio Corrêa Cordeiro. — São Leopoldo: Ciências Exatas e Tecnológicas da UNISINOS, 2007.

106 f.: il.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos. Ciências Exatas e Tecnológicas Programa Interdisciplinar de Pós-Graduação em Computação Aplicada, São Leopoldo, BR–RS, 2007. Orientador: Musse, Soraia Raupp.

I. Musse, Soraia Raupp. II. Título.

"Olhe, lá vejo meu pai

Vejo minha mãe, irmãs e irmãos.

Olhe, lá vejo meus ancestrais desde o início.

Eles clamam meu nome,

pedem que eu tome meu lugar entre eles,

nos Salões do Valhalla,

onde os bravos vivem para sempre."

– Oração dos guerreios na mitologia Nórdica que,
mortos em batalhas, eram levados a Valhalla
pelas Valquírias e lá permaneciam como heróis de guerra
sob os cuidados de Odin até o Ragnarok.

Agradecimentos

Agradeço meus pais, por todo apoio e amor que sempre dedicaram à mim e meu irmão, respeitando nossas escolhas e atitudes, por acreditarem em mim e me mostrarem que sou capaz de superar desafios maiores do que supunha.

Aos meus *segundos pais*, Valério e Rachel, que me acolheram quando precisei na minha *segunda casa*, me dando todo apoio logo que saí da casa dos meus pais.

Agradeço a Luana Dummer, meu amor, por toda cumplicidade, por estar sempre ao meu lado, principalmente nos momentos difíceis, por me ligar todos os dias pela manhã para me acordar e por entender os momentos que deixei de estar presente.

Aos amigos que estão longe mas que fazem parte da minha vida: Rodrigo, o "homem da TV" e meu irmão, pelo carinho que tem por mim e por me agüentar nos últimos 24 anos; o Decko, que mesmo lá no interior de São Paulo certamente é uma das pessoas mais amigas e dedicadas que tive o prazer de conhecer; o Bruno e a Dionísia que mesmo sabendo das minhas responsabilidades com o mestrado, nunca deixaram de me procurar e convidar para encontros *nerds* e divertidos; e ao Heinar, pelas discussões filosóficas em mensageiros instantâneos, e minha linda licença do Mathematica.

Aos colegas da Unisinos, César, Lucas, Cícero, Sidnei, Paulo e Felipe, pelas inúmeras horas que convivemos nos últimos dois anos. Por todos os momentos de diversão, refrigerantes no meio da tarde, almoços no *shopping*, dicas de programação, música e por me lembraram que não há idade de para encontrar diversão em jogos de *videogame*.

Agradeço ao professor Gerson Cavalheiro, por abrir as portas da Computação para mim. Obrigado por toda a ajuda, amizade e compreensão. Pelas viagens para congressos, jantas em restaurantes de qualidade duvidosa, por estar sempre disponível e por acreditar em mim, mesmo quando nem eu mesmo acreditava.

Aos professores Cláudio Jung, Marinho Barcelos, Marcelo Walter e Ney Lemke, que sempre se mostraram dispostos a colaborar com meu trabalho e pelas conversas sempre descontraídas e produtivas que tive com eles.

Por último agradeço a minha orientadora, Soraia Raupp Musse, por cada momento que dedicou à me orientar. Por sua confiança, suporte técnico e financeiro, seu exemplo de amor àquilo que faz e por sua cobrança constante. Sem ela este modelo jamais teria saído do papel.

Resumo

Os atuais centros urbanos concentram uma profusão de indivíduos localizados em estruturas e edificações complexas, como ginásios esportivos, centros comerciais, aeroportos, etc. Estes ambientes são claramente edificações de múltiplos andares e não podem ser representados por uma solução bidimensional em simulações baseadas em Física. Este grau de liberdade extra àqueles suportados pelos modelos atuais de simulação, encontrados na literatura, é a limitação destes em simular agentes em terrenos irregulares ou na presença de rampas e escadas.

O trabalho desenvolvido nesta dissertação encontra-se vinculado ao projeto CSHuV, Centro de Simulação de Humanos Virtuais – em colaboração com a HP Brazil R&D –, inserido no contexto do Laboratório CROMOS. Este projeto vem, nos últimos anos, modelando, implementando e validando diferentes ambientes para criação e simulação de humanos virtuais. Como resultado desta dissertação, é oferecido o suporte necessário ao projeto para simular ambientes ainda mais complexos através de formalismo físico e matemático.

Para isso, é proposta uma expansão do modelo de forças sociais, permitindo aos agentes deslocamentos espaciais dentro dos limites físicos do ambiente. Tais limites são introduzidos através de um novo conjunto de termos adicionados a equação diferencial do modelo original.

Palavras-chave: Computação Gráfica, Agentes Virtuais, Agentes Autônomos, Modelagem e Simulação, Programação Concorrente, Multiprogramação Leve.

TITLE: "SOCIAL FORCES SIMULATIONS IN THREE-DIMENSIONAL ENVIRONMENTS"

Abstract

The current urban center concentrate a large number of people located in complex buildings, as shopping centers, airports, etc. These environments are cleary buildings of multiple floors and can not be represented by a bidimensional solution of Physics simulations. This extra freedom degree is the limitation of these 2D models in simulation agents in irregular lands or at the presence of slopes and stairs.

The work developed in this master's thesis is inserted on the CSHuV project, Centro de Simulação de Humanos Virtuais – in colaboration with the HP Brazil R&D –, in the context of CROMOS Laboratory. This project comes, in the last years, modeling, implementing and validating different environments for creation and simulation of virtual human agents. As result of this master's thesis, the support necessary to the project to simulate more complex environments through physical and mathematical formalism is offered.

An expansion of the model of social forces is proposed, allowing the agent space displacements inside of the physical limits of the environment. Such limits are introduced through a new set of terms are added to the diferencial equation of the original model.

Keywords: crowd, simulation, computer graphics.

Sumário

Lista de Abreviaturas

CSHuV Centro de Simulação de Humanos Virtuais

FOSS Free and Open Source Software

GNU GNU is Not Unix

HP Hewlett-Packard

MPI Message Passing Interface

MPMD Multiple Program Multiple Data

PAD Processamento de Alto Desempenho

NASA National Aeronautics and Space Administration

POSIX Portable Operating System Interface

PVM Parallel Virtual Machine

PC Personal Computer

SMP Symetric Multiprocessor

SPMD Single Program Multiple Data

XML eXtensible Markup Language

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Dados referentes ao recenseamento realizado pelo *United States Census Bureau* revelaram para o mês de Outubro de 1999 uma população mundial de seis bilhões de habitantes, sendo o último bilhão obtido em apenas 12 anos (de acordo com resultados anteriores de cinco bilhões em 1987). A alta taxa de crescimento populacional prossegue atualmente, e a mesma organização apresentou os resultados aproximados para Janeiro de 2007 de 6,56 bilhões de indivíduos[1].

Este crescimento populacional abrupto reflete-se, potencialmente, nas grandes cidades e grandes centros industriais, na maioria das vezes, na busca por melhores oportunidades de emprego e melhores condições de vida. Desta forma, estes centros vêm sofrendo com este êxodo, nem sempre lucrativo. Locais como estações de transporte público – terminais rodoviários e estações de metrô – , calçadas nas proximidades dos centros comerciais, bancos e mesmo *shopping center* acabam apresentando características relacionadas a grandes aglomerados, interferindo diretamente na segurança e no conforto da população.

Inúmeros acidentes envolvendo multidões são reportados [Crowd Dynamics, 2006] anualmente. Nestes acidentes, nem sempre as vítimas se restringem a feridos, resultando, muitas vezes, em fatalidades. Situações como estas foram observadas, por exemplo em 1982, quando 340 pessoas morreram ao tentar

[1] Os dados estatísticos do *U. S. Census Bureau* estão disponíveis em `http://www.census.gov/` e em atualização contínua.

deixar um estádio de futebol em Moscovo, Rússia. Também em outro estádio, 83 morreram e 180 ficaram feridos em 1996 na Guatemala, em uma partida amistosa entre o país e Costa Rica. Em 1998, em Gotemburgo na Suécia, uma casa noturna incendiou resultando na morte de 63 jovens. No ano seguinte, outro incêndio em um *shopping center* em Incheon, Coréia do Sul, feriu 71 e matou outros 54. Em 2004, um supermercado em Assunção, Paraguai, queimou, matando 283.

Embora eventos isolados, estas situações revelam uma característica em comum: a imprevisibilidade. É impossível antever um incidente destas proporções, principalmente em ataques planejados, como nas ações terroristas presenciadas nos ataques às torres do World Trade Center, em 2001, e ao metrô de Londres, em 2005. Além de trabalho de prevenção na área de segurança, pouco pode ser feito para garantir que um evento destas proporções não se suceda. Este fato faz com que, a modelagem e simulação de multidões em computador seja um importante tema de pesquisa, especialmente visando prover conforto e segurança às multidões reais, fornecendo ferramentas valiosas para planejar áreas para pedestres, estações de transporte públicos, grandes edificações e centros comerciais.

Entretanto, não apenas estas aplicações são dadas a este tipo de simulação. As indústrias cinematográfica e de jogos, por sua vez, também se servem desta modelagem. No primeiro caso, a modelagem de multidões vêm sendo utilizada para auxiliar na direção de multidões e na criação de seqüências de agentes autônomos figurantes, como em seguimentos de batalhas da trilogia de "O Senhor dos Anéis" (*The Lord of the Rings*). A indústria de jogos incorpora estes modelos em seus produtos, tornando-os ainda mais realistas [Treuille et al., 2006].

Por tal diversidade de aplicações, diferentes metodologias são aplicadas para descrever esses modelos. No filme citado, a técnica empregada é baseada em inteligência artificial, porém a literatura apresenta outros modelos para descrever o comportamento de multidões, tais como os modelos baseados em Física ou ainda empregando técnicas mistas.

Atualmente os modelos físicos e matemáticos que expressam o comportamento

de multidões são o estado-da-arte quando se trata de simulações de eventos de pânico, e permitem simular o movimento de um grupo de indivíduos durante a evacuação de grandes ambientes.

1.1 Definição do Problema

Os atuais centros urbanos concentram uma profusão de indivíduos localizados em estruturas e edificações complexas, como ginásios esportivos, centros comerciais, aeroportos, etc. Estes ambientes são claramente edificações de múltiplos andares e não podem ser representados por uma solução bidimensional em simulações baseadas em Física. Este grau de liberdade extra àqueles suportados pelos modelos atuais de simulação, encontrados na literatura, é a limitação destes em simular agentes em terrenos irregulares ou na presença de rampas e escadas.

Além disto, simulações físicas são custosas, e tais modelos avaliam a interação entre cada indivíduo para atualizar suas posições espaciais em cada instante de tempo. Em ambientes complexos, como os anteriormente citados, o número de indivíduos aumenta, tornando o número de interações ainda maior. Nestes casos, torna-se necessário o uso de técnicas que desvinculem cálculos e interações que não são necessárias.

1.2 Objetivos

Este trabalho tem por objetivo modelar, matematicamente, um ambiente tridimensional para simulação de humanos virtuais em situações de emergência. Além disto, intenciona-se implementar um protótipo capaz de executar simulações compatíveis com o modelo e avaliar uma arquitetura de simulação concorrente para o modelo bidimensional, como um estudo de caso para, possivelmente, uma implementação do modelo tridimensional nesta arquitetura.

1.2.1 Objetivos específicos

- Modelar fisicamente um novo conjunto de forças que permita descrever simulação de multidão em ambientes tridimensionais;
- Implementar um protótipo capaz de simular situações de pânico em ambientes tridimensionais;
- Obter, a partir do protótipo, um conjunto de resultados passíveis de avaliação numérica; e
- Prover informações para visualização de humanos virtuais articulados e qualificação de resultados.

1.3 Organização da Dissertação

Esta trabalho de dissertação está organizado como segue. No Capítulo 2 são apresentados alguns conceitos, como agentes autônomos, grupos de agentes e comportamento. É apresentada uma revisão bibliográfica sobre o tema e discutido, em especial, tópicos sobre simulação de multidões regradas por forças sociais. Este capítulo termina retomando dois trabalhos chave para o desenvolvimento desta dissertação: o primeiro por servir de modelo-base as demais modelagens baseadas em Física, e o segundo por pertencer ao mesmo projeto que este se enquadra. Seguindo, no Capítulo 3, são apresentados alguns conceitos relacionados a programação em máquinas multiprocessadas, fundamentais para a descrição concorrente de um simulador de forças sociais em solução bidimensional. É apresentada a arquitetura paralela destas máquinas e o nível de concorrência que deve ser explorado nesta arquitetura. O capítulo apresenta, ainda, ferramentas que possibilitam obter um resultado desejado nesta configuração de computadores, e finaliza discutindo implementações de simulações em arquiteturas paralelas.

O Capítulo 4 apresenta o modelo matemático proposto no escopo do Centro de Simulação de Humanos Virtuais (CSHuV), projeto desenvolvido em cooperação com

a HP Brasil. O objetivo é prover o movimento de multidões em situações de pânico em ambientes tridimensionais. O capítulo descreve, ainda, a necessidade deste tipo de simulação junto a proposta de solução para o modelo baseado em Física proposto por Helbing, e discutido no Capítulo 2.

O protótipo implementado para avaliar o modelo matemático proposto é descrito no Capítulo 5. O capítulo apresenta, também, a ferramenta utilizada para esta etapa.

Resultados obtidos durante este trabalho de dissertação são discutidos no Capítulo 6. Estes resultados englobam os esforços realizados no modelo tridimensional para simulações tridimensionais e em arquiteturas concorrentes para o modelo bidimensional.

Por fim, o Capítulo 7 apresenta as considerações finais a respeito desta dissertação de mestrado.

Capítulo 2

Modelos de Simulação de Humanos Virtuais

Este capítulo tem por propósito apresentar uma visão geral da literatura relacionada à modelagem e simulação de multidões de humanos virtuais. O enfoque apresentado busca ressaltar aspectos comportamentais e físicos deste tipo de simulação. O estado-da-arte na área engloba o interesse das últimas três décadas em busca de um modelo para o comportamento de agentes e as razões motivadoras para tal. São abordados tópicos relevantes para simulações de humanos virtuais: agentes, grupos de agentes, concepção de agentes virtuais, concepção de forças sociais e modelos de forças sociais.

O capítulo inicia apresentando conceitos de agentes autônomos e comportamento, na seqüência expõe trabalhos relacionados a grupos de agentes autônomos e discute a concepção de um sistema de forças sociais baseadas em Física. Por fim, dois modelos de força sociais são apresentados.

2.1 Agentes

O uso do termo *agente* na literatura é comum. Neste trabalho, o termo é utilizado como a representação de um pedestre, pessoa ou qualquer outra entidade dotada de comportamento sendo capaz de atuar, agir e operar em um ambiente

virtual ou simulado. Embora o uso apresentado para o termo seja bastante específico, ele se encontra de acordo com a definição apresentada por Maes [Maes, 1995]: "agentes são sistemas computacionais que habitam ambientes complexos e dinâmicos, que percebem e atuam de forma independente, neste ambiente, na tentativa de realizar uma ação ou tarefa para a qual foram designados."

Uma definição para comportamento de agentes foi apresentada por Renault [Renault et al., 1990], que definiu o termo como o "modo que animais e humanos agem". Em outras palavras, nos permite interpretar *comportamento* como qualquer ação de um organismo vivo ou de um indivíduo que possa ser observada. Comportamento compreende instintos e hábitos, sendo governado, em organismos básicos, por simples natureza biológica e, em indivíduos mais complexos por características sócio-culturais.

A simulação de humanos virtuais busca, portanto, reproduzir o *comportamento* de agentes, atuando individualmente ou globalmente, bem como suas relações com outros e com o meio em que estão inseridos.

2.1.1 Agentes em grupos

A Ciência da Computação é capaz de oferecer um ambiente propício ao estudo de fenômenos biológicos e sociais a partir do uso de suas ferramentas. Na década de 80, Christopher Langton cunhou o termo *Vida Artificial* como sendo a reprodução de fenômenos biológicos a partir do uso destas ferramentas modernas [Langton, 1986, Langton et al., 1989]. Pelo uso da Vida Artificial é possível analisar ambientes reais através de modelos criados pelo homem para reproduzir comportamentos característicos a sistemas naturais e complementar as ciências biológicas ao sintetizar comportamentos semelhantes de organismos vivos em outros meios. De acordo com Langton, a Vida Artificial permite mudar a visão da "vida-como-conhecemos" para uma visão da "vida-como-poderia-ser". Para Langton a Vida Artificial deveria enxergar a vida como uma propriedade da organização da matéria, ao invés de uma propriedade da matéria organizada.

Um organismo é a base de uma população, uma máquina simples, governada por regras, construindo grandes agregados a partir da interação com os demais. Desta forma a Vida Artificial concentra-se em garantir o comportamento semelhante ao de sistemas naturais a partir do arranjo de máquinas simples, tornando o comportamento global essencialmente o mesmo exibido em sistemas vivos naturais. Vários autores trabalharam no sentido de reproduzir o comportamento coletivo de agentes autônomos e esta seção descreve alguns dos trabalhos mais relevantes no que se refere a evolução das pesquisas em simulação de multidões.

Esta característica de "vida-como-poderia-ser" pode também ser vista em outros trabalhos, como [Reynolds, 1987], um dos primeiros considerando grupos de agentes. Neste trabalho, Reynolds investigou o comportamento de grupos, tornando-se um dos pioneiros desta área denominada "animação comportamental", através de um novo paradigma no qual personagens autônomos determinam suas próprias ações (*self-animated characters*). Reynolds constatou que o movimento de bandos de pássaros, rebanhos de animais e cardumes de peixes apresentam um comportamento complexo e raramente visto na animação comportamental. Seu trabalho explorou métodos de simulação, como forma de evitar os tradicionais processos de criação utilizados na época, os quais exploravam a descrição da trajetória de cada agente, individualmente, ou mesmo processos semi-automatizados, como a técnica de animação por *keyframe*, na qual o animador define cenas-*chave* enquanto as demais eram obtidas através de interpolação.

Em seu trabalho, cada pássaro em um bando – *boids* e *flocks*, respectivamente – é capaz de decidir sua própria trajetória sem a intervenção do animador, uma grande vantagem sobre os métodos anteriores. Assim, descreveu um sistema comportamental distribuído, em que cada agente busca manter sua própria posição e orientação no bando através do balanceamento de três regras simples:

- **Concentração do bando**: Cada agente tenta manter-se rodeado de vizinhos por todos os lados, buscando, sempre, se estabilizar na posição média do bando.
- **Evitar colisões**: Cada agente deve manter uma distância segura dos demais,

criando um raio efetivo entre si e agentes vizinhos. Quando um agente estiver no interior deste limite de segurança de outro, há perigo de colisão entre eles. Nesta situação, os agentes devem aumentar/diminuir a velocidade, evitando a aproximação.

- **Igualar velocidades**: Cada agente tenta ajustar seu vetor posição com os dos seus vizinhos, tentando manter sua trajetória sempre paralela aos demais do grupo.

O trabalho de Reynolds mostra que animações realistas de grupos podem ser criadas a partir da aplicação de regras simples, e locais, dentro da própria estrutura do bando. O comportamento final é resultado de interações únicas e exclusivas ao domínio do bando. Embora não se trate de um sistema físico real, isto é, as equações do sistema foram modeladas a partir das regras impostas e não através de medidas e derivações de equações da dinâmica, Reynolds descreveu um sistema de partículas em um nível global igualmente rico ao comportamento de bandos de pássaros reais.

Seu ambiente simulado apresenta complexidade $\mathcal{O}(n^2)$, pois apresenta $n(n-1)/2$ interações, onde n é número de agentes no sistema. Cada agente i interage com os $n-1$ demais agentes, seguindo o esquema do algoritmo abaixo:

```
inicializarPosicoes();

enquanto(verdade) {
        desenharAgentes();
        moverTodosAgentesParaNovasPosicoes();
}
```

Na chamada `inicializarPosicoes()` os n agentes são instanciados e recebem suas posições iniciais dentro dos limites do mundo virtual. O mesmo é feito com seus vetores velocidade. O algoritmo evolui em um laço, repetindo duas etapas, atualizar a posição dos agentes na tela, caso exista uma saída gráfica, e calcular as novas posições dos mesmos no ambiente 3D.

O processo de cálculo entre os agentes é a implementação das três regras definidas anteriormente, e chamadas em

`moverTodosAgentesParaNovasPosicoes()`. Cada regra é independente, e seu resultado consiste em calcular os vetores de velocidade que formarão o vetor velocidade resultante do agente.

```
moverTodosAgentesParaNovasPosicoes() {
        vetor_t v[3];
        agente_t b;

        para cada agente b {
                v[0] = regra_1(b);
                v[1] = regra_2(b);
                v[2] = regra_3(b);
        }

        b.velocidade = b.posicao + v[0] + v[1] + v[2];
        b.posicao += b.velocidade;
}
```

Cada agente busca voar em direção ao centro de massa do conjunto de pássaros. Para fins práticos, permite-se negligenciar a massa individual considerando-as idênticas. O centro do grupo é percebido por ele ao avistar os demais agentes do conjunto, desta forma a posição do centro de massa dos $n-1$ pássaros é dada pela Equação 2.1.

$$\vec{c_j} = \frac{1}{n-1} \sum_{j=1, i \neq j}^{n} \vec{x_j} \tag{2.1}$$

A segunda regra consiste em manter os agentes próximos, porém não abaixo de um limite definido. Na prática esta regra é aplicada para evitar a colisão entre um agente com os demais.

A terceira regra tenta estabilizar a velocidade de um agente com a velocidade de cruzeiro dos outros agentes. Desta forma, cada um tenta ajustar seu vetor velocidade com o do seu vizinho, tentando manter sua trajetória sempre paralela aos demais do grupo.

$$\vec{v_j} = \frac{1}{n-1} \sum_{j=1, i \neq j}^{n} \vec{v_j} \tag{2.2}$$

Este trabalho serviu como inspiração para outros modelos simulados, substituindo antigas técnicas de animação comportamental por modelos

matemáticos, não necessariamente reais, mas que evoluem apresentando características observadas na natureza. Exemplos do uso desta técnica são encontrados na indústria cinematográfica, em filmes como *Lion King*, *Batman Returns* e *Jurassic Park*, entre outros.

Também baseados em Física, Tu e Terzopoulos [Tu and Terzopoulos, 1994] propuseram um modelo comportamental para descrição de Vida Artificial de cardumes, em que a interação entre cada peixe é modelada através de um sistema físico massa-mola. Seu trabalho apresenta, agregado aos agentes, visão sintética e percepção do meio, que influencia suas intenções, porém suas reações não são previsíveis. Neste trabalho são considerados aspectos hidrodinâmicos do movimento dos peixes e comportamentos biológicos, tais como acasalamento, caça, fuga de predadores e alimentação (Figura 2.1). Estes aspectos baseiam-se nas características biológicas dos cardumes, como o agrupamento de peixes menores em grupos, para aumentar a chance de sobrevivência diante de predadores.

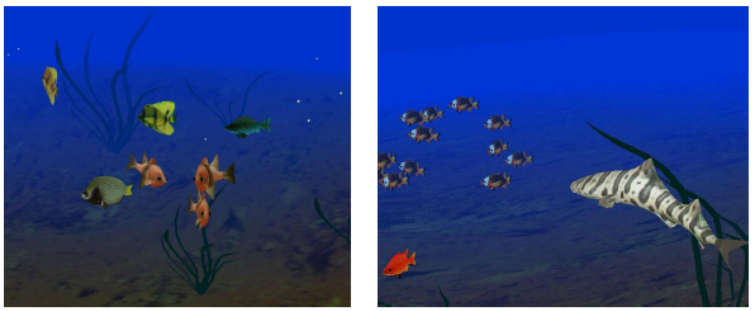

FIGURA 2.1 – Representação do trabalho realizado por Tu e Terzopoulos. A figura da apresenta um cardume com pequenos peixes. A segunda mostra a formação de um grupo diante de um predador [Tu and Terzopoulos, 1994].

Ao contrário dos anteriores, que aplicam conceitos de Física, Mataric [Mataric, 1994] serviu-se de 20 robôs móveis e descreveu o comportamento emergente a partir de um modelo de aprendizado de regras sociais em sistemas multi-agentes. Realizou um estudo sobre como aumentar os benefícios individuais médios de um grupo maximizando os benefícios coletivos. Para modelar o comportamento de grupos, foram selecionados alguns comportamentos básicos, tais como evitar colisão, agrupar, dispersar, seguir um líder e buscar por algo (*foraging*). Obteve, desta

forma, uma série de comportamentos de alto-nível, por exemplo, formação de grupos (*flocking*).

Também refletindo comportamentos sociais, foi realizado um trabalho a partir da constatação de que uma formiga é capaz de encontrar o menor caminho entre a fonte de alimentação e o formigueiro, deixando rastros de feromônios ao longo da sua trajetória. Dorigo e Gambardela [Dorigo and Gambardella, 1997] propuseram um modelo para simulação de colônias de formigas, denominado *Ant Colony System* no qual a quantidade de feromônios deixado por uma formiga permite a outra decidir, ou não, se deve prosseguir por um determinado caminho. A Figura 2.2 ilustra a técnica utilizada. Quando uma formiga encontra um ponto de decisão, toma um direção aleatória. Assumindo, por princípio, que a velocidade de uma formiga é constante, a formiga que encontrar o próximo ponto de decisão em um tempo menor atravessou o caminho mais curto. Isto faz com que o caminho fique marcado por feromônios e leve outras formigas a escolher seus próprios caminhos. O resultado é um comportamento emergente autocatalítico[1], com formação de filas e minimização de distâncias, sem necessidade de comunicação entre agentes. Os autores apresentam o sistema como uma ferramenta de propósito geral, o qual pode ser utilizado para resolver problemas de minimização combinatorial, como o problema do caixeiro viajante.

Outro modelo baseado em Física foi proposto por Bouvier *et al.*, em 1997, que apresentou uma analogia ao eletromagnetismo em [Bouvier et al., 1997], introduzindo o conceito de "cargas-de-decisão" e "campos-de-decisão". Utilizou um sistema de partículas adaptadas para estudar multidões humanas, através de um grupo de partículas interagentes. O movimento, objetivo e decisão do agente é baseado apenas em forças newtonianas. As cargas de decisão são influenciadas pelos campos de decisão da mesma forma que uma partícula carregada é influenciada por um campo elétrico. Em seu trabalho foi modelada uma força de atrito para evitar o aumento excessivo de velocidade, permitindo aos agentes uma velocidade limite.

[1]Dorigo [Dorigo and Gambardella, 1997] descreve comportamento emergente autocatalítico como um comportamento de retorno positivo, no qual o processo é máquina que reforça a si próprio, apresentando uma convergência rápida.

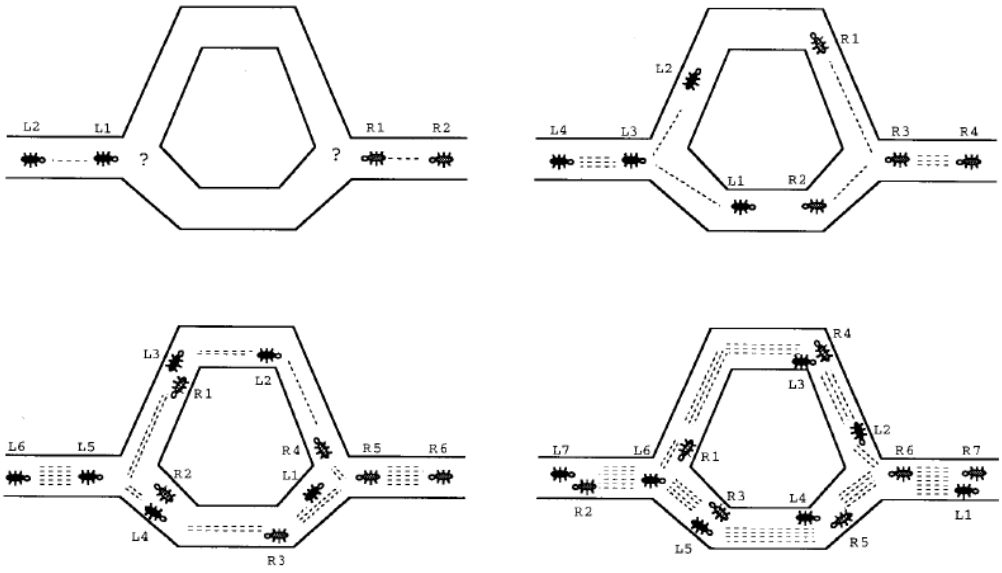

FIGURA 2.2 – Modelo de Dorigo, que explora o fenômeno da informação sem a comunicação entre agentes. Um grupo de formigas ao encontrar uma região ambígua toma uma decisão aleatória. Suas decisões rapidamente levam o sistema a solução ideal, onde a quantidade de feromônios é maior. Na figura, as linhas tracejadas representam a quantidade de feromônio [Dorigo and Gambardella, 1997].

Os agentes evitam colisão e podem deslocar-se em grupos.

Brogan e Hodgins utilizaram dinâmica para descrever o comportamento de grupos [Brogan and Hodgins, 1995]. Em seu trabalho consideraram um algoritmo para evitar colisões, que determina a velocidade desejada de cada agente dada a localização e velocidade dos demais e obstáculos visíveis. Foi desenvolvido um modelo de percepção para determinar os obstáculos de cada agente em uma etapa anterior àquela de deslocamento. Como resultado, foram capazes de reproduzir o movimento de robôs e ciclistas.

Musse apresentou um modelo para simular o comportamento de uma população genérica, em ambientes complexos em [Musse and Thalmann, 1997]. Neste modelo foram utilizados conceitos da Sociologia para expressar o comportamento dos agentes. Em 2001, Musse e Thalmann [Musse and Thalmann, 2001] propuseram um modelo hierárquico para descrever multidões com diferentes níveis de autonomia: guiados, programados e autônomos. Como resultado, desenvolveu o *ViCrowd*, uma aplicação para modelar e gerar

multidões virtuais baseados em grupos de indivíduos.

Goldensteing e colaboradores [Goldenstein et al., 1999] desenvolveram um modelo dinâmico não linear para simular grandes quantidades de agentes em um ambiente que pode ser adaptado em tempo real. Esse modelo apresenta comportamentos de baixo nível dos agentes, como evitar colisão e encontrar um caminho a fim de alcançar um objetivo. Os agentes são dotados de orientação e interagem com o ambiente através do conhecimento da posição e da velocidade dos objetos.

Buscando investigar o comportamento de multidões em situações de pânico, Helbing [Helbing et al., 2000] modelou a interação entre agentes distintos através de um modelo matemático generalizado de forças sócio-psicológicas e físicas. Como resultado, seu modelo era capaz de descrever o comportamento de um grande número de agentes em tais situações críticas, como evacuações de locais públicos durante um incêndio. Seu trabalho será apresentado na Seção 2.2.1 com maiores detalhes.

Ulicny e Thalmann [Ulicny and Thalmann, 2001] propuseram a combinação de uma máquina de estados finitos e regras para o controle de agentes em uma abordagem multi-camadas. Em baixo nível, todo comportamento complexo é implementado por máquinas hierárquicas de estados finitos, e são associados aos estados psicológicos (medo, capacidade de locomoção, etc). Em alto nível, as regras selecionam comportamentos complexos baseados nos estados dos agentes e do ambiente, como habilidade de desviar, fugir, etc.

Evers e Musse [Evers and Musse, 2002] desenvolveram um simulador baseado em memórias virtuais para gerenciar o comportamento de indivíduos. Desta forma, o estado emocional individual em experiências passadas influencia nas decisões frente a um novo evento.

Braun [Braun et al., 2003, Braun et al., 2005] expressou a individualidade dos agentes estendendo o modelo físico de Helbing, dotando cada um de interesses e objetivos próprios. Esta extensão baseia-se na inovação conceitual sobre os agentes, munindo-os de *percepção–decisão–ação*. Seu trabalho será retomado na Seção 2.2.2.

Mais recentemente, Treuille [Treuille et al., 2006] em 2006, apresentou um modelo baseado em dinâmica contínua, adaptado de [Hughes, 2003], através de um campo de potencial dinâmico [Arkin, 1987] herdado da robótica. Seu modelo integra simultaneamente navegação global com a capacidade de desviar de obstáculos e outros agentes, sem a necessidade de descrever explicitamente o controle de colisão. Seu trabalho baseia-se em grandes multidões com objetivo comum, não sendo aplicável em casos onde agentes possuem intenções distintas, pois não utiliza dinâmica baseada em agentes. É adotado uma perspectiva contínua do sistema, do movimento visto como a minimização da energia por partícula. O resultado de seu trabalho pode ser visto na Figura 2.3.

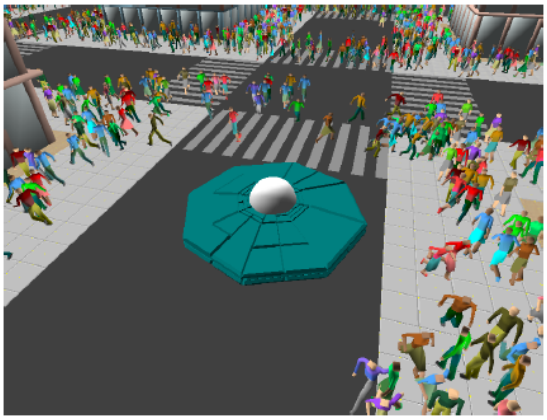

FIGURA 2.3 – Resultado apresentados por Treuille de uma situação de pânico: Interação de uma multidão com um objeto voador não identificado [Treuille et al., 2006].

Nesta seção foram apresentados trabalhos que se classificam em dois grupos no escopo de modelagem de humanos virtuais: *i*) sistemas baseado em regras e *ii*) modelos baseados em Física. De acordo com essa classificação, o modelo proposto neste trabalho enquadra-se no segundo grupo.

2.2 Forças Sociais

Embora o comportamento humano seja de aparência caótica ou irregular o suficiente para que se possa predicar, modelos estocásticos baseados em Física foram desenvolvidos restringindo a descrição dos comportamentos por probabilidades

encontradas em grande populações de indivíduos. Esta idéia foi seguida pelo modelo de pedestres baseado em cinética dos gases, proposto em [Helbing, 1992], que teve inspiração em analogias apresentadas por Henderson em meados da década de 70 da teoria de fluidos. Entretanto, outra abordagem para as mudanças de comportamento de pedestres foram sugeridas por Lewin [Lewin, 1951] que, em seu livro de 1951, atribuiu as mudanças de comportamento a *campos-sociais* ou *forças-sociais*.

Analogamente ao trabalho de Reynolds [Reynolds, 1987], já descrito neste capítulo, forças sociais não representam forças físicas reais, mas descrevem, matematicamente, comportamentos sociais de forma igualmente rica ao comportamento real de uma multidão de pedestres (*comportamento emergente*).

Helbing descreveu, matematicamente, aspectos observados no comportamento social de multidões. Embora este modelo matemático evolua cada agente individualmente, características do grupo e do meio são preservadas. Observa-se isto quando extrapola a premissa de que, sem influência de forças externas – paredes, outros agentes, obstáculos, etc. –, indivíduos buscam sempre caminhos que evitem contornos, optando sempre pelo menor caminho. Ao introduzir outros agentes (e obstáculos) seu modelo traduz o respeito pela região pessoal de um pedestre através de uma força de caracter repulsivo, força esta definida como *força social*.

Atualmente os modelos físicos baseados em agentes são o estado-da-arte para a computação de dinâmica de pedestres. A modelagem dos movimentos individuais dos pedestres resulta da descrição macroscópica do fluxo dos pedestres, e permite, por exemplo, encontrar rotas para evacuação em situações de pânico, desenvolver e construir *facilities*.

A seguir, são discutidos dois modelos baseados em interação de agentes a partir de forças modeladas de acordo com leis físicas para representar o comportamento de agentes em situações de emergência. O primeiro analisado trata-se do modelo de Helbing, que vêm servindo de base para grande parte das simulações baseadas em Física. O segundo, de Braun, é uma especialização do modelo de Helbing que permitiu dotar os agentes de características individuais.

2.2.1 Modelo de Helbing

Helbing e colaboradores [Helbing et al., 2000] propuseram um modelo que associa forças físicas e sócio-psicológicas com o objetivo de descrever o comportamento de multidões humanas em situações de pânico. Esse modelo utiliza um sistema de n partículas físicas, onde cada partícula i de massa m_i, tem um valor de velocidade desejada v_0 em uma direção indicada por um vetor unitário $\hat{e}_i^0$. Esta partícula tende a adaptar sua velocidade instantânea $\vec{v}_i$ a estas condições desejadas dentro de um certo intervalo de tempo τ_i. Simultâneamente, as partículas tendem a manter uma distância dependente da velocidade em relação às outras partículas j e paredes p, usando forças de interação (de caracter repulsivo). A mudança da velocidade no tempo t é dada pela equação diferencial proposta por Helbing $\vec{F}_i^H$:

$$m_i \frac{d}{dt}\vec{v}_i = \vec{F}_i^H = m_i \frac{v_i^0 \hat{e}_i^0 - \vec{v}_i(t)}{\tau_i} + \sum_{\substack{i=1 \\ i \neq j}}^{n} \vec{f}_{ij} + \sum_{w} \vec{f}_{iw} \tag{2.3}$$

onde τ_i apresenta-se no primeiro termo à direita da equação, $\vec{f}_{ij}$ e $\vec{f}_{iw}$ no segundo e terceiro termos, respectivamente.

A tendência psicológica de dois pedestres i e j (os quais são geometricamente representados por discos de raios R_i e R_j, respectivamente) a permanecerem afastados um do outro é descrita pela expressão:

$$\vec{f}_{ij}^{1} = A_i e^{(R_{ij} - d_{ij})/B_i} \hat{n}_{ij} \tag{2.4}$$

onde R_i e R_j representam os raios dos agentes i e j respectivamente. A_i e B_i são constantes, $d_{ij} = |\vec{r}_i - \vec{r}_j|$ denota a distância entre o centro de massa dos pedestres e $\hat{n}_{ij} = (\vec{r}_i - \vec{r}_j)/d_{ij}$ é o vetor normalizado apontando do pedestre j para o pedestre i. $R_{ij} = R_i + R_j$ representa a soma dos raios de i e j. As constantes A_i e B_i utilizadas por Helbing são apresentadas na Tabela 2.1.

São assumidas duas forças adicionais para impedir a interpenetração corporal dos agentes, quando estes entram em contato. Uma força radial:

Parâmetro	Valor
m	$80kg$
v_i^0	$(0,8; 1,0; 1,5)m/s$
τ_i	$0,5s$
A_i	$2000N$
B_i	$0,08m$
k	$1,2 \times 10^5 kg/s^2$
κ	$2,4 \times 10^5 kg/ms$

TABELA 2.1 – Parâmetros utilizados por Helbing em seu modelo.

$$\vec{f}_{ij}^{2} = k(R_{ij} - d_{ij})\hat{n}_{ij} \tag{2.5}$$

e uma força de atrito dada por:

$$\vec{f}_{ij}^{3} = \kappa G(R_{ij} - d_{ij})\Delta v_{ji}^{t} t_{ij} \tag{2.6}$$

que impede movimento tangencial entre os agentes que estão em contato. Na Equação 2.6, $t_{ij} = (-n_{ij2}, n_{ij1})$ é a direção tangencial (onde n_{ijk} é a k-ésima componente do vetor $\hat{n}_{ij}$ e $\Delta v_{ji}^{t} = (\vec{v}_j - \vec{v}_i) \cdot \hat{t}_{ij}$ é a diferença de velocidade tangencial, enquanto k e κ representam constantes. A função $G(x)$ representa uma função de *Heaviside* que vale 0 se os agentes não se tocam e x caso contrário. Em suma:

$$\vec{f}_{ij} = \underbrace{A_i e^{(R_{ij} - d_{ij})/B_i}\hat{n}_{ij}}_{\vec{f}_{ij}^{1}} + \underbrace{k(R_{ij} - d_{ij})\hat{n}_{ij}}_{\vec{f}_{ij}^{2}} + \underbrace{\kappa G(R_{ij} - d_{ij})\Delta v_{ji}^{t}\hat{t}_{ij}}_{\vec{f}_{ij}^{3}} \tag{2.7}$$

O tratamento dado à interação dos agentes com as paredes é análogo àquele entre agentes, e dado pela expressão:

$$\vec{f}_{ip} = A_i e^{(R_{ip} - d_{ip})/B_i}\hat{n}_{ip} + k(R_{ip} - d_{ip})\hat{n}_{ip} + \kappa G(R_{ip} - d_{ip})(\vec{v}_i \cdot \hat{t}_{ip}) \cdot \hat{t}_{ip} \tag{2.8}$$

Os parâmetros utilizados pelos autores para a aquisição de resultados estão na Tabela 2.1 e os resultados observados por este modelo corroboram com alguns dados observados em cenários reais de evacuações. É apontada a formação de *arcos*

em saídas estreitas conforme a Figura 2.4. O aumento do módulo da velocidade desejada ocasiona um aumento significativo no tempo de evacuação. Além disso, a transição para um movimento desordenado devido ao engarrafamento em uma saída causado pelo aumento da velocidade desejada.

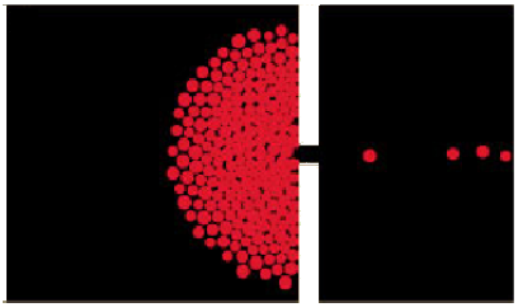

FIGURA 2.4 – Resultado apresentados por Helbing demonstrando o efeito arco descrito anteriormente. [Helbing et al., 2000]

Além destes, outros aspectos foram observados por Helbing e colaboradores. A comunicação entre agentes foi simulada para avaliar os efeitos entre agentes quando estes deveriam encontrar a saída em uma sala sem visibilidade, devido a presença de fumaça. Com isso, concluíram que as melhores condições para sobrevivência é um equilíbrio entre comportamentos individualistas, onde o agente decide, com comportamentos informados, resultado da comunicação. Como resultado, mais agentes encontraram as saídas e mais rotas foram utilizadas.

Em outro ambiente simulado foi considerado um corredor com largura variável, em que desejava-se observar os efeitos do ambiente sobre os agentes. Observou-se o efeito de espalhamento da multidão na região alargada, o que resultou em um engarrafamento na região final conforme a Figura 2.5.

2.2.2 Modelo de Braun

Braun propôs um modelo para o estudo do impacto de características individuais dos agentes em simulações de multidões durante situações de emergência. Em seu trabalho, utilizou-se do modelo de Helbing apresentado na Seção 2.2.1, o qual foi expandido a fim de incorporar diferentes individualidades para o comportamento de agentes e grupos.

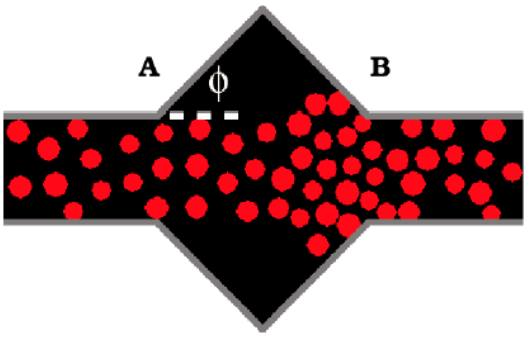

FIGURA 2.5 – Resultado publicado por Helbing *et al.*, demonstrando o engarrafamento na segunda metade do ambiente físico (B), devido ao alargamento no corredor em A, que gerou uma dispersão dos agentes [Helbing et al., 2000].

A motivação de seu trabalho foi permitir reações diferentes a agentes dependendo de suas características individuais e do grupo ao qual pertencem. Determinados agentes podem sofrer um dano menor em uma situação de pânico que outras, ou ainda dependendo da estrutura dos grupos existentes, a ação individual pode ser alterada em função do grupo ao qual o agente pertence, pois ele pode retornar ao local de perigo para resgatar um membro de seu grupo.

Braun acrescentou aos agentes do modelo de Helbing (Seção 2.2.1) um conjunto de parâmetros, permitindo a criação de uma população heterogênea com diferentes características. À cada agente i foram definidos os seguintes parâmetros:

- Id_i: Identificador do agente i;
- $Familia_i$: Identificador da família. Em seu modelo, uma família é um grupo pré-definido que representa um vínculo entre os agentes;
- DE_i: Nível de dependência do agente i. Esta dependência tem seu valor normalizado em um intervalo fechado de [0,1] e representa a necessidade de ajuda para locomoção física;
- AL_i: Contrário ao nível de dependência, o nível de altruísmo representa agentes dispostos a ajudar agentes com necessidade de locomoção. Seu valor é, também, normalizado em um intervalo fechado [0,1]; e,
- v_i^0: Módulo da velocidade desejada do agente.

Foi redefinido o valor da velocidade desejada (v_i^0), passando a ser governada pela expressão:

$$v_i^0 = (1 - DE_i)\, v^m \tag{2.9}$$

onde v^m é a velocidade máxima estipulada para um agente do sistema e DE_i o nível de dependência do agente i. Conseqüentemente, se um agente for totalmente dependente ($DE = 1$), sua velocidade desejada será nula, o que pode representar agentes com insuficiência de mobilidade. Ao considerar $AL = 0$ e $DE = 0$, Braun recupera o modelo proposto por Helbing.

Braun propôs uma formação de grupos relacionada com a força de altruísmo, $\vec{Fa}_{ij}$, implementada como uma interação entre dois ou mais agentes de uma mesma família. A Equação 2.10 representa a força modelada,

$$\vec{Fa}_{ij} = KAL_iDE_j|\vec{d}_{ij} - \vec{d}_{id}|\hat{e}_{ij} \tag{2.10}$$

onde $\vec{d}_{ij}$ representa a distância do agente j em relação ao agente i, $\vec{d}_{id}$ é a distância do agente i a saída do ambiente simulado, K é uma constante de simulação e $\hat{e}_{ij}$ é o vetor unitário do agente i ao agente j.

Estes novos parâmetros introduzidos no modelo, fornecem informações importantes do sistema: quanto maior for o parâmetro AL_i e DE_j, dos agentes i e j, maior será o valor da força de altruísmo $\vec{Fa}_{ij}$, a qual irá apontar, na direção ao agente j. Em seu trabalho, o resgate de um agente dependente por um agente altruísta faz com que as velocidades desejadas dos dois se igualem, de maneira que eles possam mover-se juntos.

2.3 Conclusão

O modelo de simulação proposto neste trabalho visa estender o modelo de Helbing para um ambiente tridimensional, porém mantendo características de propostas anteriores baseadas neste mesmo modelo.

Os conceitos apresentados neste capítulo são retomados no Capítulo 4 onde é apresentado um modelo compatível com Helbing e Braun capaz de permitir a simulação de multidões em ambientes tridimensionais.

Se tratando de simulações interagentes, isto é, onde em cada instante um as ações de um agente atua sobre outro, simulações físicas tornam-se custosas, em termos computacionais. Para cada agente introduzido no sistema, um novo conjunto de forças deve ser calculado. Como no trabalho de Reynolds, o modelo de forças sociais de Helbing também apresenta $n(n-1)$ interações, onde n é o número de agentes no ambiente. Assim, com complexidade da simulação é $\mathcal{O}(n^2)$, o modelo de Helbing necessita de abordagens específicas para redução deste custo. O Capítulo 3 descreve um ambiente composto por computadores multiprocessados e ferramentas que podem auxiliar esta classe de simulação, enquanto o Capítulo 5 descreve, para implementação do protótipo tridimensional, uma metodologia de redução por contextos específicos.

Capítulo 3

Processamento em Aglomerado de Computadores

Este capítulo introduz conceitos de processamento em aglomerados de computadores relevantes ao desenvolvimento deste trabalho. O capítulo inicia apresentando aglomerados de computadores: a arquitetura paralela selecionada para desenvolvimento da implementação bidimensional de Helbing [Helbing et al., 2000]. Em particular são identificados os diferentes níveis de concorrência encontrados nesta arquitetura. Na seqüencia, Seção 3.2, são apresentadas exemplos de ferramentas de programação utilizadas para desenvolvimento de aplicações paralelas em aglomerados, ressaltando técnicas de execução e de escalonamento intra e entre-nós utilizadas para otimizar a utilização dos recursos do *hardware*. Na Seção 3.4 são apresentadas implementações de simulações distribuídas exemplificando o uso de aglomerados em trabalhos relacionados ao proposto. O capítulo finaliza avaliando a necessidade deste paradigma de computação em simulações no contexto deste trabalho.

3.1 Processamento Paralelo

Máquinas denominadas *computadores pessoais* (PCs, do inglês *Personal Computers*) apresentam desempenho desejado para a maioria das tarefas domésticas.

Entretanto, há classes de aplicações que requerem capacidades de processamento bem superiores, como os problemas encontrados nos domínios da Computação Científica, como Astronomia, Matemática Computacional, Meteorologia, Bioinformática e Física. Muitas aplicações desenvolvidas para resolver questões destas áreas podem ter seus tempos de execução estendidos por semanas, podendo mesmo ocorrer casos onde estas aplicações não finalizem sua execução por falta de recursos. Neste sentido, soluções de *hardware* propondo arquiteturas paralelas provêem uma nova perspectiva para execução de tais aplicações. O preço pago por este poder computacional é o alto custo de arquiteturas e a dificuldade em programá-las.

Esta seção apresenta a arquitetura de aglomerados de computadores de custo relativamente baixo, e suas características principais.

3.1.1 Arquitetura paralela

Nas últimas décadas a área de Processamento de Alto Desempenho (PAD) vem buscando alternativas a essas questões: custo do hardware paralelo e dificuldade de programação. No início da década de 90 foi proposta uma nova abordagem para arquiteturas paralelas, denominada *Computação Baseada em Aglomerados* [Becker et al., 2002]. Sempre próxima ao desenvolvimento tecnológico da indústria de componentes eletrônicos, mídias de armazenamento e principalmente de dispositivos de comunicação e processamento, a proposta é de construir uma arquitetura paralela com recursos computacionais encontrados facilmente no mercado. Um aglomerado, desta forma, consiste em um conjunto de computadores associados através de uma rede de comunicação para suportar processamento paralelo. Este tipo de arquitetura se permite sustentar com computadores disponíveis ao público geral: computadores pessoais encontrados no mercado à um custo relativamente baixo e com o sistema de livre utilização GNU/Linux. Desta forma, torna-se possível manter um sistema atualizado, com peças de reposição e atualização, evitando a defasagem.

A utilização de computadores pessoais de custo reduzido foi abordada,

pela primeira vez, por Thomas Sterling e Donald J. Becker [Becker et al., 2002] do *Goddard Space Flight Center* da NASA (*National Aeronautics and Space Administration*), e tornou-se sinônimo de computação paralela de baixo custo (em Cluster Beowulf). A equipe dos pesquisadores elaborou um aglomerado de 16 nós – em um aglomerado cada máquina recebe o nome de nó, ou nodo –, com processadores 486 e rede Ethernet 10MB/s capaz de atingir 46 MFLOPS[1] de desempenho máximo e custara uma quantia inferior a U$ 50.000,00 (dólares americanos). Três anos após, com o advento dos processadores Intel Pentium Pro, o aglomerado atingira a marca de 2 GFLOPS[2] com 32 nós.

Atualmente, com intuito de diminuir custos e acelerar trocas de dados, os fabricantes de *hardware* passaram a vender placas com suporte a múltiplos processadores, chegando ao mercado computadores denominados *multiprocessados*. Geralmente, processadores de uma máquina multiprocessada acessam uma região de memória compartilhada através de um barramento de alta velocidade, e a comunicação entre processos é feita através da região comum de memória com operações ordinárias `load` e `store`. Multiprocessadores recebem o adjetivo *simétrico* (SMP - *Symmetric Multiprocessor*) pois todos seus processadores apresentam igual acesso ao barramento e à memória, não apresentando privilégio por parte do sistema operacional a nenhum dos processadores no atendimento de requisições. Isto faz com que o desempenho da máquina caia, a medida que aumenta a disputa por seu acesso.

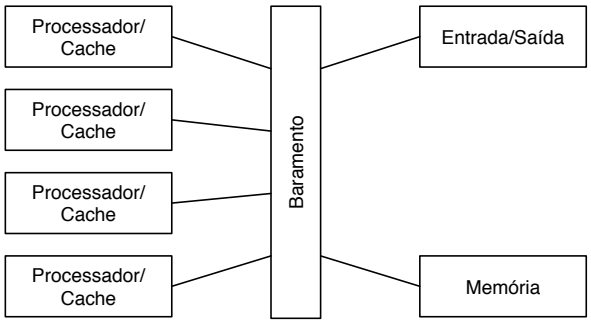

FIGURA 3.1 – Arquitetura de uma máquina multiprocessada simétrica. A comunicação entre os recursos do hardware é feito sobre um barramento comum.

[1] 1 MFLOP é o equivalente a 1 (um) milhão de operações de ponto flutuante por segundo.
[2] 1 GFLOP é o equivalente a 1 (um) bilhão de operações de ponto flutuante por segundo.

A Figura 3.1 representa a arquitetura típica de uma máquina SMP, e um fator que compromete a escalabilidade destas máquinas é o uso de um barramento como rede de interconexão. O barramento impede a construção de SMPs com uma grande quantidade de processadores por se tratar de um canal compartilhado que só permite uma transação por vez. No entanto, é possível destacar algumas vantagens destas máquinas SMPs:

- **Simetria**: qualquer processador pode acessar qualquer parte da memória ou qualquer dispositivo de entrada e saída
- **Espaço de endereçamento único**: somente uma cópia do sistema operacional reside na memória. O próprio sistema escolhe em qual processador um dado processo deve ser executado, dependendo da carga do sistema, obtendo de forma simples, um equilíbrio dinâmico de carga.
- **Baixa latência**: a troca de dados de um processo à outro é feita por acessos ordinários à uma porção de memória comum. Não é necessário copiar dados.
- **Replicação e coerência**: a localidade de dados é garantida pelas *caches* dos processadores, cuja coerência é garantida pelo *hardware*.

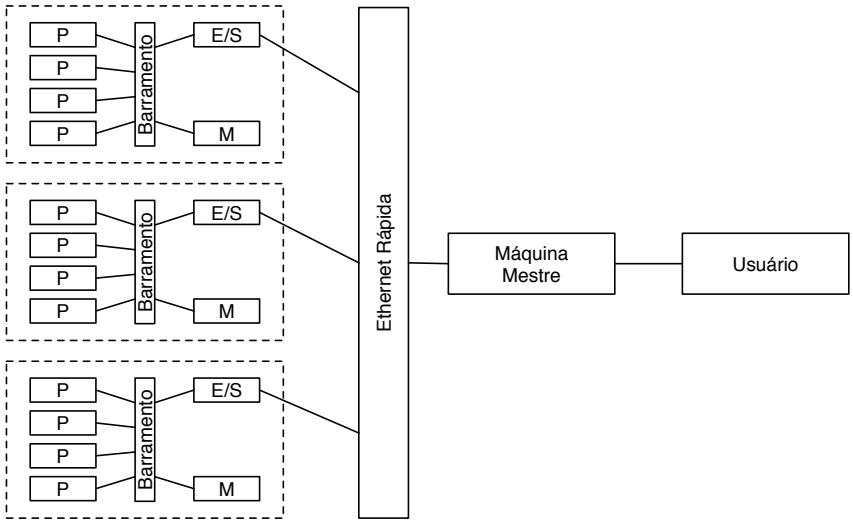

FIGURA 3.2 – Arquitetura de um aglomerado formado por máquinas SMP, unidas por uma rede Ethernet rápida e acessado através de uma máquina mestre.

Um aglomerado de computadores representa a forma mais popular de computador paralelo. Tecnicamente, um aglomerado não é um computador, e sim um conjunto de computadores interligados por uma rede de rápida velocidade, organizado para dar ao usuário a impressão de estar usando uma única máquina, da mesma forma que o Beowulf da NASA. Como os demais dispositivos das máquinas do aglomerado, as interfaces de rede são interfaces comuns encontradas no mercado, porém em versões mais rápidas, como Gigabit Ethernet ou Myrinet. Elas utilizam protocolos de comunicação rápidos, como mensagens ativas, permitindo deixar o sistema operacional de lado durante a comunicação evitando a sobrecarga e permitindo acesso direto, em nível de usuário, às interfaces de rede. A Figura 3.2 representa um aglomerado contendo nós SMP e interligados por uma rede Ethernet de alta velocidade. Nesta figura, o processador de cada máquina é representado por P, a memória por M e os dispositivos de entrada e saída por E/S. A arquitetura de um aglomerado supõe uma única conexão ao mundo externo, sendo feita a partir de uma *Máquina Mestre*, pela qual todo usuário da máquina paralela tem acesso aos nós.

As principais características de um aglomerado são:

- **Independência**: Cada elemento possui um ou mais processadores, memória e dispositivos de entrada e saída independentes. Executa o seu próprio sistema operacional, e pode ser desvinculado do aglomerado sem que isto afete o funcionamento dos outros. Da mesma forma, é possível agregar novas máquinas facilmente ao aglomerado, tornando o desempenho do sistema ainda mais elevado.

- **Imagem única do sistema**: Um aglomerado é um recurso computacional único, ao contrário de um sistema distribuído onde cada elemento é um recurso individual.

- **Conexão especializada**: Os elementos de um aglomerado são geralmente conectados por uma rede de alta velocidade.

3.1.2 Níveis de concorrência

Como mostram as figuras 3.1 e 3.2, dois níveis distintos de concorrência podem ser explorados em aglomerados de SMPs. Em uma visão local, a arquitetura SMP apresenta apenas uma região de endereçamento de memória, ao contrário do aglomerado, que se justifica por uma memória distribuída entre os nós. Esta seção se caracteriza por analisar estes diferentes níveis de concorrência.

Concorrência intra-nó: A concorrência é dita **intra-nó** quando as atividades concorrentes são executadas dentro da mesma máquina multiprocessada, seja ela parte de um aglomerado ou uma máquina independente. Este tipo de concorrência pode ser classificada de duas formas distintas, e dependem do número de fluxos de execução em um dado momento e o número de processadores disponíveis na arquitetura.

- Concorrência real; e,
- Concorrência temporal.

A primeira, *concorrência real*, ou simplesmente paralelismo, ocorre quando o número de processadores for igual, ou superior, ao número de fluxo de execuções. O paralelismo caracteriza execução simultânea de atividades de uma aplicação. No entanto, a existência de uma quantidade maior de fluxos de execuções que o número de processadores disponíveis gera compartilhamento de recursos da máquina, e a concorrência é dita *temporal*. Enquanto a concorrência real reflete o paralelismo encontrado no hardware, a concorrência temporal encontra-se associada às características da aplicação.

Tanto na concorrência real como na concorrência temporal, diferentes fluxos de execução compartilham informações através de operações ordinárias `load` e `store` em um espaço de endereçamento comum. Isto garante que um certo dado escrito em memória por um fluxo de execução possa ser lido por uma instrução de outro fluxo qualquer. Porém, ao contrário de execuções seqüenciais, devem ser tomados cuidados extras ao acesso da memória pelos diferentes fluxos, introduzindo

mecanismos que garantam a sincronização das operações de leitura e escrita de dados compartilhados. Dentre estes mecanismos de sincronização, os mais populares são aqueles que garantem exclusão mútua no acesso da memória, *mutexes* e os que realizam controle no avanço da execução, seja criando novos fluxos (`create`) ou aguardando o término de outro fluxo (`join`).

Existem diversas ferramentas que propõem o uso de mecanismos de execução baseados em múltiplos fluxos. Entre estas, ferramentas baseadas nos padrões OpenMP [OpenMP, 1997, Dagum and Menon, 1998] e POSIX para threads [American Nacional Standards Institute, 1994].

Concorrência entre-nós: Como observado anteriormente, a arquitetura de um aglomerado apresenta outro nível de concorrência a ser explorado: **entre-nós**. A Figura 3.2 permite observar que cada nó da arquitetura apresenta seus próprios recursos de processamento e armazenamento. Desta forma, não é possível realizar trocas de dados entre as tarefas utilizando um espaço de endereçamento comum, pois cada nó possui seu próprio espaço de endereçamento. Para que uma tarefa possa contribuir com outra, é necessário o uso de uma rede de interconexão entre os nós. O mecanismo típico de comunicação de dados faz uso de mecanismos de troca de mensagens, com operações do tipo `send` e `receive`.

É possível citar algumas ferramentas que permitam a troca de mensagens para o compartilhamento de dados, além do padrão *sockets*: PVM, do inglês *Parallel Virtual Machine* e MPI, *Message Passing Interface*. Esta última será analisada com maiores detalhes na próxima seção.

3.1.3 Ganho de desempenho

As seções anteriores discutem a necessidade de explorar a concorrência em dois níveis, para obter melhor desempenho na utilização dos recursos do aglomerado. Entretanto, a utilização simultânea destes dois mecanismos não é uma tarefa simples e o emprego dos dois níveis de concorrência se justifica pelo ganho que pode ser obtido pelo sobreposição parcial dos tempos de latência devido as comunicações

pela execução de cálculo efetivo [Valiant, 1990]. No entanto, um novo grau de desempenho ainda pode ser obtido através de técnicas de escalonamento aplicativo. Através de um desmembramento do problema em partes, o escalonamento aplicativo explora as características do problema reduzindo o número de comunicações para solução em um aglomerado.

3.2 Ferramentas de Programação

Como discutido na Seção 3.1.2, a programação de aglomerados de SMPs exige uma correta exploração dos dois níveis de concorrência apresentados pela arquitetura para obter bons índices de desempenho. Portanto, ferramentas que forneçam mecanismos de execução baseadas em múltiplos fluxos (intra-nó) e trocas de mensagens para o compartilhamento de dados (entre-nós) são necessárias. Esta seção apresenta duas ferramentas que foram selecionadas para uso neste trabalho: Threads POSIX e *Message Passing Interface* (MPI).

3.2.1 Multiprogramação leve

Esta seção aborda o modelo de programação baseado em multiprogramação, tendo como base o padrão POSIX para *threads*. A Seção 3.1.1 apresentou a arquitetura SMP que vêm sendo utilizada como nó em aglomerados de computadores. Como visto, esta arquitetura apresenta memória compartilhada que requer cuidados que permitam garantir a consistência nas operações (*load* e *store*) no acesso aos dados compartilhados. Esta seção aborda o modelo de programação baseado em multiprogramação, tendo como base o padrão POSIX para *threads*, tornando possível o compartilhamento de dados entre diferentes processos concorrentes.

Em um primeiro momento, a proposta de exploração da concorrência em computadores se deu através da execução de diversos processos concorrentes sobre os recursos de uma mesma arquitetura monoprocessada, caracterizando a concorrência temporal. Neste caso, além de aumentar a disputa do processador pelos processos,

o objetivo era compartilhar o mesmo espaço de endereçamento da memória para aumentar o desempenho global dos programas. Deste então, esta estrutura vêm sendo implementada acompanhando os avanços de *hardware* impostos pela indústria, e com as máquinas SMPs foi possível explorar a concorrência real (paralelismo). Desta forma, um grupo de instruções foi criada para possibilitar o controle do acesso às variáveis compartilhadas pelos múltiplos processadores. Estas instruções fornecem mecanismos de sincronização no acesso dos dados, tornando possível o acesso simultâneo de outros processos a mesma variável.

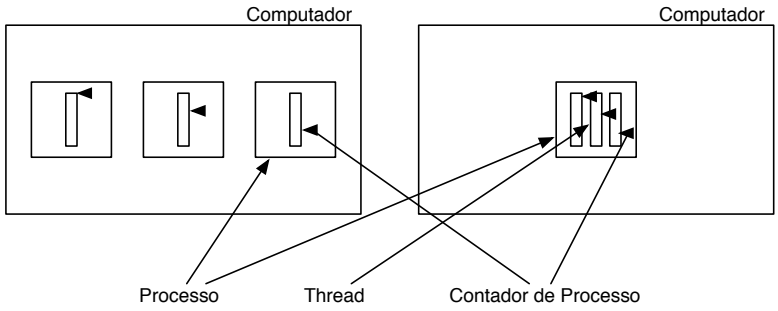

FIGURA 3.3 – Diagrama ilustrando a execução de *threads* e processos. A figura da esquerda ilustra três processos executando em computador, cada um com seu contador de programa. A figura da direita ilustra três processos leves executando no interior de um único processo.

A exploração de desempenho em uma única máquina abriu espaço para a multiprogramação leve, ou *multithreading*, que explora o mesmo modelo de programação oferecido pela utilização de processos concorrentes. Atualmente, a multiprogramacão leve permite que vários fluxos de execução sejam instanciados no interior de um processo (Figura 3.3). Cada um destes fluxos de execução recebe o nome de processo leve, ou simplesmente *thread*. O termo processo leve, expressa o custo de manipulação desta pelo sistema operacional, que é muito menos onerosa que a manipulação de processos pelo escalonador do sistema operacional. Uma característica importante da *thread* é o compartilhamento de memória com todas as outras *threads* ativas no contexto do processo em que se encontra. Em outras palavras, a área de memória do processo fica disponível a todas *threads* ativas, sendo necessárias apenas instruções para controle do acesso aos dados compartilhados.

Outra característica é em relação ao escalonamento das *threads* no processador. Uma aplicação não perde o processador no momento que uma *thread* é bloqueada (em função de instruções de *load* e *store*, por exemplo), pois outra *thread* torna-se apta a utilizar os recursos de processamento.

A implementação de *threads* encontra-se, atualmente, disponível em diferentes ferramentas, e possui implementações tanto em *hardware*, como em *software*. O restante deste documento descreve a implementação do suporte de *software* para *threads*, definido pelo padrão POSIX e implementado por diversos sistemas operacionais, como Unix, Linux e Mac OS X. A forma com que a *thread* é escalonada no uso do processador e quais os seus direitos sobre o processador depende do modelo de escalonamento implementado.

A manipulação de *threads* é feita sob um conjunto de primitivas definidas pelo padrão POSIX. Cabe ao programador descrever explicitamente a função que será executada em concorrência a outra. Esta função pode receber parâmetros através de ponteiros para a memória compartilhada e pode igualmente retornar dados via posição da memória. A criação de um fluxo de execução que permitirá a função definida pelo usuário de ser executada se dá através da invocação de uma primitiva `pthread_create`. Os parâmetros da instrução *join* consistem em um endereço de memória para uma variável que identifica a *thread*; possíveis atributos que possam ser considerados para sua execução e manipulação; a função que será executada e um ponteiro para uma região de memória com os argumentos da função. Em contraponto, a primitiva `pthread_join` apresenta apenas dois argumentos, o primeiro que identifica a *thread* da qual é requisitado o resultado e o segundo com o ponteiro para o endereço onde será armazenado o resultado retornado.

A comunicação entre *threads* não se limita aos parâmetros utilizados como entrada em uma *thread* ou ao resultado obtido através de uma instrução *join*. A própria memória do processo pode ser utilizada na comunicação entre as *threads*, através de instruções de leitura e escrita. O problema é assegurar que apenas uma das *threads* tenha acesso a um dado compartilhado em um determinado

instante, empregando mecanismos de sincronização como a *exclusão mútua*. Esta sincronização garante a coerência do dado, evitando que uma *thread* em execução acesse resultados errôneos ou incompletos sobre a execução. Neste caso, a sincronização tem por objetivo controlar a execução de conjuntos de instruções que acessam uma área de memória. É atribuído a este conjunto de instruções o nome de *seção crítica*.

Mutex, do inglês *mutual exclusion* é um construtor de sincronização que permite que uma *thread* tenha acesso exclusivo a uma área de dados. Tendo exclusividade sobre o dado, garante-se que uma seção crítica pode ser executada sem que outra *thread* tenha alguma instrução que manipule a mesma área de dados, que traria inconsistência a este valor.

O funcionamento do *mutex* é baseado em instruções do tipo *lock/unlock*. Quando uma *thread* entra em uma seção crítica, ela bloqueia o acesso a outras *threads*, impedindo o avanço delas pela seção (*lock*). Ao sair da seção crítica, o acesso volta a ser liberado a outras *threads* (*unlock*). A *thread* restará bloqueada aguardando a liberação do *mutex* caso realize uma operação *lock* enquanto outra *thread* esteja executando sua seção crítica. Neste caso o *lock* já foi pego, na execução da operação *unlock* uma das *threads* bloqueadas no *lock* será selecionada para assumir o *mutex*.

3.2.2 Message Passing Interface (MPI)

Considerando a execução de uma aplicação em um aglomerado de computadores, um dado produzido em um certo nó por um determinado processo poderá, eventualmente, ser consumido por outro processo em um nó distinto. Estas comunicações entre tarefas de nós diferentes é uma das operações mais custosas em se tratando de processamento concorrente em arquiteturas de memória distribuída. Esta seção apresenta o conceito de *troca de mensagens* e discute MPI, um padrão definido para troca de mensagens.

Trocas de mensagens são soluções clássicas para o problema de comunicação em ambientes de memória distribuída que permitem realizar comunicações introduzindo um sobrecusto à execução da aplicação. O mecanismo de comunicação é basicamente síncrono, isto é, para cada mensagem enviada por um processo, deve existir um destinatário correspondente (*send* → *receive*), embora algumas bibliotecas implementem também variantes assíncrona destas primitivas. Cada mensagem conterá, em seu cabeçalho, o endereço de uma função a ser executada no receptor, e no seu corpo, os dados que devem ser processados por esta função.

Message Passing Interface, ou MPI, como é normalmente referenciado, é um padrão *de facto* para programação paralela para ambientes distribuídos que implementa o mecanismo de *troca de mensagens*. Foi definida por um órgão denominado MPI Forum[3] com a participação de Universidades, laboratórios de pesquisa e indústria. Neste Fórum foram elaboradas primitivas que permitem a sincronização e comunicação de tarefas. O padrão MPI não suporta o conceito de espaço de endereçamento global, sendo a comunicação entre processos feita, exclusivamente, através de troca de mensagens, o que caracteriza suas aplicações como programas distribuídos.

Em execução, uma aplicação MPI cria um conjunto fixo de processos no momento de sua inicialização[4]. Embora cada processo possa executar em um processador físico com um programa diferente, o que o caracterizaria como MPMD (Múltiplos Programas Múltiplos Dados - *Multiple Program Multiple Data*), a forma usual de programar utiliza o modelo SPMD (Um Programa Múltiplos Dados - *Sigle Program Multiple Data*), no qual uma única aplicação é executada em cada um dos processadores e a execução de trechos do programa é feita por processador.

MPI engloba uma série de conceitos que são abordados abaixo:

- **Processo**: O número de processos em execução é indicado pelo número de processadores especificados pelo usuário no momento da execução. Quando o número de processos criados é maior que o número de processadores físicos

[3] http://mpi-forum.org
[4] A versão MPI-2 do padrão permite variação no número de nós

disponíveis, a criação de processos é cíclica, distribuindo os processos de acordo com a lista de processadores especificada na configuração do ambiente.

- **Mensagem**: É o conteúdo da comunicação entre dois processos, formada por:
  - **Envelope**: Responsável pela identificação dos processos, tanto o transmissor como o receptor. Contém, também, o rótulo da mensagem.
  - **Dado**: Contém os dados que se deseja enviar ou receber. Contém o endereço de onde o dado se localiza; o número de elementos do dado na mensagem e o tipo do dado.
- **Rank**: Quando um processo é inicializado, recebe uma identificação única atribuída pelo sistema. Este valor, definido como um *inteiro* é utilizado para identificar explicitamente o processo destinatário ou o remetente de uma mensagem, *send* e *receive*, respectivamente.

Estes conceitos estão ligados a um conjunto de características que definem MPI. Sua interface de programação, por exemplo, fornece as primitivas a tais conceitos, independe da linguagem de programação utilizada (podendo ser C/C++, FORTRAN ou mesmo Java). Independente da linguagem, permite programação para multiprocessadores e redes heterogêneas, podendo ser utilizada tanto para o desenvolvimento de aplicações finais, quanto como ambiente de programação. Em ambas arquiteturas, máquinas multiprocessadas ou aglomerados, MPI permite a programação SPMD através de comunicação síncrona e assíncrona. No primeiro caso, o nó que invocar a primitiva `MPI_Send` ficará bloqueado até que a comunicação seja concluída. Da mesma forma, a primitiva `MPI_Recv` bloqueia o nó até que a mensagem desejada chegue na fila de mensagens. Caso o bloqueio não seja desejado, MPI apresenta um conjunto de primitivas não bloqueantes (assíncronas): `MPI_Isend` e `MPI_Irecv`.

3.3 Escalonamento

Esta seção discute as operações realizadas pelo escalonador para controlar a execução de aplicativos paralelos. Estas operações permitem controlar a distribuições dos dados na memória e mapear as tarefas sobre uma arquitetura distribuída.

3.3.1 Mapeamento de dados

Para controlar o acesso, e a distribuição, dos dados sobre os módulos de memória disponíveis em um arquitetura distribuída é necessária uma rede de conexão.

O escalonamento aplicativo gerencia a atribuição local e a migração de dados sobre os módulos de memória da arquitetura paralela. O gerenciamento local consiste em atribuir um dado em um dos módulos de memória disponível. A migração consiste em mover o dado de um modulo à outro. Estas operações são realizadas através de troca de mensagens.

Com o objetivo de reduzir o *overhead* introduzido pela troca de dados, diferentes técnicas foram desenvolvidas, seja para reduzir o número de mensagens ou mascarar este *overhead* necessário com cálculos efetivos.

Para reduzir o número de troca de mensagens, aplicam-se técnicas que buscam aumentar a localidade dos dados, acessando-os, sempre que possível, localmente. Um dos métodos utilizados neste caso consiste em realizar as operações sobre um determinado dado naquele nó que o contém em seu módulo de memória. À esta técnica é dada o nome de *owner-compute-rule*, e seu princípio consiste em distribuir pelos módulos de memória dos processadores os dados e executar, sobre tal processador, operações sobre esses dados. Outras técnicas agrupam em um mesmo processador as operações que executam comunicação entre si [Karypis and Kumar, 1995, Yang and Gerasoulis, 1992], ou empregam a duplicação dos dados sobre todos os nós [Jacqmot, 1996].

Em casos onde não se pode evitar a troca de mensagens, aborda-se a

necessidade de utilizar os processadores enquanto a comunicação não ocorra. Tal aproveitamento dos processadores pode ser explorado antecipando as comunicações entre os nós, preparando os dados e as mensagens para que estejam disponíveis no momento que requisitadas. Outra técnica é aproveitar o processador com uma *thread* enquanto outra está bloqueada realizando a troca de mensagens (*multithreading*) [Valiant, 1990].

3.3.2 Mapeamento de tarefas

O problema encontrado no mapeamento de tarefas consiste em determinar quais serão executadas sobre qual processador e o momento em que serão executadas.

Tal operação de mapeamento consiste em atribuir tarefas aos processadores. Neste momento, estas podem estar prontas para serem executadas ou não. Uma tarefa pronta é considerada executável a partir do momento quem ela é mapeada sobre um processador. Uma tarefas não pronta não deve ser executada antes que suas dependências tenham sido satisfeitas.

Em relação ao mecanismo de tarefas, duas abordagens podem ser consideradas:

- **Lançamento Automático**: a tarefa é lançada no momento em que estiver mapeada e todas suas dependências foram resolvidas; e,
- **Lançamento Retardado**: o lançamento não é automático, sendo definido por alguma estratégia de escalonamento [Lowenthal et al., 1996].

Embora possam existir diversas tarefas no estado de execução, em um determinado instante de tempo, apenas uma é executada pelo processador. Para que ocorra o compartilhamento do tempo de processamento é necessário um mecanismo de preempção;

- **Não preemptivo**: uma vez lançada uma tarefa, ocupa o processador até terminar sua execução ou chamar, explicitamente, uma operação de sincronização.

- **Preemptivo**: a execução é dada no processador em uma fatia de tempo. No final desta fatia, ou em uma operação de sincronização, o processador é liberado, até que ganho novamente a possibilidade de retornar a execução.

Outra questão é como garantir uma repartição homogênea de cargas entre os processadores. Tal repartição é realizada no início da execução, no primeiro momento, pelo mapeamento de tarefas. Durante a execução da aplicação, no entanto, este mapeamento pode sofrer desbalanceamento. São consideradas duas alternativas

- **Migração restrita**: apenas tarefas mapeadas ainda não lançadas são consideradas. Adaptado para casos onde não exista preempção de tarefas; e,

- **Migração de tarefas em execução**: Bem mais oneroso, requer a preempção da tarefa, empacotamento, envio e relançamento.

3.4 Implementações de Simulações Distribuídas

A literatura apresenta um número considerável de trabalhos relacionados à simulações de sistemas naturais, seja abordando características químicas, físicas ou biológicas, conforme a classificação de Levin [Levin, 1989], para os problemas denominados "grandes desafios" à ciência:

- Química quântica, mecânica estatística e física relativística;

- Cosmologia e astrofísica;

- Dinâmica de Fluidos e turbulência;

- Física do estado sólido e supercondutividade;

- Biologia, farmacologia, sequenciamento de genoma, engenharia genética, encapsulamento de proteínas, atividade enzimática e modelagem celular;

- Medicina e modelagem de órgãos e tecidos; e,
- Meteorologia e clima global.

Muitos dos trabalhos realizados nestas áreas utilizam o método de Monte Carlo, definindo sistemas fracamente acoplados, que são trivialmente paralelizáveis [Coddington, 1993, Lemke, 2002]. Em algumas implementações, diferentes simulações podem ser obtidas de forma completamente independente, através da alteração de parâmetros de inicialização, e o resultado final é obtido através da combinação das saídas geradas por todas as simulações, reduzindo o erro estatístico. Outras implementações exploram a decomposição de domínios [Cercato et al., 2006, Koradi et al., 2000, Plimpton and Hendrickson, 1996, Srinivasan et al., 1997]. Em tais casos, o sistema simulado é dividido em regiões no espaço. Desta forma, a simulação pode ser processada de forma independente em cada região. Além disto, algumas aplicações necessitam dos resultados das bordas das regiões vizinhas, como apresentado em [Dorneles et al., 2003].

Plimpton discutiu estratégias de divisão [Plimpton and Hendrickson, 1996] no contexto de simulações de dinâmica molecular, onde o elemento atuante na simulação é identificado como *átomo*. O primeiro algoritmo apresentado pelo autor é denominado *decomposição atômica*. Neste algoritmo, átomos representam a unidade de trabalho, sobre os quais o cálculo pode se realizado de forma independente. O segundo algoritmo, denominado *decomposição de forças*, define como unidade de trabalho um bloco de átomos, sendo o sistema a ser simulado em blocos de igual tamanho. O terceiro algoritmo, intitulado *decomposição espacial*, divide o espaço em termos de células 3D. A análise encontrada em [Plimpton and Hendrickson, 1996] discute estes algoritmos em termos de facilidade de implementação, custos de comunicação e potencial de balanceamento de carga. O algoritmo de decomposição atômica foi caracterizado como uma estratégia de simples implementação, com um alto potencial de balanceamento de carga, mas com alto custo de comunicação. O algoritmo de decomposição de forças também é de implementação relativamente simples e apresenta potencial para balanceamento de carga e demanda menor carga

de comunicação. No entanto, não é escalável para problemas maiores. Finalmente, o algoritmo de decomposição espacial é de implementação mais complexa, sofre mais facilmente de desbalanceamento de cargas, embora mais escalável.

Dois outros modelos de simulação são apresentados com maiores detalhes. O primeiro interessa pelos recursos de implementação utilizados e o segundo pela proximidade ao problema tratado no presente trabalho.

3.4.1 Modelo de Potts

Gusatto e colaboradores [Gusato et al., 2005] descreveram um novo esquema para simular estruturas celulares (células biológicas, bolhas ou grãos de policristais) baseados no modelo celular de Potts. No modelo original, cada partícula é calculada aleatoriamente, utilizando a estratégia de Monte Carlo. Logo, a cada *timestep*, cada partícula tem possibilidade igual de ser escolhida para ser computada. Como a computação do novo estado da partícula requer como entrada os estados das partículas vizinhas, este modelo introduz uma grande quantidade de sincronizações. Propuseram uma solução baseada na redução da aleatoriedade quando uma partícula é escolhida, reduzindo os requerimentos de sincronização.

Em trabalho posterior [Cercato et al., 2006], apresentaram uma implementação para aglomerados de computadores baseada em troca de mensagens e multiprogramação leve. Uma "zona fantasma" envolve cada região com os dados das partículas vizinhas. Durante a etapa de cálculos não existe sincronização, sendo esta explorada apenas quando existe alteração em tal zona. O trabalho utilizou o uso de ferramentas padrão para troca de mensagens, *sockets* e *threads* POSIX para multiprogramação leve.

3.4.2 Modelo de Quinn para forças sociais

Quinn e colaboradores [Quinn et al., 2003] descreveram uma simulação de forças sociais utilizando aglomerados de computadores como ambiente de execução. Exploraram o paralelismo de dados segundo o modelo mestre/escravo, o qual permite

o controle da evolução da simulação. Nesta implementação, uma lista global contendo regiões a serem calculadas é gerenciada em um nó mestre, enquanto nós trabalhadores consomem estas regiões para realizar o processamento da simulação propriamente dita. O fato de que há cálculo das regiões vizinhas é também considerado. Nesta implementação, uma "zona fantasma" cerca cada região: cada zona fantasma contém cópias dos agentes controlados por outras regiões. A sincronização da simulação é realizada de forma distribuída: ao final do cálculo de um *timestep*, cada região envia para suas regiões vizinhas os agentes que se deslocaram na direção destas, também recebendo os agentes que vieram para o seu interior. A implementação desta aplicação foi realizada exclusivamente em MPI, sem utilizar multiprogramação e não levando em consideração as características da simulação de humanos virtuais em situações de emergência (que gerou desbalanceamento de carga).

3.5 Conclusão

O modelo de simulação com solução bidimensional exige o uso de técnicas de processamento de alto desempenho para obter resultados em tempo hábil, pois, como discutido no Capítulo 2, trata-se de um modelo Físico de complexidade $\mathcal{O}(n^2)$, onde n é o número de agentes que se deseja simular. A alternativa escolhida para o desenvolvimento deste trabalho foi feita visando utilizar o aglomerado de máquinas SMPs disponíveis nas instalações do Programa Interdisciplinar de Pós Graduação em Computação Aplicada, da Universidade do Vale do Rio dos Sinos.

Porém, como discutido neste capítulo, a construção de um programa que execute em um aglomerado requer questões importantes que devem ser consideradas. A primeira é referente a memória. Além do custo computacional, claramente evidente, esta classe de simulação exige muita manipulação da memória física. Este capítulo apresentou duas ferramentas que visam suprir estas necessidades. Para concorrência intra-nó foi apresentado o padrão POSIX para threads, capaz de gerenciar a mesma região de endereçamento para diferentes processos. Para

concorrência entre-nós, discutiu-se o padrão MPI, que se utiliza de uma série de primitivas para troca de mensagens entre processos que possam residir em domínios (máquinas) diferentes.

Outra questão que deve ser considerada, é o balanceamento de cargas entre os nós de um aglomerado. Uma distribuição de cargas que leve em consideração as características da aplicação visando diminuir o número de comunicações entre os nós. De qualquer forma, é inevitável o uso de trocas de mensagens em um aglomerado, e a melhor forma de abordar esse *overhead* dos tempos de comunicação é através da exploração de *threads* em uma concorrência intra-nó para cálculo efetivo [Valiant, 1990] reduzindo o impacto das comunicações. O modelo apresentado na Seção 3.4.2 não apresenta este balanceamento de carga descrito em função da aplicação, nem explora a concorrência intra-nó em sobreposição aos tempos de comunicação, ao contrario da solução apresentada por Cercato (Seção 3.4.1) que utiliza *threads* e comunicação através de *sockets*.

Capítulo 4

Modelo Tridimensional para Forças Sociais

Conforme descrito, os atuais centros urbanos concentram uma profusão de indivíduos, e grande parte deles em estruturas e edificações complexas, tais como ginásios esportivos, centros comerciais e aeroportos. Estes ambientes, tridimensionais, apresentam claramente um grau de liberdade extra àqueles suportados pelos modelos atuais de simulação. Esta é uma limitação dos modelos baseados em Física encontrados na literatura que, embora apresentem resultados satisfatórios de simulação, são incapazes de simular agentes em terrenos irregulares ou com a presença de rampas e escadas.

Este capítulo introduz o modelo tridimensional de simulação de multidões, baseado em Física, proposto neste trabalho.

4.1 Nomenclatura

Este capítulo utiliza conceitos e palavras para definir propriedades específicas, sejam em relação à geometria do ambiente físico ou ao modelo matemático. Todo plano é identificado por uma letra grega minúscula, por exemplo γ e π, como usual na matemática. Um plano paralelo a superfície terrestre é definido como um **andar**. A mudança de andares é feita por em uma **rampa**, isto é, um plano inclinado por

um ângulo α (tal que $0 < \alpha < \pi/2$) em relação a superfície terrestre. Em cada andar e rampa existe uma região denominada **atrator**, uma região espacial para onde os agentes de um determinado plano se deslocam. Cada andar, rampa ou sala (ambiente cercado por paredes) é denominado **contexto**. Todo agente que deixa uma destas regiões e passa à outra, está realizado uma troca de contexto. Portanto, cada contexto apresenta um, ou mais, atratores. A troca de contexto é realizada quando o agente atinge o atrator daquele ambiente.

4.2 Ambientes Tridimensionais

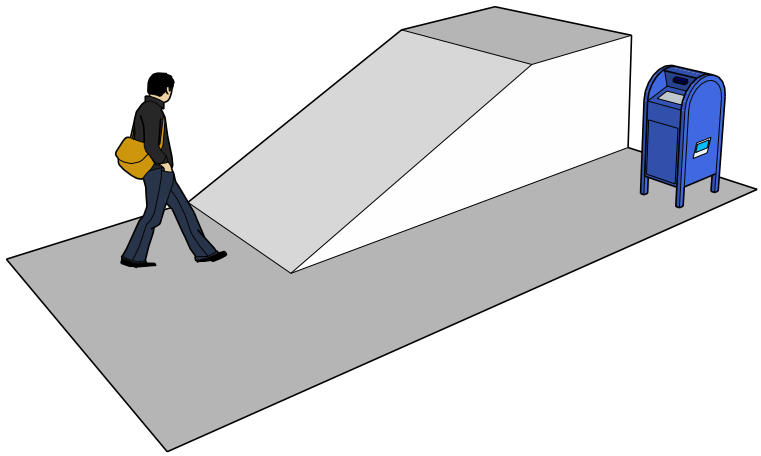

FIGURA 4.1 – Representação em ambiente virtual de um ambiente tridimensional, contendo uma rampa entre dois planos de níveis diferentes e uma caixa postal como obstáculo estático.

A Figura 4.1 apresenta um ambiente virtual tridimensional formado por três planos, dois horizontais e paralelos, unidos por um terceiro, representando uma rampa, e um obstáculo estático representado por uma caixa postal. Nesta configuração descrita, caso tenha seu objetivo situado no andar superior, o agente deve estar apto a subir a rampa. Não obstante, o movimento do agente na configuração deve ser compatível com o modelo de Helbing, ou seja, regido por equações físicas do movimento. Portanto, para tais situações, é importante que o modelo matemático sofra alterações que permitam descrever o novo sistema de

forças atuantes sobre cada agente bem como a relação de interação entre agentes distintos. Desta forma, assume-se dois postulados:

Postulado 4.2.1 *Dois agentes i e j em planos γ e π, respectivamente, não devem interferir na trajetória de outrem quando γ e π não forem conexos. Ou seja, dois agentes localizados em planos não conectados não devem interferir na trajetória um do outro.*

Postulado 4.2.2 *O agente localizado em uma rampa deve apresentar reação à superfície, normal do plano, e deve estar sob influência de um conjunto de componentes referentes à ação gravitacional.*

É importantes destacar que no Postulado 4.2.1 a palavra *planos* é utilizada tanto para andares, como para rampas.

4.3 Modelo Físico

Em seu modelo de forças sociais, Helbing descreve o somatório de forças sobre um agente i, expresso por $\vec{F}_i^H$, com massa m_i e velocidade $\vec{v}_i$ por uma equação diferencial (Eq-4.1)[1], onde o primeiro termo a direita da igualdade refere-se a força de direcionabilidade, responsável por governar o agente à velocidade desejada de módulo v_i^0 na direção $\hat{e}_i^0$ em um tempo τ_i. Os termos restantes são responsáveis pelo efeito repulsivo, de caracter social, entre o agente e os $n-1$ demais (para todo $i \neq j$) e entre o agente e as paredes w, segundo e terceiro termos, respectivamente.

$$m_i \frac{d}{dt}\vec{v}_i = \vec{F}_i^H = m_i \frac{v_i^0 \hat{e}_i^0 - \vec{v}_i(t)}{\tau_i} + \sum_{\substack{i=1 \\ i \neq j}}^{n} \vec{f}_{ij} + \sum_{w} \vec{f}_{iw} \tag{4.1}$$

Esta equação apresenta uma solução válida apenas em um ambiente bidimensional. A inexistência de limitações espaciais neste modelo, tal como a presença de rampas (como apresentado na Figura 4.1), não permite, simplesmente,

[1] A equação já foi apresentada no Capítulo 2 e foi reproduzida para facilitar o desenvolvimento da leitura.

a inclusão de uma nova componente espacial. Em uma situação onde agentes encontram-se em andares distintos a adição desta componente, acrescido ao fato da força de interação entre agentes ser na direção do raio vetor que os une, implicaria em uma força de repulsão mútua em sentidos inaplicáveis no mundo real.

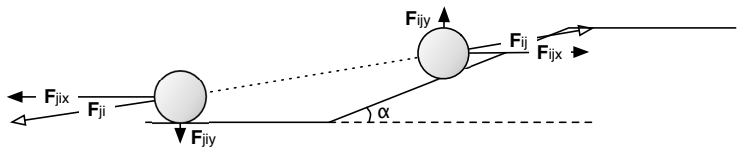

FIGURA 4.2 – Diagrama de corpo livre dos agentes i e j explorando as forças de repulsão social associada a eles e componentes das mesmas no plano de movimento dos agentes.

A Figura 4.2 apresenta a linha pontilhada na direção que une os dois agentes e os vetores $\vec{F}_{ij}$ e $\vec{F}_{ji}$ como força repulsiva entre os agentes i e j. Estas forças, iguais em módulo com diferença apenas no sentido, não são paralelas ao plano do movimento dos agentes (andares) e seus resultados são indesejados, pois acrescem ao conjunto de forças atuantes sobre o agente uma aceleração de orientação impraticável.

Além da adição de uma nova componente espacial ao modelo, o mesmo deve sofrer modificações que garantam a **distinção de planos** apresentada no Postulado 4.2.1, evitando a interação entre agentes nestas situações críticas. A próxima seção aborda o tópico matemático sobre planos e define um novo termo para a eliminação dos problemas apontados.

4.4 Distinção de Planos

Esta seção apresenta, em linhas gerais, uma visão matemática do *plano*, envolvendo suas propriedades e definições matemáticas. A seção finaliza sugerindo a primeira alteração no modelo de Helbing para prover suporte à ambientes tridimensionais.

4.4.1 Definição de um plano

A Geometria Analítica, comumente denominada de geometria cartesiana – em referência ao filósofo e matemático francês René Descartes (1596-1650) – é o ramo da matemática que estuda a geometria utilizando princípios da álgebra. Permite descrever, através de equações, vetores, retas, planos e outras superfícies quadráticas com base nas coordenadas cartesianas.

Nesta seção o foco é o plano, espaço matemático que comporta todos os pontos em de uma descrição espacial bidimensional. No espaço tridimensional, é um objeto de duas dimensões que comporta um conjunto de pontos sob a condição de *coplanariedade*.

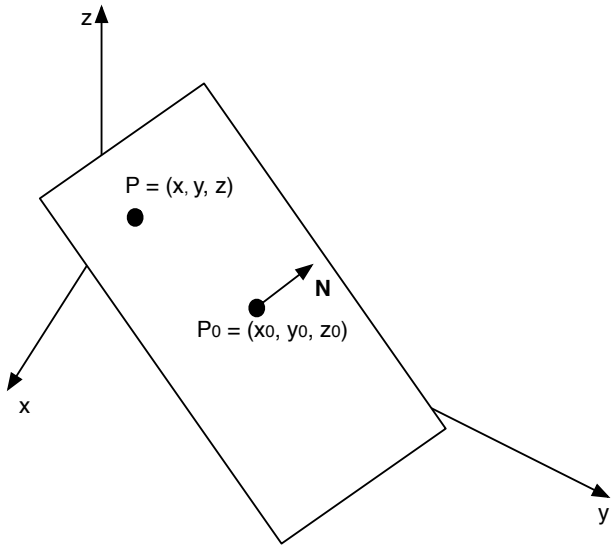

FIGURA 4.3 – Representação de um plano genérico, em espaço tridimensional, com vetor normal $\vec{n}$. A derivação da equação do plano (Eq-4.4) utiliza o ponto $P_0 = (x_0, y_0, z_0)$, que define o vetor normal, e um ponto qualquer $P = (x, y, z)$.

Um vetor $\vec{N} = N_x\hat{\imath} + N_y\hat{\jmath} + N_z\hat{k}$ define uma família de planos paralelos, e normais a ele. É atribuído ao vetor $\vec{N}$ o nome de *vetor normal* e a especialização de um plano, isto é, a distinção de um plano entre os demais, é feita através da escolha de um ponto $P_0 = x_0\hat{\imath} + y_0\hat{\jmath} + z_0\hat{k}$ pertencente a este plano e que seja colinear ao vetor normal. Desta forma um ponto $P = x\hat{\imath} + y\hat{\jmath} + z\hat{k}$, qualquer, do plano e o ponto P_0 formam um vetor $\vec{P_0P} = (x - x_0)\,\hat{\imath} + (y - y_0)\,\hat{\jmath} + (z - z_0)\,\hat{z}$ perpendicular ao vetor normal $\vec{N}$. Assim, o produto escalar entre os vetores $\vec{N}$ e $\vec{P_0P}$ é igual a

zero, como mostrado na Equação 4.2. A Equação 4.3 representa a mesma condição da igualdade anterior, porém exposta em sua forma estendida.

$$\vec{N} \cdot \vec{P_0 P} = 0 \tag{4.2}$$

$$N_x (x - x_0) + N_y (y - y_0) + N_z (z - z_0) = 0 \tag{4.3}$$

ou, como referenciado na literatura:

$$N_x x + N_y y + N_z z = -d \tag{4.4}$$

onde $d = -N_x x_0 - N_y y_0 - N_z z_0$.

Voltando ao problema físico em voga, a verificação de coplanariedade de dois agentes i e j pode ser restringida à verificação da existência de um ponto em um plano. Através da equação do plano (Eq-4.4) do agente i, plano-γ, por exemplo, verifica-se se o ponto espacial onde está localizado o agente j encontra-se neste.

A partir da posição espacial do agente i, é possível encontrar um vetor normal ao seu plano (considerando a posição do agente i como ponto P_0 na definição do plano descrita). A verificação de coplanariedade é feita atribuindo a posição do agente j ao ponto genérico P. Caso a igualdade da Equação 4.3 não se satisfaça, fica evidente a não-coplanariedade dos agentes.

4.4.2 Epsilon de omissão de termos

É possível estabelecer uma analogia com a função delta de Kronecker[2] para expressar o Postulado-4.2.1, que exprime a distinção de planos entre agentes, através de uma notação compacta ϵ_{ij}, definida na Equação 4.5.

$$\epsilon_{ij} = \begin{cases} 1, & \text{se } i \text{ e } j \text{ pertencem a planos conexos} \\ 0, & \text{se } i \text{ e } j \text{ não pertencem a planos conexos} \end{cases} \tag{4.5}$$

[2] O delta de Kronecker, que pode ser interpretado como uma versão discreta da função delta, é definido por $\delta_{ij} = 0$ quando $i \neq j$ e $\delta_{ij} = 1$ quando $i = j$.

Desta forma, quando dois agentes em interação social não pertencerem ao mesmo plano, o resultado do épsilon de omissão de termos tem valor nulo, eliminando o termo de repulsão da equação de movimento. Em outras palavras, isto significa que o termo é igual a 1 (um) apenas na condição de coplanariedade e de conectividade dos planos dos agentes, conforme garantido pelo postulado de **distinção de planos**. Matematicamente, isto significa transformar o termo de colisão entre dois agentes da equação de movimento (Eq-4.1) em um novo termo, representado na troca da Equação 4.6.

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} \vec{f}_{ij} \rightarrow \sum_{\substack{i=1 \\ i \neq j}}^{n} \vec{f}_{ij} \epsilon_{ij} \tag{4.6}$$

4.5 Sistema de Forças Físicas

Matematicamente é possível expressar um conjunto de forças físicas atuantes sobre uma partícula através de uma única força resultante $\vec{F}_{res}$ cujo resultado é obtido através da soma vetorial de todas as n forças $\vec{f}_i$ (para $i = 1, 2, ...n$) que agem sobre ela (Eq-4.7). Essa força é, então, vista como a única força do sistema e seu resultado sobre a partícula é idêntico ao apresentado pelas outras forças quando em conjunto.

$$\vec{F}_{res} = \sum_{i=1}^{n} \vec{f}_i \tag{4.7}$$

Esta característica, que pode ser encontrada em livros didáticos de Física Geral, foi explorada na literatura de Humanos Virtuais por Helbing, Braun e Seyfried, entre outros, ao introduzir novos termos à equação de movimento. Helbing, conforme descrito no Capítulo 2, somou ao termo de direcionabilidade o caracter social. Braun descreveu a individualidade de agentes permitindo a interação entre agentes com dependência de movimento com outros altruístas. Seyfried [Seyfried et al., 2006] suprimiu o termo social introduzido por Helbing e descreveu um novo termo baseado na relação velocidade-densidade.

Semelhantemente, este trabalho propõe a inclusão de um novo termo ($\vec{F}_i^{nova}$) responsável pela influência gravitacional sobre o agente, e reações a superfície, conforme apresentado na Equação 4.7. O modelo resultante deve ser capaz de representar as forças atuantes sobre um agente i, em um instante de tempo qualquer, na situação em que o agente se encontra. A proposta de um novo termo à equação de movimento estabelece, então, uma nova forma a equação

$$\vec{F}_i = \vec{F}_i^H + \vec{F}_i^{nova} \tag{4.8}$$

4.5.1 Forças atuantes

Esta seção apresenta a segunda alteração ao modelo, uma descrição da força $\vec{F}_i^{nova}$ especificada no Postulado-4.2.2.

Com o objetivo de tornar o termo mais adequado ao movimento tridimensional de um agente, considera-se um agente sujeito a resistência do ar, além da resistência encontrada com o solo (força de atrito cinético de contato). Estas forças, que não se encontram no modelo proposto por Helbing, fornecem uma tupla $\{\gamma, \mu_c\}$ de parâmetros que podem ser ajustados de acordo com dados aferidos de situações reais, e representam, respectivamente, o coeficiente de resistência do ar e de atrito cinético. Isto se faz importante para reduzir o impacto da força gravitacional sobre o agente, tratado até o presente momento como uma esfera. Caso contrário, um agente estaria livre para "escorregar" até a região inferior da rampa.

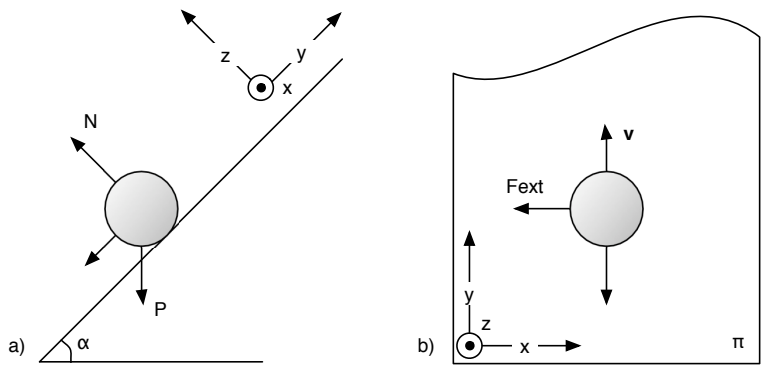

FIGURA 4.4 – Representação de um agente em uma rampa sob ação de forças externas (Interação + Peso)

As expressões utilizadas para os termos de resistência são encontrados empiricamente, na Física, e tem-se que a força de resistência do ar, $\vec{R}$ é proporcional a velocidade do corpo, podendo ser escrita da seguinte forma:

$$\vec{R} = -\gamma\vec{v} \tag{4.9}$$

Da mesma forma, fenomenologicamente, sabe-se que o atrito cinético de contato $\vec{f_c}$ é proporcional à reação normal entre as superfícies em contato e não entre a área destas superfícies, ao contrário do que se possa esperar. Desta forma, a Equação 4.10 exprime a relação de proporcionalidade através de uma constante μ_c que representa o coeficiente de atrito cinético. Neste termo, ao contrário da Equação 4.9, a relação não é vetorial, e o valor da força de atrito é atribuído proporcionalmente ao módulo da normal N.

$$f_c = \mu_c N \tag{4.10}$$

Percebe-se, a partir da Figura 4.4, que na direção representada pelo eixo z não há movimento e, portanto, a componente da força peso P naquele eixo iguala-se ao valor da normal à superfície (Eq-4.11).

$$\begin{aligned} N &= P_z \\ N &= P\cos(\theta) \\ N &= mg\cos(\theta) \end{aligned} \tag{4.11}$$

No entanto, na direção do eixo y, tem-se os novos termos de atrito, e a força resultante atuante sobre o corpo nesta direção, $\vec{F_y}$, é descrita por:

$$\begin{aligned} \vec{F}_y &= \vec{P}_y + \vec{f_c} + \vec{R} \\ F_y &= P_y - f_c - R \\ m\frac{d}{dt}v_y &= mg\sin\theta - \mu_c N - \gamma v_y \end{aligned} \tag{4.12}$$

4.5.2 Modelo proposto

As alterações descritas aqui, representam uma formalização matemática dos postulados apresentados na primeira seção deste capítulo: *i*) distinção de planos, e *ii*) reação a superfície e interação gravitacional. Ambas modificações acrescentam novos termos à equação de movimento do Helbing (Eq-4.1), que reescrita é apresentada na Equação 4.13.

$$m_i\frac{d}{dt}\vec{v}_i = \vec{F}_i = m_i\frac{v_i^0\hat{e}_i^0 - \vec{v}_i(t)}{\tau_i} + \sum_{\substack{i=1 \\ i\neq j}}^{n}\vec{f}_{ij}\epsilon_{ij} + mg\sin\theta - \mu_c N - \gamma\vec{v} \tag{4.13}$$

onde $\vec{F}_i$ é a força resultante sobre o agente i.

Estas modificações, embora acrescentem ao algoritmo um conjunto novo de operações, não alteram o número de interações entre agentes do modelo prévio e planar. A verificação de coplanariedade não elimina a necessidade de cálculo entre dos agentes i e j, mas apenas rejeita ou não o resultado daquela interação. Desta forma, este método apresenta $1/2n(n-1)$ interações em cada ciclo de processamento, mantendo sua complexidade $\mathcal{O}(n^2)$.

4.6 Posicionamento em Relação ao Estado-da-Arte

O modelo proposto neste trabalho expande os modelos atualmente utilizados para simulação de humanos virtuais em situações de evacuação através da inclusão de uma nova direção espacial, que proporciona descrever ambientes tridimensionais,

e um novo conjunto de termos à equação proposta por Helbing.

Embora compatível com os modelos de forças sociais, este modelo inova em buscar por simulações em ambientes físicos tridimensionais, que permitam uma descrição detalhada de ambientes complexos.

Capítulo 5

Protótipo Desenvolvido

Este capítulo descreve o protótipo elaborado durante esta pesquisa para avaliar o modelo matemático de comportamento de multidões durante situações emergenciais em ambientes tridimensionais. Conforme descrito no Capítulo 4, o modelo proposto é uma extensão daquele desenvolvido por Helbing e colaboradores [Helbing et al., 2000], e utiliza o conceito de forças sociais para interações e tratamento de colisão entre os agentes virtuais. Além disto, o capítulo expõe as ferramentas utilizadas nesta etapa do trabalho e as fases do desenvolvimento do protótipo.

5.1 Ferramentas Utilizadas

O tratamento dado pelo aplicativo Mathematica [Wolfram, 1991, Cordeiro, 2006], da Wolfram Research, às questões referentes à Matemática, Física e outros campos da Ciência e Indústria, fortaleceram a escolha da aplicação como base para o desenvolvimento do protótipo capaz de simular agentes sob situações de pânico em ambientes tridimensionais.

Não apenas oferecendo uma interface para cálculos matemáticos simples e fácil de prototipar, o Mathematica oferece uma linguagem de programação própria, de múltiplos paradigmas, comunicação com aplicações externas, como FORTRAN/C/C++/Java, e permite a criação de extensões às funções já suportadas

nativamente por sua arquitetura, através de pacotes de expansão (*Mathematica Packages*).

As seções seguintes descrevem com maiores detalhes a arquitetura do protótipo confeccionado sobre o Mathematica e os *Packages* criados para descrição de ambientes tridimensionais e simulação nestes ambientes.

5.2 Descrição da Arquitetura

Do ponto de vista de uma aplicação em execução, o simulador apresenta um início, seguido de uma seqüência de instruções que levam ao resultado esperado no término da execução. É importante, ainda, em um protótipo de simulador, a existência de diversos componentes capazes de retornar diferentes resultados durante uma aquisição de dados, permitindo que estes sejam aproveitados de diferentes maneiras. Por exemplo, para análise numérica de consistência de dados ou visualização gráfica. Esta seção descreve este fluxo de instruções do simulador e suas particularidades em relação ao modelo proposto no Capítulo 4.

A Figura-5.1 ilustra o fluxo de execução do simulador, abstraindo a etapa de interação entre os agentes, fase que é descrita na seqüência do capítulo. Torna-se evidente a separação em três etapas: *i*) entrada de dados e pré-processamento, *ii*) processamento e *iii*) retorno de resultados. Cada etapa, apresentada abaixo, será discutida com maior número de detalhes na seqüência deste capítulo.

- **Entrada de dados e pré-processamento**: a primeira etapa do fluxo de execução do simulador é atribuída à leitura das informações relevantes: descrição do ambiente físico onde a simulação irá executar, informações estatísticas sobre os agentes virtuais e parâmetros de configuração do modelo matemático. É realizado ainda o pré-processamento dos dados, para avaliar a consistência dos mesmo. Inconsistências nos dados interrompem a execução da simulação.

- **Processamento**: a etapa de processamento é subdividida em duas, uma que

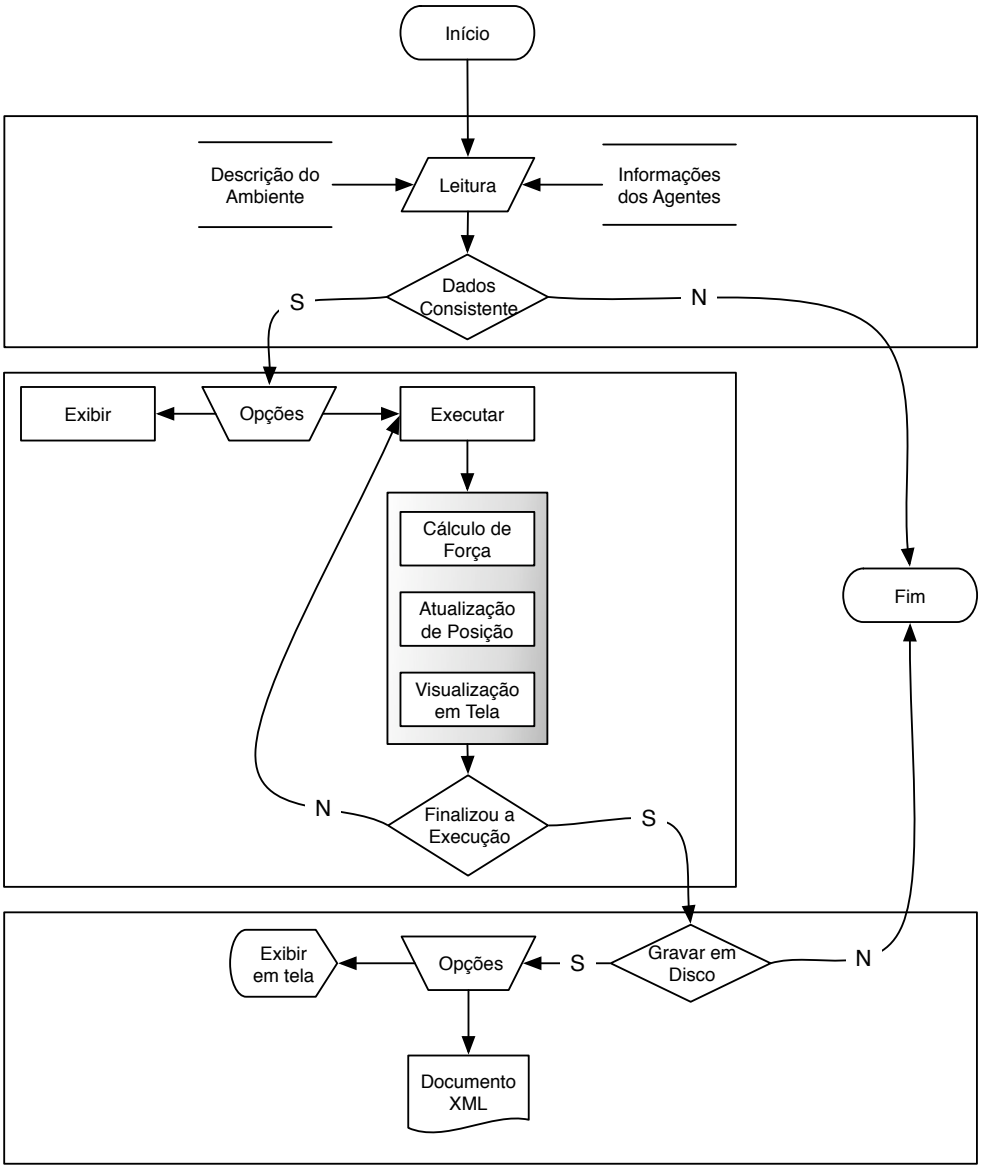

FIGURA 5.1 – Fluxo de execução do simulador. Na figura, detalhes sobre as interações entre os agentes são simplificados e descritos na Figura 5.2 com maior número de detalhes.

permite execução da simulação e outra que permite a visualização do ambiente tridimensional especificado como entrada na primeira etapa.

- **Retorno de Resultados**: Após o término da execução da simulação, é possível exportar os dados produzidos de duas formas distintas. A primeira é através de um arquivo XML, com especificações do ambiente e dos agentes em cada passo de simulação (δt) e a segunda, através de uma seqüência animada de cada um destes passos.

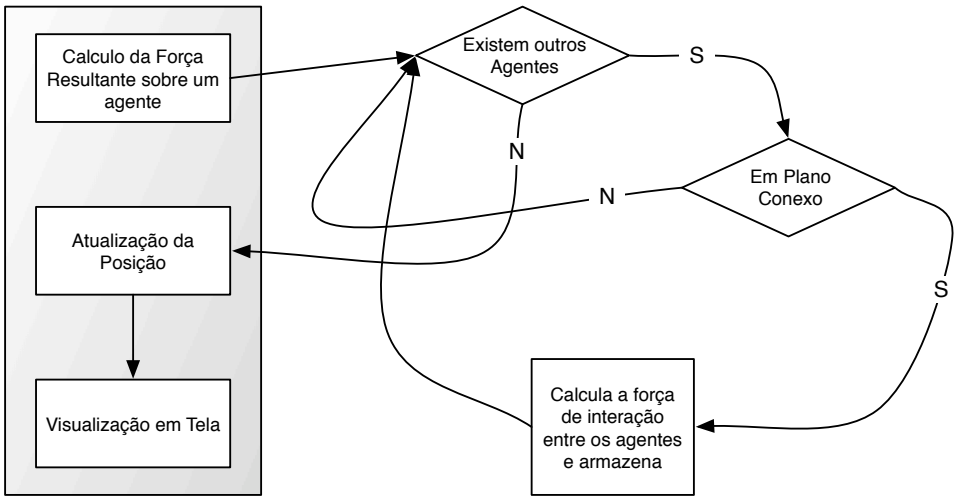

FIGURA 5.2 – Fluxo de execução de um *timestep* de simulação, para apenas um agente. O *timestep* de um agente caracteríza-se por três etapas, conforme discutido no modelo de Boids de Reynolds [Reynolds, 1987] (Capitulo-2): *i*) Cálculo da força atuante sobre o agente no dado instante de tempo, *ii*) Atualização da posição e *iii*) visualização da nova posição.

O fluxo de execução das interações entre os agentes é apresentado na Figura 5.2. A figura descreve a evolução, em um único passo de integração numérica, responsável por descrever a mudança na posição de um agente i do instante de tempo t para $t + \delta t$.

A primeira etapa do fluxo em um *timestep* para um agente i é feita verificando a existência de outros agentes no ambiente[1]. Tratando-se de um modelo baseado em Física, em que parte das forças sobre um agente i é calculada através da interação com os $n-1$ demais agentes (onde n é o número total de agentes), é fácil perceber que na inexistência de outros agentes no ambiente físico a força resultante sobre o agente é, exclusivamente, feita pelas paredes do ambiente, pela força de direcionabilidade e, eventualmente, pelas forças que atuam sobre o agente quando este está sobre uma rampa. A Equação 5.1 ilustra a força resultante sobre o agente i livre de interação dos demais agentes, com a anulação do termo social descrita por Helbing (Seção 2.2.1) na Equação 2.3.

[1] É importante perceber que 'ambiente' contempla toda e qualquer parte da região física descrita para a simulação.

$$m_i \frac{d}{dt}\vec{v}_i = m_i \frac{v_i^0 \hat{e}_i^0 - \vec{v}_i(t)}{\tau_i} + \sum_a \vec{f}_i \tag{5.1}$$

onde $\sum_a \vec{f}_i$ são todas as forças do ambiente que atuam sobre o agente, sejam paredes ou componentes de forças quando este está sobre uma rampa.

Entretanto, a existência de outros agentes no ambiente abre duas possibilidades: a presença de agentes em planos conexos ou em planos desconexos. A interação entre agentes que encontram-se em planos desconexos é descartada pelo Postulado 4.2.1 apresentado no Capítulo 4 e a força resultante sobre o agente recai àquela descrita anteriormente: torna-se nula a interação social e apenas as interações com o meio e a *força de vontade* atuam. No entanto, há interação social entre agentes de planos conexos. Esta é importante, não apenas no cálculo da interação entre agentes de salas vizinhas, em um mesmo andar, como na troca de contextos de tipos distintos, como um andar e uma rampa. Este mecanismo de interação pode ser interpretado, singelamente, como uma *visão* do agente, que *percebe* os demais nos contextos conectados ao dele, evitando colisão na troca de contexto.

Em uma situação de multidão com confinamento espacial (como uma porta ou corredor), em que as distâncias entre agentes são pequenas, existe a necessidade de um *timestep* controlado, para evitar problemas numéricos de integração. Eventualmente um agente poderia sobrepor sua posição com a de um agente localizado logo adiante, se não houvesse a interação com este localizado a frente. É o que ocorre quando não é levado em consideração a interação entre contextos distintos, e a probabilidade de colisão entre agentes na região da troca de contextos torna-se alta.

$$\epsilon_{ij} = \begin{cases} 1, & \text{se } i \text{ e } j \text{ pertecem a planos conexos} \\ 0, & \text{se } i \text{ e } j \text{ não pertecem a planos conexos} \end{cases} \tag{5.2}$$

Desta forma mantém-se coerente o protótipo com o modelo proposto, em destaque o épsilon de omissão de termos descrito na Seção 4.4.2 e representado

na Equação 5.2, e, ao invés de calcular a interação de um agente com todos os demais para, então, verificar se a força deve ser computada, apenas verifica-se se contexto no qual os agentes encontra-se localizados.

5.3 Descrição do Ambiente Tridimensional

A descrição do ambiente, em uma simulação de agentes que encontram-se em situação de emergência, é fundamental para assegurar que o modelo matemático reproduz com exatidão as soluções possíveis para tal região espacial. De forma semelhante ao modelo de Helbing, em que paredes são utilizadas para confinar agentes em um contexto (salas, por exemplo), o modelo tridimensional deve ser capaz de definir contextos ainda mais abstratos para os agentes, através do conceito de paredes, andares e rampas.

Estes conceitos permitem descrever o ambiente tridimensional no protótipo, fornecendo um conjunto de objetos que podem ser utilizados de forma a permitirem a visualização da simulação em andamento, pois, tratando-se de uma simulação física em ambiente tridimensional, é importante que o protótipo apresente mecanismos de visualização para validação visual.

5.3.1 Primitivas de descrição

Com o objetivo de simular situações de pânico em ambientes complexos, foram criadas primitivas específicas para caracterização dos ambientes. Estas primitivas delimitam as dimensões do ambiente virtual, onde a simulação está contida fisicamente, e estendem a classe `Graphics3D` do Mathematica retornando objetos que serão utilizados pelas primitivas de visualização, conforme descrito anteriormente.

Cada conceito apresentado possuiu uma primitiva específica, permitindo a descrição de ambientes como o apresentado na Figura 5.3. Na figura, cada representação consiste em um estágio da construção do ambiente. Na primeira, são descritos os andares do ambiente, na segunda, as rampas e na terceira são

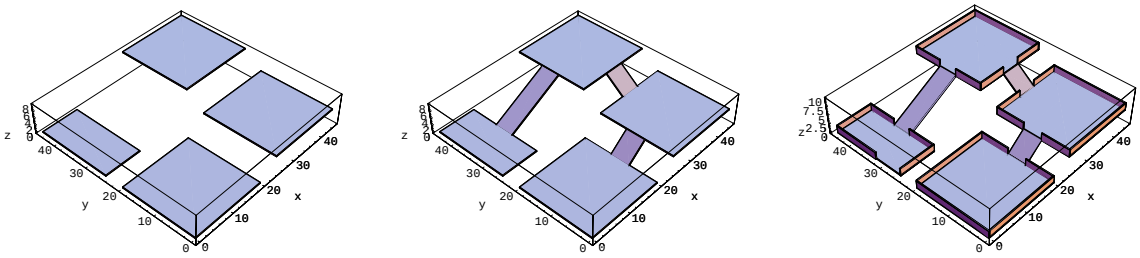

FIGURA 5.3 – Representação gráfica do ambiente, ilustrados através das primitivas de descrição. Na Figura, os conceitos de andares, rampas e paredes são demonstrados, nesta ordem.

adicionadas as paredes. Isto é feito através primitivas:

- **DrawFloor**: Permite descrever andares onde a simulação irá evoluir.
- **DrawWall**: Primitiva para descrição das paredes, que limitam o ambiente a ser simulado em um determinado andar.
- **DrawStair**: Descrição das rampas do ambiente simulado. Conforme descrito no Capítulo 4, estas são responsáveis por unirem dois andares, distintos.

A implementação das primitivas foi realizada de acordo com a especificação de *Packages* do Mathematica. Desta forma, a primeira etapa do fluxo de execução traduz informações sobre o ambiente em objetos gráficos passíveis de serem visualizados, e traduzem para o simulador os limites do ambiente. Abaixo a especificação das primitivas no formato da documentação do Mathematica:

```
In[1]:= Get["DrawWorld3D.m"];

In[2]:= ?DrawFloor
      DrawFloor[p_, d_] retorna um objeto da classe Graphics3D com origem
        em p e dimensões d (p e d são listas de três elementos: x, y e z,
        respectivamente) que representa o solo/andar/patamar do ambiente
        tridimensional.

In[3]:= ?DrawWall
      DrawWall[p_, d_, t_] retorna um objeto da classe Graphics3D com origem
```

```
        em p, dimensões d e ângulo de rotação t, que representam as paredes do
        ambiente tridimensional. p recebe um valor x, y e z. d recebe as
        dimensões de comprimento e altura da parede (x,y) e t um valor real
        que representa o ângulo (Por exemplo, -Pi/2).

In[4]:= ?DrawStair
      DrawStair[p_, a_, l_, t_] retorna um objeto da classe Graphics3D com
        origem em p, altura relativa ao patamar ao qual está ligado a,
        comprimento l e ângulo de rotação t. p recebe um valor {x, y, z}.
        a recebe valor positivo ou negativo (em relação ao andar/patamar), l recebe
        um valor relativo a projeção do comprimento da rampa/escada e t um valor
        real que representa o ângulo de rotação (Por exemplo, -Pi/2).
```

5.3.2 Descrevendo um Ambiente

Através das primitivas de discutidas na Seção 5.3.1, é possível especificar ambientes tridimensionais para o simulador. A descrição do ambiente completo é dada por uma lista contendo todos os itens individuais que descrevem o ambiente. Por exemplo,

```
floor01 = DrawFloor[{0, 0, 0}, {20, 15}];
floor02 = DrawFloor[{0, 0, 4}, {5, 5}];
floor03 = DrawFloor[{0, 10, 8}, {5, 5}];

wall01 = DrawWall[{0, 0, 0}, {20, 2}, 0];
wall02 = DrawWall[{0, 0, 0}, {15, 2}, -Pi/2];
wall03 = DrawWall[{0, 15, 0}, {20, 2}, 0];
wall04 = DrawWall[{0, 0, 4}, {5, 2}, -Pi/2];
wall05 = DrawWall[{0, 0, 4}, {5, 2}, 0];
wall06 = DrawWall[{0, 10, 8}, {5, 2}, -Pi/2];
wall07 = DrawWall[{5, 10, 8}, {5, 2}, -Pi/2];
wall08 = DrawWall[{0, 10, 8}, {5, 2}, 0];
wall09 = DrawWall[{20, 0, 0}, {5, 2}, -Pi/2];
wall10 = DrawWall[{20, 10, 0}, {5, 2}, -Pi/2];
```

```
stair01 = DrawStair[{5, 0, 4}, -4, 5, 0];
stair02 = DrawStair[{5, 5, 4}, 4, 5, -Pi/2];

ambiente = {wall10, wall09, wall07, wall06, wall05, wall04, wall03, wall02,
            wall01, floor01, floor02, floor03, stair01, stair02};
```

Todas as dimensões utilizadas para descrição dos ambientes na implementação estão de acordo com o Sistema Internacional de medidas (SI) [Taylor, 1995, Taylor, 2001], isto é, a unidades utilizada pelas primitivas para descrever dimensões espaciais é dada em *metros*.

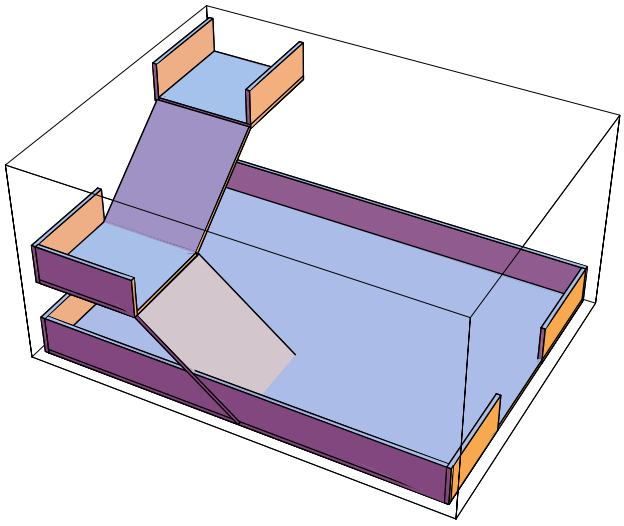

FIGURA 5.4 – Representação de um ambiente tridimensional descrito pelas primitivas de descrição.

Uma vez descritos os elementos que compõem o ambiente, estes são adicionados a uma lista (`ambiente`, no exemplo acima) e podem ser visualizados pela primitiva `Show[ambiente]`. A Figura 5.4 ilustra o ambiente descrito nesta seção.

5.4 Seleção de Contextos

Durante a execução da simulação, cada agente apresenta um identificador do contexto em que se encontra. Associar um contexto a um agente implica em definir para este um objetivo físico real, ou seja, uma posição espacial para a qual deverá se deslocar. Para esta posição é dado nome de *atrator*, e cada contexto possui ao

menos um destes pontos de convergência. A alteração de contexto por parte do agente está associada ao momento em que este atinge seu objetivo, isto é, chega até o atrator do contexto em que se encontra.

O mecanismo de troca de contexto é implementado a partir de uma *árvore de contextos* que atribui ao nó mais significativo o contexto final que cada agente deve conquistar. A Figura 5.5 apresenta um ambiente virtual e uma possível árvore de contexto associada. Na figura, os agentes localizados no contexto 5 deslocar-se-ão até o atrator deste contexto, localizado na interseção com o contexto 4. De acordo com a árvore de contextos definida para os agentes, seguirão buscando novos contextos de acordo com o grafo orientado até atingirem o contexto 1.

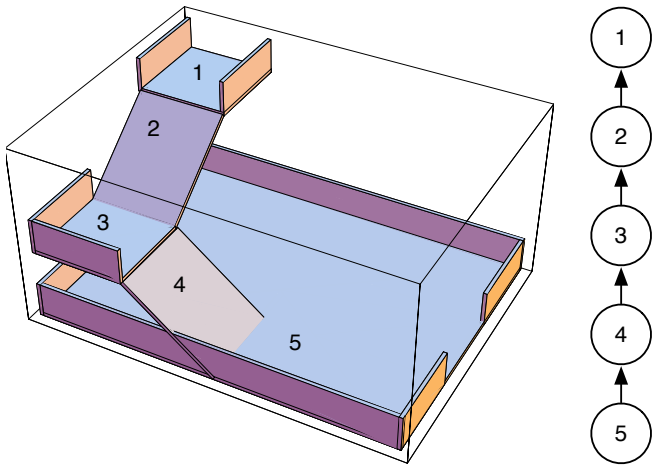

FIGURA 5.5 – Representação do ambiente da Figura 5.4 com seus contextos e a árvore de contextos associada a sua evolução.

Entretanto, a árvore de contextos não é estática e idêntica a todos os agentes, como aquela apresentada na Figura 5.5. A Figura 5.6 apresenta uma árvore de contextos para um agente localizado no plano inferior de um ambiente que lhe permite duas alternativas. Um mecanismo de seleção deve associar ao agente um identificador para seu contexto que, pelo grafo, poderia ser 8 ou 9. Neste ponto, diferentes estratégias podem ser utilizadas para a escolha; por exemplo pelo atrator *i*) mais próximo, *ii*) com menor densidade de agentes em um determinado instante de tempo ou *iii*) selecionado de forma aleatória. No protótipo optou-se pela escolha aleatória, evitando determinismo durante diferentes execuções do simulador.

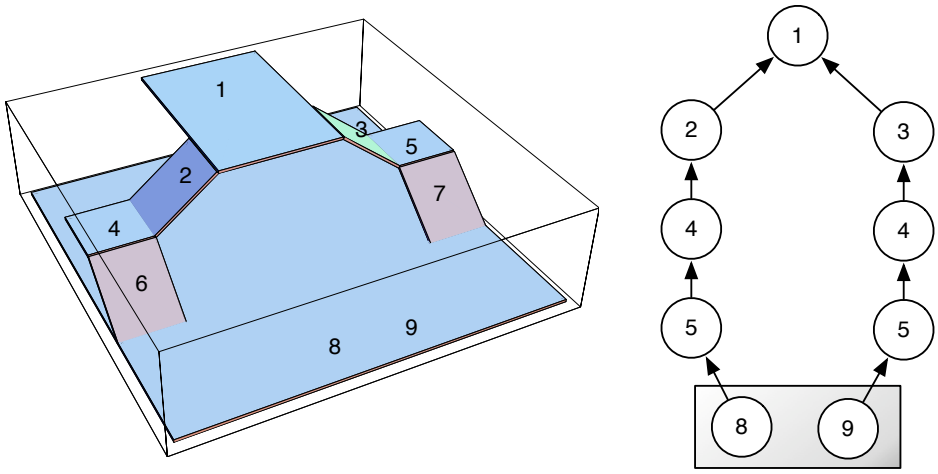

FIGURA 5.6 – Representação de um ambiente com árvore de contextos dinâmica, representada pela dualidade do contexto 8 e 9. Nesta situação, é atribuído um identificador de forma aleatória ao agente, dentre os possíveis.

5.5 Descrição de Funções do Simulador

Esta seção discute o passo de integração utilizado na integração numérica da equação diferencial para cada agente. Na seqüência, descreve-se o método utilizado para a integração numérica.

5.5.1 Avaliação do passo de integração (δt)

Equações de movimento para agentes virtuais, e demais sistemas físicos, apresentam soluções analíticas contínuas, isto é, são representados por soluções que envolvem *todos* os instantes de tempo. Entretanto, soluções computacionais exigem que este espaço seja discretizado, em pequenas frações de tempo, para poder evoluir. A estes intervalos de tempo é dado o nome de *timestep*, ou δt, da simulação.

Simulações executadas no Mathematica apresentam, na maioria das vezes, um desempenho inferior àqueles oferecidos por linguagens compiladas, como C ou C++. Portanto, foi realizada uma análise para reduzir o tempo de processamento da simulação, através da redução do passo de integração numérica (*timestep*). A avaliação foi feita sobre uma característica do problema em questão: a convergência assintótica da energia cinética para um valor estacionário.

Durante o intervalo da simulação, em uma região plana e livre de confinamento

espacial, por exemplo em um andar, os agentes buscam estabilizar suas velocidades de acordo com a velocidade desejada, definida pelo módulo v^0 (Eq. 2.3). Desta forma, o valor da energia cinética total do sistema busca aproximar-se de um valor limite, estabilizando neste valor. A energia cinética total do sistema no instante t, dada por K_t, é a soma da energia cinética de cada agente neste instante de tempo, e expressada pela Equação 5.3.

$$K_t = \frac{1}{2}\sum_{i=1}^{n} m_i v_i(t)^2 \tag{5.3}$$

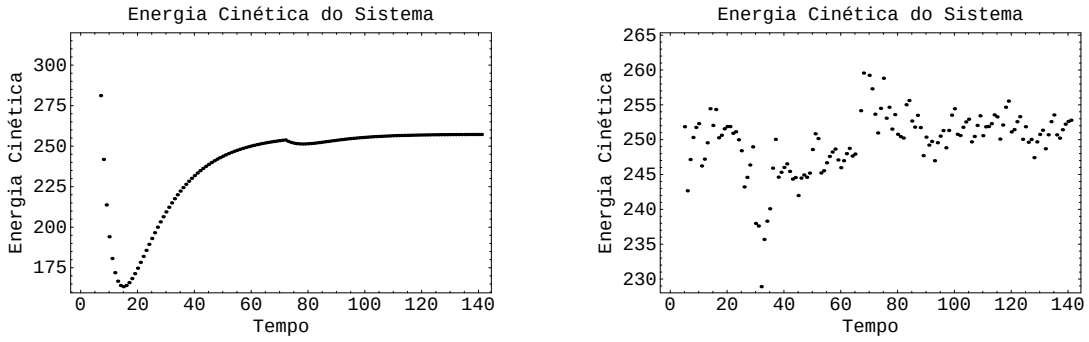

FIGURA 5.7 – Gráficos da energia cinética obtidos com *timesteps* diferentes. O primeiro gráfico apresenta um $\delta t = 0.05$ e o segundo $\delta t = 0.1$.

A Figura 5.7 apresenta as curvas das energias cinéticas para dois diferentes valores de δt. A primeira, com $\delta t = 0.05$, apresenta o comportamento citado, de estabilização. Na segunda, com $\delta t = 0.1$, observa-se um nível de desordem dado pelo erro numérico das colisões entre os agentes. A sobreposição entre dois agentes, isto é, a inter-penetração espacial, resulta em uma aceleração de grandes proporções, lançando os agentes para posições distantes, sob altas velocidades. Assim, estes agentes passam a apresentar energias cinéticas que não correspondem a realidade de um ambiente contínuo.

Entretanto, a análise deve ser feita em regiões planas, pois no momento em que os agentes trocam de contexto, para uma rampa por exemplo, sua velocidade em módulo varia causando a interpretação de uma falsa colisão.

É importante ressaltar o comportamento em forma de *vale*, ou depressão, apresentado no gráfico esquerdo da Figura 5.7, no intervalo de tempo $0 < t < 40$.

No momento inicial da execução da simulação os agentes são inicializados com velocidades aleatórias (em módulo e sentido). Após, a força atuante sobre cada agente apresenta uma característica restauradora, direcionando-os para o atrator do contexto em que se encontram. Neste instante, muitos agentes passam a apresentar velocidades, em módulo, próximas a zero, por estarem mudando o sentido de movimento. Após este instante suas velocidades passam a crescer, dentro dos limites impostos pelo modelo.

A utilização de *timesteps* menores garante a veracidade dos resultados, porém compromete o tempo de processamento. A análise feita sobre a energia cinética permite encontrar um valor para balancear estas duas quantidades.

5.5.2 Integração numérica

O protótipo utiliza a implementação do algoritmo de Verlet [Rapaport, 2004], oferecida pelo Mathematica, para a evolução das equações diferenciais. Este algoritmo apresenta ótimos resultados em tempo computacional contra erro numérico, permitindo o uso de *timesteps* relativamente maiores ao método de Newton. Para um passo temporal igual a δt, o erro introduzido por este algoritmo, segundo Rapaport, é de $\mathcal{O}(\delta t^4)$ para a posição e $\mathcal{O}(\delta t^2)$ para a velocidade.

5.6 Saída de Resultados

Para avaliar a qualidade da simulação, foram desenvolvidos diferentes módulos para saída dos resultados gerados pelo simulador: *i*) saída padrão do Mathematica, uma vez que as primitivas de descrição do ambiente tridimensional utilizam a classe `Graphic3D` do Mathematica, conforme descrito anteriormente, *ii*) saída OpenGL através de um *plugin* externo ao Mathematica [Kuska, 2007] e *iii*) saída em arquivo `XML`, para o reprodutor *Player2*, desenvolvido no Projeto CSHuV.

Capítulo 6

Resultados Obtidos

Este capítulo apresenta os resultados obtidos a partir deste trabalho. A avaliação dos resultados de um sistema tão complexo como este, que busca representar interações entre humanos reais, não é tarefa fácil, e poderia, por si só, ser interpretada como uma atividade de pesquisa, envolvendo profissionais das mais diversas áreas. Desta forma, este capítulo não se propõe a validação de resultados, e sim a discutir resultados obtidos a partir da implementação sobre o modelo tridimensional apresentado para simulações em ambientes complexos – capítulos 5 e 4, respectivamente –, demonstrando a potencialidade do modelo desenvolvido.

A Seção 6.1 apresenta os resultados obtidos com o protótipo implementado com diferentes conjuntos de dados. Na seqüência, a Seção 6.2 finaliza o capítulo discutindo os resultados obtidos sobre a implementação concorrente para o modelo bidimensional, já apresentados em forma de artigos [Cordeiro et al., 2005, Cordeiro et al., 2006, Cavalheiro et al., 2006].

6.1 Modelo Tridimensional

Como descrito no Capítulo 4, o modelo matemático apresenta inúmeros parâmetros que devem ser calibrados para obter resultados que descrevam multidões reais. Esta seção apresenta os resultados obtidos com o protótipo tridimensional, apresentado no Capítulo 5, através da variação destes parâmetros para medir o

impacto sobre um conjunto de configurações. As próximas seções avaliam o impacto da atuação da força peso sobre um agente, da variação dos parâmetros de atrito dinâmico e do coeficientes para a resistência do ar. Na seqüência é apresentado o resultado em conjunto destas forças atuando sobre um agente. A seção finaliza apresentando alguns resultados da visualização de humanos virtuais articulados.

6.1.1 Atuação da força peso

Todo agente sobre uma rampa é capaz de perceber uma aceleração sobre si, resultante das componentes da força peso no plano em que se encontra. Desta forma, foi avaliado o impacto da força peso sobre o agente, com o intuito de verificar se o modelo se encontrava de acordo com o observável no mundo real. Foi utilizado o ambiente tridimensional representado na Figura 6.8, com um agente, para verificar apenas a atuação da força peso, descartando a interação entre agentes e demais forças atuantes.

A Figura 6.1 apresenta o gráfico para tal ambiente[1]. Em um instante inicial da simulação o agente encontra-se em condição aleatória de velocidade (módulo e diração). No momento em que a força de restauração (dada pelo termo de direcionabilidade na equação do movimento) atua, atinge-se o primeiro pico de energia cinética. A primeira depressão no gráfico representa a primeira rampa em que o agente sobe. No primeiro andar, o agente volta a conquistar a velocidade desejada imposta pelo modelo, até encontrar uma segunda rampa. Na intervalo entre $650 < t < 780$ o agente encontra-se descendo uma rampa inclinada e, nesta situação, suas componentes da força peso favorecem o ganho de velocidade.

Foram realizados outros testes para determinar o impacto da força peso no modelo, e todos resultaram em gráficos com características semelhantes. A Figura 6.2 apresenta duas novas aquisições de dados, com agentes de diferentes massas. No gráfico à esquerda é possível perceber que o agente encontrava-se localizado próximo ao atrator de seu contexto, e em poucos passos de integração passou ao

[1]Percebe-se claramente as mudandas de contextos no gráfico, pois são representadas pelas descontinuidades.

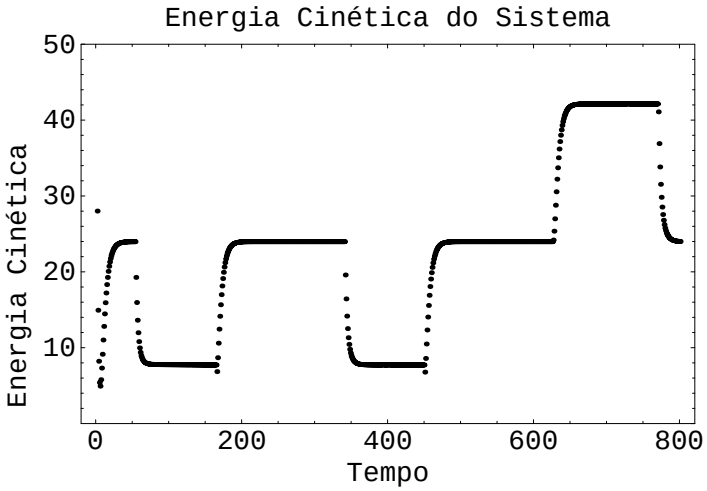

FIGURA 6.1 – Gráfico representando a energia cinética de um agente durante sua trajetória no mundo virtual da Figura 6.8, com atuação da força peso.

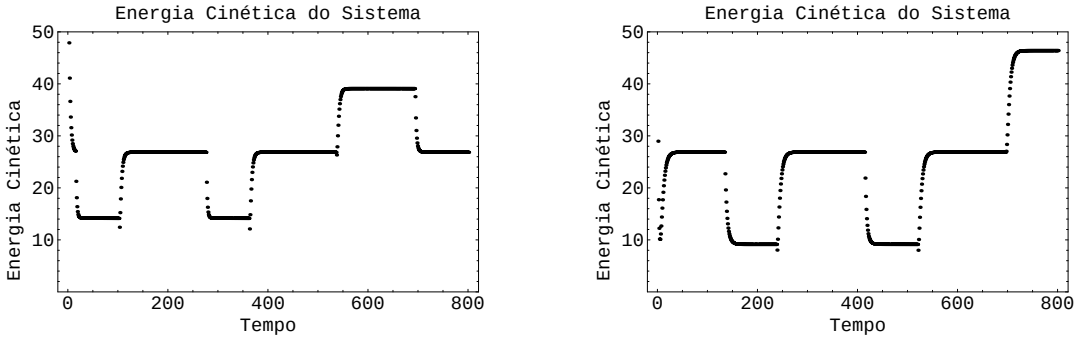

FIGURA 6.2 – Gráfico da energia cinética para agentes de diferentes massas durante trajetória no mundo virtual com atuação da força peso nas rampas.

contexto da rampa. No gráfico à direita, percebe-se o mesmo comportamento dos demais, entretanto, neste, o efeito da força peso apresentou maior impacto no valor da energia cinética. Isto se deve ao fato do agente possuir uma massa maior que a apresentada pelos agentes dos exemplos anteriores.

6.1.2 Diferentes parâmetros de atrito cinético

Foram feitos testes para simular o impacto da força de atrito cinético sobre um agente em uma rampa. É importante ressaltar que o atrito cinético é dado por um coeficiente que multiplica a reação do agente à superfície (vetor Normal).

Os resultados foram obtidos para os coeficientes $\mu = 0, 0.00005, 0.005, 0.01$ e 0.05 e estão representados nas Figuras 6.3, 6.4 e 6.5.

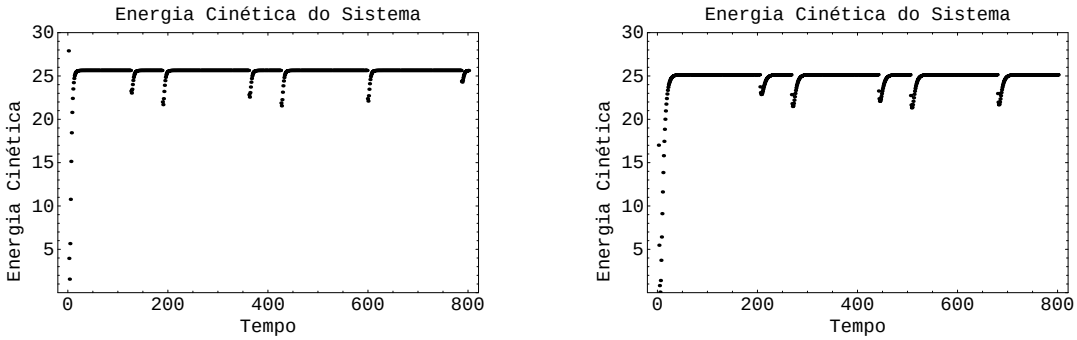

FIGURA 6.3 – Gráficos para a energia cinética com impacto da força de atrito cinético de coeficientes $\mu = 0$ e $\mu = 0.00005$

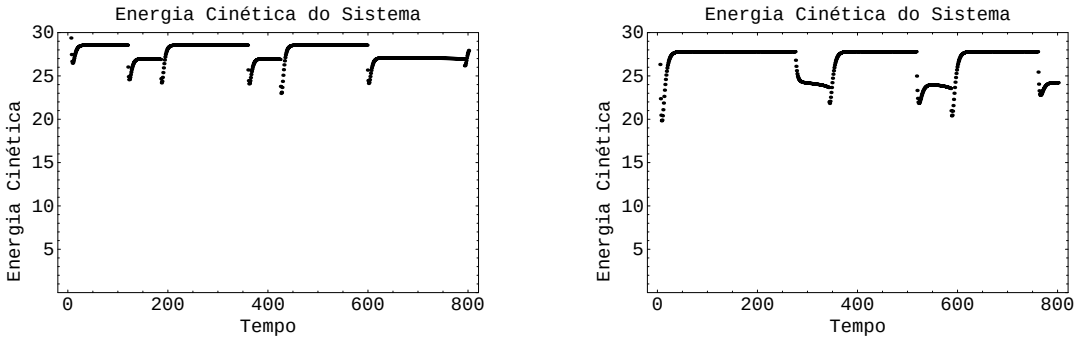

FIGURA 6.4 – Gráficos para a energia cinética com impacto da força de atrito cinético de coeficientes $\mu = 0.005$ e $\mu = 0.01$

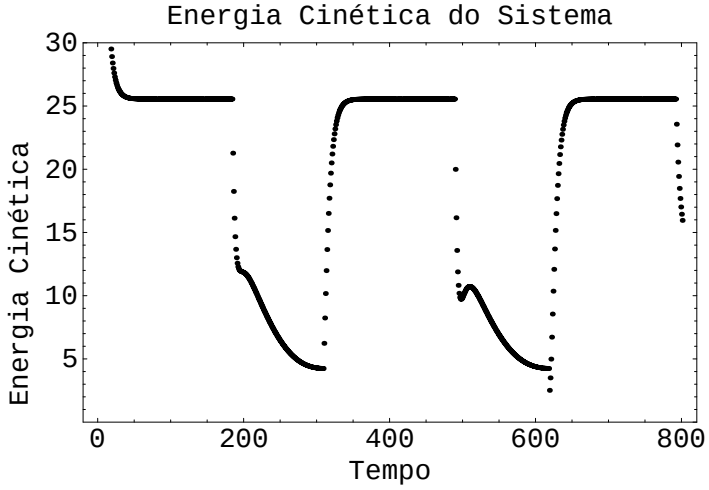

FIGURA 6.5 – Gráfico para a energia cinética com impacto da força de atrito cinético de coeficiente $\mu = 0.05$

Em todas as situações o efeito da força de atrito mostrou-se de acordo com o modelo, reduzindo a velocidade do agente em cada uma das rampas. É importante reparar que houve redução na velocidade do agente mesmo na última rampa, que pela classificação dos contextos é uma descida. Isto se deve a inexistência da força

peso nestes testes.

6.1.3 Diferentes parâmetros de resistência do ar

Semelhante ao atrito cinético, a força de resistência do ar expressa uma ação contrária ao movimento do agente. Portanto, semelhantemente aos resultados apresentados na seção anterior, estes devem apresentar redução na velocidade do agente mesmo em uma rampa utilizada para descida dos agentes. Entretanto, a força de resistência do ar é uma força diretamente proporcional à velocidade, levando a solução à um valor limite, estabilizando a velocidade do agente. Estas características podem ser observadas na Figura 6.6.

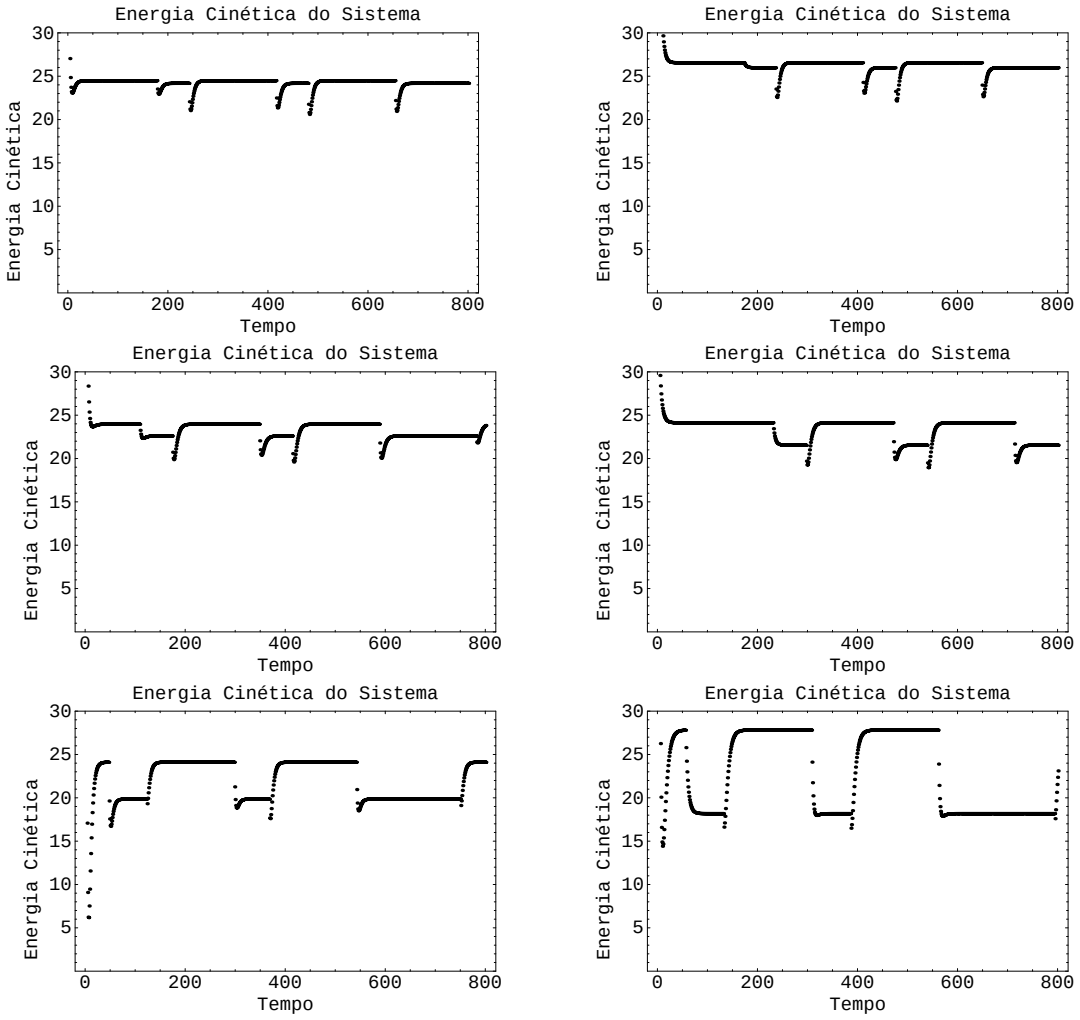

FIGURA 6.6 – Gráficos para a energia cinética com impacto da resistência do Ar com parâmetros $\gamma = 1, 2, 4, 8, 12$ e 32.

6.1.4 Impacto combinado

Para obter um maior grau de realismo no movimento dos agentes, o modelo descrito no Capítulo 4 apresenta uma solução composta por todas as forças descritas acima.

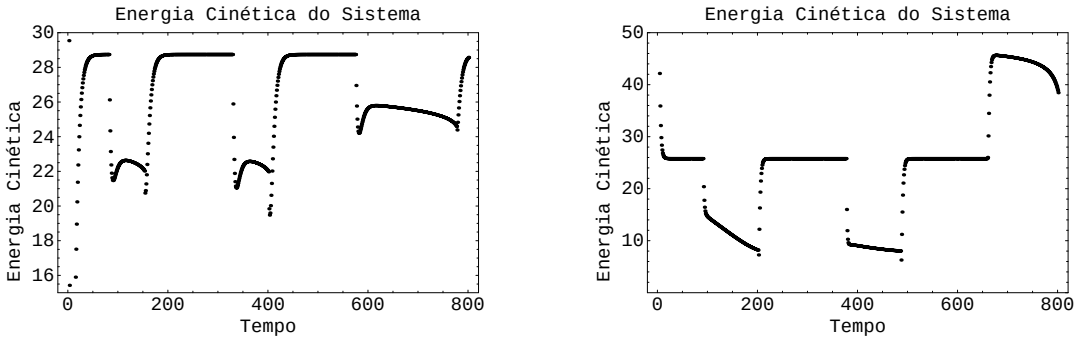

FIGURA 6.7 – Gráficos para a energia cinética com efeito combinado de forças atuantes.

A Figura 6.7 apresenta os resultados para forças combinadas atuando sobre um agente. No gráfico à esquerda foi realizado um teste apenas com as forças de oposição ao movimento (atrito e resistência do ar), com valores $\mu = 4$ e $\gamma = 0.01$. O gráfico à direita apresenta o resultado final, com o acréscimo da força peso atuando sobre o agente.

Neste último gráfico, o agente perde velocidade enquanto sobe a rampa. O efeito é justificado pela atuação da força peso e do atrito dinâmico. Entretanto, na rampa onde o agente desce, a força peso atua de força positiva ao movimento, porém a força de resistência do ar, contrária ao movimento e proporcional a velocidade do agente, é capaz de desacelerar seu movimento.

6.1.5 Resultados Visuais

Através da visualização dos resultados na forma de animação de agentes articulados é possível, não apenas verificar a coerência da solução, como prover resultados que ilustram a potencialidade do modelo desenvolvido e sua aplicabilidade. Desta forma, foram criados ambientes através da ferramenta SketchUp [Google, 2007] e exportados para o formato .osg (OpenSceneGraph)

[Burns and Osfield, 2004], e importados diretamente no reprodutor *Player2*, desenvolvido no CSHuV. A Figura 6.8 apresenta o ambiente gerado na ferramenta, com e sem a aplicações de texturas nos elementos.

Como resultado prático de agentes articulados, a Figura 6.9 apresenta diferentes ângulos de câmera de uma mesma seqüência. São apresentados agentes em fuga para uma região mais alta do ambiente. O resultado da interação entre os agentes pode ser visto na Figura 6.10, na qual os agentes buscam o andar superior do ambiente, porém respeitando o espaço inter-pessoal através das forças sociais.

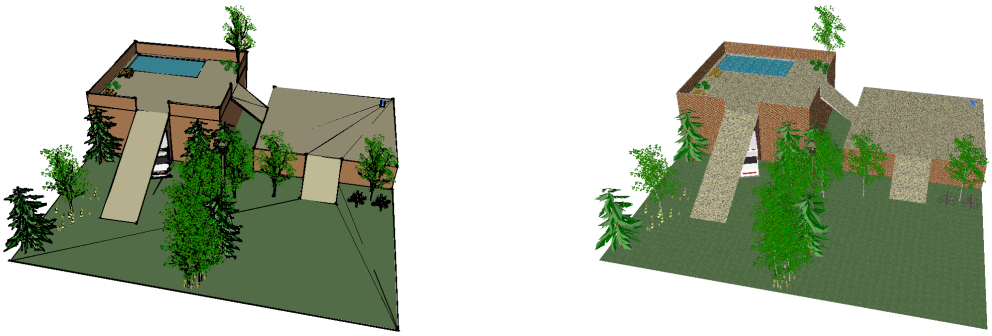

FIGURA 6.8 – Ambientes gerados para simulação com auxílio da ferramenta SketchUp [Google, 2007]

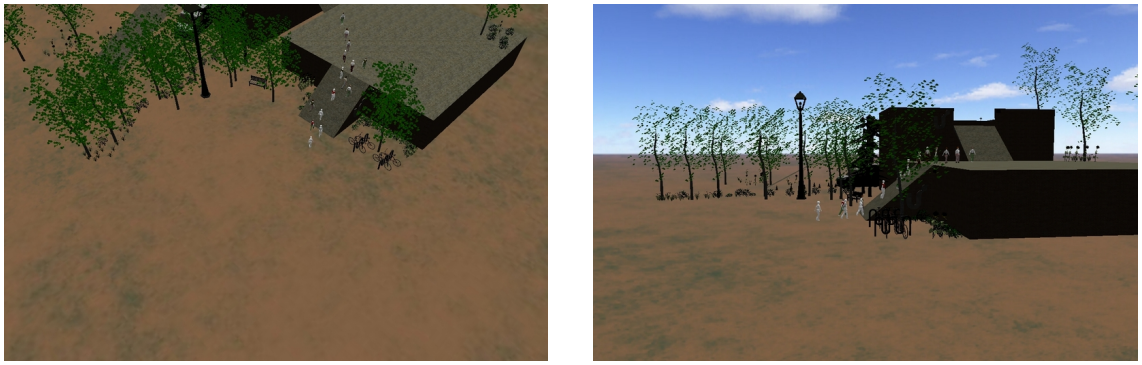

FIGURA 6.9 – Visualização de uma simulação de articulados com o uso do Player2.

6.2 Modelagem da Concorrência

A etapa de modelagem da concorrência visa identificar para uma determinada aplicação sua estrutura em implementação concorrente. Entre os diversos aspectos

FIGURA 6.10 – Interação social entre agentes articulados em uma rampa.

que devem ser considerados, citamos a identificação das tarefas concorrentes e a estrutura de sincronização e de comunicações.

6.2.1 Identificação do custo computacional

O modelo de simulação bidimensional apresentado neste trabalho possui duas fontes principais de custo computacional, o número de agentes simulados e o erro aceitável para os resultados. O número de agentes simulados reflete uma entrada do problema, relacionada ao número de indivíduos evoluindo em uma dada realidade modelada. A segunda fonte de carga modelada está relacionada com a precisão dos resultados oferecidos pela simulação. Esta precisão está refletida na granularidade da simulação. Por granularidade entende-se a freqüência em que as interações entre os agentes são realizadas. Desta forma, quanto menor o *timestep* utilizado para avanço do tempo da simulação, maior é a quantidade de cálculos necessária.

Considerando o número de agentes n, e m obstáculos, a complexidade da computação em cada *timestep* é $\mathcal{O}(n^2 + n \times m)$. De forma a reduzir o número de cálculos em cada *timestep*, buscamos identificar o espaço físico onde as forças são relevantes, pois a magnitude das forças decresce com a distância. Denominamos estes espaços físicos de *células*.

6.2.2 Identificação da unidade de trabalho

A concorrência do modelo de simulação de multidões apresentada se encontra no paralelismo de dados da aplicação: o cálculo de forças pode ser realizado sobre cada agente de forma independente a cada *timestep* transcorrido da simulação. No entanto, o custo de cálculos de um novo estado para um agente a cada *timestep* não justifica a concorrência neste nível. Também observamos que uma simples decomposição espacial em regiões independentes (domínios) não pode ser explorada no nosso modelo de simulação, pois há uma relação restrita de forças entre agentes de forma que o estado do sistema deve ser comunicado entre as regiões a cada *timestep* simulado. O conceito de células, definido anteriormente, é estendido para implementar o programa concorrente.

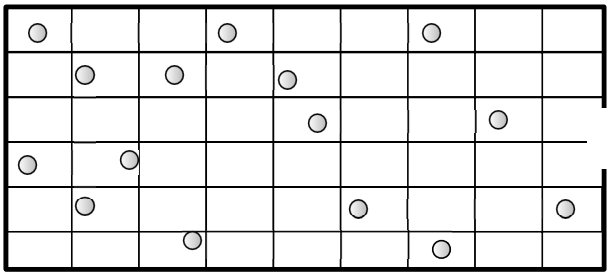

FIGURA 6.11 – Espaço simulado dividido em células.

A solução adotada divide o espaço físico em um número fixo de células. A Figura 6.11 apresenta a divisão em termos de células para uma determinada área simulada. Nesta figura, os agentes são representados por círculos, a divisão em células é representada por linhas finas enquanto as paredes por linhas grossas. É importante notar que nesta representação, a área de influência de um agente abrange no mínimo uma célula, podendo, em caso extremo, se estender por até quatro células.

Nesta implementação, o cálculo da simulação é aplicado de forma concorrente sobre todas as células. Para obter tal nível de concorrência, as informações contendo o estado da simulação são duplicadas: cada *timestep* t_i tem como "entrada" o estado S_{i-1} e produz como "saída" o estado S_i. S_{i-1} representa o estado da simulação no instante de tempo t_{i-1} e S_i o novo estado computado.

A unidade de trabalho (tarefa) é, portanto, definida em termos de células. É importante notar que estas tarefas possuem diferentes cargas computacionais (cargas irregulares), uma vez que no interior de cada célula podem existir um número diferente de agentes.

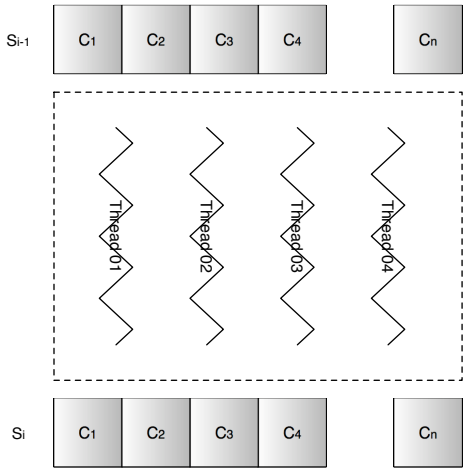

FIGURA 6.12 – Diagrama expressando a concorrência intra nodos.

6.2.3 Escalonamento

Definida a unidade de trabalho para a execução concorrente, procura-se a definição de uma estratégia de execução eficiente da simulação em aglomerado de computadores. Neste tipo de arquitetura, um conjunto de nós (mono, dual ou quad processados) encontram-se ligados a uma rede de comunicações (normalmente de alto desempenho) e dedicados à execução de uma única aplicação. Portanto, o escalonamento deve ser realizado considerando os dois níveis de concorrência que a arquitetura proporciona: intra e entre nós.

O escalonamento intra-nó é realizado explorando *multiprogramação leve*, como apresentado na Figura 6.12. O conjunto S_{i-1} de células é armazenado em uma lista, a qual é consumida por um conjunto de *threads* dedicadas à execução do cálculo sobre cada célula. Desta forma, a descrição da concorrência da aplicação, que é dado em termos de cálculo de célula, pode ser dissociada do paralelismo real que pode ser obtido efetivamente no nó, que é dado em termos de *threads* de trabalho (na Figura 6.12 são representadas quatro *threads*).

O nível de escalonamento intra-nó leva em consideração a distribuição de carga computacional entre os nós. A distribuição inicial do trabalho é realizada de forma estática: o conjunto de c células que compõem o sistema simulado é decomposto em p domínios, onde p é o número de nós do aglomerado. Inicialmente, cada nó recebe c/p células.

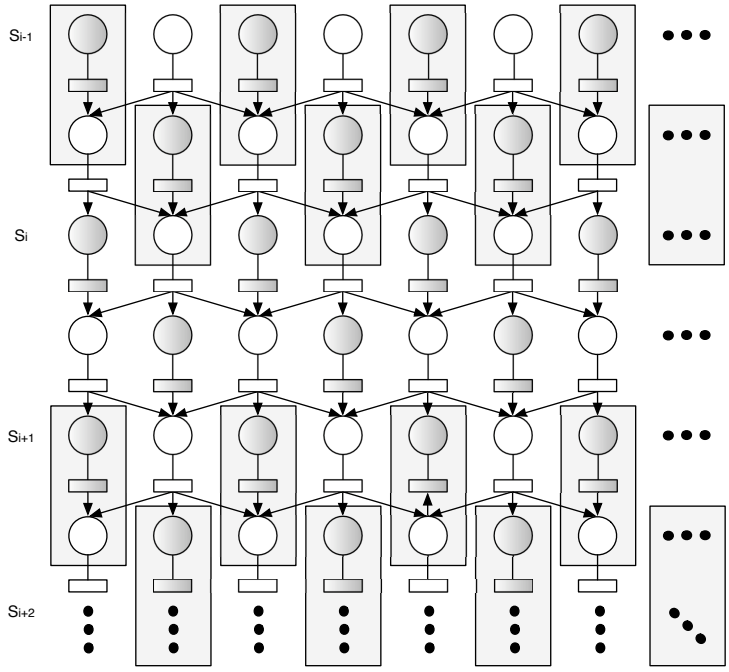

FIGURA 6.13 – Diagrama do ambiente de execução distribuído, explorando a concorrência entre nodos.

A execução é realizada conforme apresentado na Figura 6.13. Nesta figura, o cálculo de forças em cada domínio é representado por círculos. Cada coluna vertical corresponde à seqüência de operações realizadas por cada um dos nós do aglomerado. Esta figura apresenta os pontos de sincronização entre os cálculos das forças entre agentes situados em regiões de borda de cada domínio: círculos cheios representam o cálculo das forças nas células pertencentes às regiões centrais do domínio, enquanto círculos vazados representam o cálculo das forças nos agentes situados em células pertencentes às bordas do domínio. De forma análoga, retângulos cheios correspondem aos novos estados obtidos para as células centrais do domínio e retângulos vazados correspondem aos novos estados das células em regiões de borda do domínio. Deve-se notar que o cálculo do estado S_i das células

situadas nas posições de borda necessitam que o estado S_{i-1} tenha sido computado. Para facilitar a visualização, são apresentadas as sincronizações de cada nó com dois vizinhos, quando na realidade as sincronizações se dão entre quatro nós. Esta estratégia permite sobrepor os tempos associados ao sobrecusto das comunicações das células de borda com o cálculo efetivo das células centrais do domínio, reduzindo o tempo de ociosidade de processamento.

6.2.4 Balanceamento de carga

O balanceamento de carga é aplicado na execução distribuída da simulação em um aglomerado de computadores. A estratégia de deslocar a computação das forças atuantes sobre as células centrais e limítrofes dos domínios, somada à estratégia de execução *multithreading*, permite sobrepor os sobrecustos das comunicações entre os nós [Valiant, 1990]. Como apresentado anteriormente, a distribuição inicial da carga computacional se dá pela simples divisão das células entre os nós de processamento. Durante a execução, o balanceamento de carga se dá de forma dinâmica. A unidade de carga utilizada como base para o balanceamento é o número de agentes em cada nó. Assim, considerando que uma dada simulação é realizada sobre n agentes, o objetivo do balanceamento de carga é manter todos os nós com $n/p+\delta$ (δ corresponde a uma variação aceitável sobre a média ótima).

Embora estratégias de balanceamento de carga [Willebeek-LeMair and Reeves, 1993] apresentem grandes sobrecustos devido às necessidades de controle global de carga, esta se apresenta como uma opção interessante para o modelo de execução da simulação. As trocas de informações de carga computacional, no algoritmo implementado, são realizadas apenas entre nós vizinhos, sendo que a equalização da carga ocorre sempre entre dois nós a cada momento. Caso seja verificado o desbalanceamento de carga entre os nós p_k e p_j durante a troca de estados S_i, na troca de estados em S_{i+1} o nó sobrecarregado descarta sua região da borda, reduzindo seu domínio, e o menos carregado tem seu domínio estendido. Esta estratégia é particularmente interessante em situações

de evacuação, onde agentes tendem a dirigir-se a um mesmo destino: a saída do ambiente. Este comportamento gera desbalanceamento de carga entre os nós, sendo compensado pela redistribuição das células.

Capítulo 7

Conclusão

Esta trabalho apresentou um modelo matemático que possibilita a expansão do modelo de Helbing de forças sociais bidimensionais para ambientes tridimensionais. Esta expansão é desejada por permitir simulações de ambientes mais complexos em relação aqueles simulados pelos atuais modelos e implementações.

Neste capítulo serão abordados os resultados obtidos com o protótipo e restrições encontradas durante este trabalho. Na seqüência são descritos trabalhos futuros que podem ser realizados com foco no modelo aqui apresentado. Por último são apresentadas as contribuições desta dissertação de mestrado.

7.1 Análise de resultados

No Capítulo 6 foram apresentados resultados obtidos através do protótipo desenvolvido. Através de inúmeros testes realizados com o protótipo, com diferentes parâmetros e configurações, foi observado que o modelo se comportou como o previsto e aceitável para este tipo de simulação. A Figura 6.7 exemplifica o fato através da adição de todas as forças atuando sobre o agente no mesmo instante. Como esperado, o agente perdeu velocidade enquanto subia a rampa. Efeito este justificável pela atuação da força peso e do atrito dinâmico. Durante a descida da rampa, a força peso atuou de força positiva ao movimento, porém a força de resistência do ar, contrária ao movimento e proporcional a velocidade do agente, foi

capaz de desacelerar seu movimento.

7.2 Restrições

Embora a ferramenta Mathematica apresente um ótimo desempenho computacional, não se pode comparar sua performance com uma linguagem de programação compilada. Desta forma, o protótipo implementado apresenta restrições em relação ao tempo de execução, não sendo capaz, por exemplo, de evoluir uma simulação em tempo real.

Foram encontradas barreiras tecnológicas durante a aquisição de dados. Eventualmente máquinas com maior poder computacional e com tecnologia Intel Core-Duo, possam realizar estas simulações em um tempo bastante inferior aos apresentados, pois o Mathematica explora a concorrência de seus algoritmos na nestas arquitetura multiprocessada.

Simulações realizadas com o protótipo se mostraram livre de colisões mesmo em situações com 100 agentes confinados em uma região de 20×20 metros com atrator em um único ponto. Entretanto simulações com maior número de agentes retornam ao problema da barreira tecnológica imposta e tais simulações passam a apresentar um grande tempo e, eventualmente, há a necessidade de diminuir o *timestep* da simulação.

Como no modelo de Helbing, percebeu-se uma grande dependência do tamanho do *timestep* escolhido para a integração numérica. Foi realizado um estudo para um valor ótimo dentro das simulações realizadas, porém, este valor encontra-se muito inferior a valores mensuráveis em situações reais.

Além da dependência do *timestep*, o modelo de Helbing é extremamente dependente dos parâmetros utilizados para a simulação (Tabela 2.1). Pequenas variações não resultam em movimentos realísticos.

Encontrar valores corretos para os parâmetros não é simples. O modelo apresentado é suficientemente genérico e apresenta uma série de parâmetros que devem ser calibrados através de uma validação do modelo. Esta validação não

foi realizada, porém o estudo realizado demonstrou as potencialidades do modelo através de gráficos que representam o comportamento esperado para humanos reais.

Entretanto, para a utilização real deste protótipo como ferramenta de segurança e desenvolvimento de edificações, é importante que tal validação seja realizada, na busca de encontrar os parâmetros corretos para o conjunto de equações.

7.3 Trabalhos Futuros

Esta pesquisa abre uma série de trabalhos que podem ser realizados a partir dos resultados aqui apresentados:

- **Desempenho**: Implementar o modelo em uma linguagem de programação que possa ser compilada para obter maior desempenho. É possível explorar técnicas de processamento de alto desempenho nesta etapa, porém é necessário um estudo do custo computacional associado a este modelo e de técnicas para o escalonamento das tarefas geradas.

- **Validação e calibração de dados**: Para que o simulador possa representar humanos reais em situações de pânico, é importante a calibração de dados e validação destes através de técnicas computacionais, como a Visão Computacional. O uso de câmeras de vídeo em locais estratégicos pode fornecer dados referentes ao fluxo de pedestres, por exemplo, e requer uma linha de pesquisa apenas para este trabalho.

- **Análise comportamental**: Conforme descrito anteriormente, simulações de humanos virtuais apresentam uma série de características que fogem do escopo da computação. Para auxiliar na validação e calibração de dados é importante que seja feita uma pesquisa psicológica e sociológica sobre o comportamento de pedestres.

7.4 Contribuições

A principal contribuição deste trabalho foi o desenvolvimento de um modelo matemático parametrizável capaz de simular multidões em ambientes tridimensionais através de uma abordagem Física. A partir deste estudo, foram obtidos resultados que vão de acordo com o esperado para humanos reais em situações de evacuação. Desta forma, este trabalho pode ser uma ferramenta útil para análise de evacuação de ambientes complexos, com múltiplos andares, treinamento de agentes de segurança e até mesmo para planejamento arquitetônico.

Bibliografia

[American Nacional Standards Institute, 1994] American Nacional Standards Institute (1994). Ieee - standard for information technology: Portable operationg system interface (posix). part i: system application program interface (api) – amendment 1 – reatime extension [c language]. 1109 Spring Street, Suit 300, Silver Spring, MD 30910, USA, IEEE Computer Society Press, 1994, xii + 590p. IEEE Std 1003.1b-1993 (formely known as IEEE P10003.4; includes IEEE Std 1003.1-1990). Approved September 15 ,1993, IEEE Standards Board. Approved April 14, 1994.

[Arkin, 1987] Arkin, R. (1987). Motor schema based navigation for a mobile robot: An approach to programming by behavior. *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, 4:264–271.

[Becker et al., 2002] Becker, D. J., Sterling, T., Savarese, D., Dorband, J. E., Ranawake, U. A., and Packer, C. V. (2002). Beowulf : A Parallel Workstation for Scientific Computation. Internet : http://www.beowulf.org/papers/ICPP95/icpp95.html.

[Bouvier et al., 1997] Bouvier, E., Cohen, E., and Najman, L. (1997). From crowd simulation to airbag deployment: particle systems, a new paradigm of simulation. *Journal of Electronic Imaging*, 6:94–107.

[Braun et al., 2003] Braun, A., Bodmann, B. E. J., Oliveira, L. P. L., and Musse, S. R. (2003). Modelling individual behavior in crowd simulation. In *Proceedings of Computer Animation and Social Agents 2003*, Brunswick, N.

[Braun et al., 2005] Braun, A., Bodmann, B. J., and Musse, S. R. (2005). Simulating virtual crowds in emeergency situations. 1:244–282.

[Brogan and Hodgins, 1995] Brogan, D. C. and Hodgins, J. K. (1995). Group behaviors for systems with significant dynamics. In *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 528–534.

[Burns and Osfield, 2004] Burns, D. and Osfield, R. (2004). Open Scene Graph A: Introduction, B: Examples and Applications. *Proceedings of the IEEE Virtual Reality 2004 (VR'04)-Volume 00*.

[Cavalheiro et al., 2006] Cavalheiro, G. G. H., Musse, S. R., Jung, C., Cordeiro, O. C., and Silveira, C. L. B. (2006). Vigilância e segurança com sistemas computacionais de alto desempenho. 1:(CD–ROM).

[Cercato et al., 2006] Cercato, F. P., Mombach, J. C. M., and Cavalheiro, G. G. H. (2006). High performance simulations of the celular potts model. In *HPCS '06: Proceedings of the 20th International Symposium on High Performance Computing Systems and Applications (HPCS'06)*, pages (CD–ROM). IEEE Computer Society.

[Coddington, 1993] Coddington, P. D. (1993). An analysis of distributed computing software and hardware for applications in computational physics. pages 179–186.

[Cordeiro, 2006] Cordeiro, O. C. (2006). Mathematica: O Mac a serviço da matemática. *Revista Mac+*, (7):62–67.

[Cordeiro et al., 2005] Cordeiro, O. C., Braun, A., Silveira, C. B., Musse, S. R., and Cavalheiro, G. G. H. (2005). Concurrency on social forces simulation model. *V-Crowd, International Workshop on Crowd Simulation*.

[Cordeiro et al., 2006] Cordeiro, O. C., Musse, S. R., and Cavalheiro, G. G. H. (2006). Explorando concorrência em simulação de humanos virtuais. *Escola Regional de Alto Desempenho (ERAD)*, pages 79–80.

[Crowd Dynamics, 2006] Crowd Dynamics (2006). Crowd disasters. http://www.crowddynamics.com/ acessado em 25-Agosto-2006.

[Dagum and Menon, 1998] Dagum, L. and Menon, R. (1998). Openmp: an industry standard api for shared-memory programming. *Computational Science and Engineering, IEEE [see also Computing in Science & Engineering]*, 5(1):46–55.

[Dorigo and Gambardella, 1997] Dorigo, M. and Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*, 1(1):53–66.

[Dorneles et al., 2003] Dorneles, R. V., Diverio, T. D., and Navaux, P. O. A. N. (2003). Dynamic load balancing in pc clusters: An application to a multi-physics model. *SBAC-PAD*, 00:192.

[Evers and Musse, 2002] Evers, T. F. and Musse, S. R. (2002). Build artificial memory to autonomous agents using dynamic and hierarquical. 1:164–170.

[Goldenstein et al., 1999] Goldenstein, S., Large, E., and Metaxas, D. (1999). Non-linear dynamical system approach to behavior modeling. *The Visual Computer*, 15:349–364.

[Google, 2007] Google (2007). SketchUp:vModel your World. http://sketchup.google.com/ acessado em 16-Janeiro-2007.

[Gusato et al., 2005] Gusato, E., Mombach, J. C. M., Cercato, F. P., and Cavalheiro, G. G. H. (2005). An Efficient Parallel Algorithm to Envolve Simulations of the Cellular Potts Model. *Parallel processing letters*, 15(1-2):199–208.

[Helbing, 1992] Helbing, D. (1992). A fluid-dynamic model for the movement of pedestrians. *Complex Systems*, 06(05):391–415.

[Helbing et al., 2000] Helbing, D., Farkas, I., and Vicsek, T. (2000). Simulating dynamical features of escape panic. *Nature*, 407:487–490.

[Hughes, 2003] Hughes, R. L. (2003). The flow of human crowds. *Annual Review of Fluid Mechanics*, 35:169–182.

[Jacqmot, 1996] Jacqmot, C. (1996). Load Management in Distributed Computing Systems: Towards Adaptive Strategies. Technical report, Technical Report, Ph. D. Thesis, Departement d¿Ingenierie Informatique, Universite catholique de Louvain.

[Karypis and Kumar, 1995] Karypis, G. and Kumar, V. (1995). *A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs*. Army High Performance Computing Research Center.

[Koradi et al., 2000] Koradi, R., Billeter, M., and Güntert, P. (2000). Point-centered domain decomposition for parallel molecular dynamics simulation. *Computer Physics Communications*, 124:139–147.

[Kuska, 2007] Kuska, J. P. (2007). MathGL3d: An Interactive OpenGL Based Viewer for Mathematica's 3D Graphics. http://library.wolfram.com/infocenter/MathSource/2986/ acessado em 16-Janeiro-2007.

[Langton, 1986] Langton, C. (1986). Studying artificial life with cellular automata. *Physica D*, 2(1-3):120–149.

[Langton et al., 1989] Langton, C. et al. (1989). *Artificial Life*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Lemke, 2002] Lemke, N. (2002). Aplicações de alto desempenho trivialmente paralelizáveis. *Escola Regional de Alto Desempenho (ERAD)*, (2):107–138.

[Levin, 1989] Levin, E. (1989). Grand challenges to computation science. *Commun. ACM*, 32(12):1456–1457.

[Lewin, 1951] Lewin, K. (1951). *Field Theory in Social Science*. Harper, New York.

[Lowenthal et al., 1996] Lowenthal, D., Freeh, V., and Andrews, G. (1996). Using Fine-Grain Threads and Run-Time Decision Making in Parallel Computing. *Journal of Parallel and Distributed Computing*, 37(1):41–54.

[Maes, 1995] Maes, P. (1995). Artificial life meets entertainment: Life like autonomous agents. In *Communications of the ACM*.

[Mataric, 1994] Mataric, M. J. (1994). Leaning to behave socially. In *SAB94: Proceedings of the third international conference on Simulation of adaptive behavior : from animals to animats 3*, pages 453–462, Cambridge, MA, USA. MIT Press.

[Musse and Thalmann, 1997] Musse, S. R. and Thalmann, D. (1997). A model of human crowd beahavior. *EUROGRAPHICS Workshop on Computer Animation and Simulation*, pages 39–51.

[Musse and Thalmann, 2001] Musse, S. R. and Thalmann, D. (2001). Hierarquical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics*, 7:152–164.

[OpenMP, 1997] OpenMP (1997). OpenMP: A Proposed Standard API for Shared Memory Programming. http://www.openmp.org/specs/mp-documents/paper/paper.ps. Visitado em Maio de 2006.

[Plimpton and Hendrickson, 1996] Plimpton, S. and Hendrickson, B. (1996). A new parallel method for molecular dynamics simulation of macromolecular systems. *Journal of Computational Chemistry*, 17(3):326–337.

[Quinn et al., 2003] Quinn, M. J., Metoyer, R. A., and Zaworski, K. H. (2003). Parallel implementation of the social force model. In *Int. Conf. in Pedestrian and Evacuation Dynamics*.

[Rapaport, 2004] Rapaport, D. (2004). *The Art of Molecular Dynamics Simulation*. Cambridge University Press.

[Renault et al., 1990] Renault, O., Thalmann, D., and Magnenat-Thalmann, N. (1990). A vision-based approache to behavioral animation. *The Journal of Visualization and Computer Animation*, 1(1):18–21.

[Reynolds, 1987] Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, New York, NY, USA. ACM Press.

[Seyfried et al., 2006] Seyfried, A., Steffen, B., and Lippert, T. (2006). Basics of modelling the pedestrian flow. *Physica A: Statistical Mechanics and its Applications*, 368(1):232–238.

[Srinivasan et al., 1997] Srinivasan, S., Ashok, I., Jonsson, H., Kalonji, G., and Zahorjan, J. (1997). Parallel short-range molecular dynamics using the adhara runtime system. *Comput. Phys. Comun.*, 102:1–3.

[Taylor, 2001] Taylor, B. (2001). *The International System of Units (SI)*. US Dept. of Commerce, Technology Administration, National Institute of Standards and Technology.

[Taylor, 1995] Taylor, B. N. (1995). *Guide for the Use of the International System of Units (SI)*. National Institute of Standards and Technology.

[Treuille et al., 2006] Treuille, A., Copper, S., and Popović, Z. (2006). Continuum crowds. In *SIGGRAPH 2006: Proceedings of the 33rd annual conference on Computer graphics and interactive techniques.*

[Tu and Terzopoulos, 1994] Tu, X. and Terzopoulos, D. (1994). Artificial fishes: physics, locomotion, perception, behavior. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 43–50, New York, NY, USA. ACM Press.

[Ulicny and Thalmann, 2001] Ulicny, B. and Thalmann, D. (2001). Crowd simulation for interactive virtual environments and vr training systems. In *Computer Animation and Simulation '01*, SpringerComputerScience, pages 163–170. Proceedings of the Eurographics Workshop in Manchester.

[Valiant, 1990] Valiant, L. G. (1990). A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111.

[Willebeek-LeMair and Reeves, 1993] Willebeek-LeMair, M. H. and Reeves, A. P. (1993). Strategies for Dynamic Load Balancing on Highly Parallel Computers. *IEEE Transactions on Parallel and Distributed Systems*, 4(9):979–993.

[Wolfram, 1991] Wolfram, S. (1991). *Mathematica: a system for doing mathematics by computer.* Addison Wesley Longman Publishing Co., Inc. Redwood City, CA, USA.

[Yang and Gerasoulis, 1992] Yang, T. and Gerasoulis, A. (1992). PYRROS: static task scheduling and code generation for message passing multiprocessors. *Proceedings of the 6th international conference on Supercomputing*, pages 428–437.