UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RENATO SILVEIRA

Planejamento de Movimento para Grupos utilizando Campos Potenciais

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof[a]. Dr[a]. Luciana Porcher Nedel
Orientador

Prof. Dr. Edson Prestes
Co-orientador

Porto Alegre, abril de 2015

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Silveira, Renato

Planejamento de Movimento para Grupos utilizando Campos Potenciais / Renato Silveira. – Porto Alegre: PPGC da UFRGS, 2015.

78 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2015. Orientador: Luciana Porcher Nedel; Co-orientador: Edson Prestes.

1. Planejamento de movimento, planejamento de movimento para grupos, formações, computação gráfica. I. Nedel, Luciana Porcher. II. Prestes, Edson. III. Title.

“Se soubéssemos o que estávamos fazendo, não seria pesquisa, seria?”
— ALBERT EINSTEIN

AGRADECIMENTOS

Gostaria agradecer a todas as pessoas e instituições que deram sua contribuição para a realização deste trabalho ou que, de alguma forma, incentivaram-me a continuar. Gostaria de agradecer aos meus orientadores, Luciana Nedel e Edson Prestes, pelo empenho e pela referência que foram para mim, e por me incentivar em continuar sempre. Agradeço a Luciana por ter-me possibilitado estes 2 anos de intenso aprendizado e por aumentar a minha auto-estima nos momentos em que tudo era trevas. Agradeço ao Edson pela disponibilidade de conversar, pela genialidade ao sugerir idéias e por nunca me deixar na mão, além das boas conversas informais.

Agradeço a todos do grupo de computação gráfica da UFRGS, por todos os momentos. Em especial, agradeço: ao Leandro A. F. Fernandes, por ter contribuído com os vídeos durante a submissão dos artigos; ao Marcos Slomp, ao Fábio Bernardon, ao Carlos Dietrich, ao Giovane R. Kuhn e ao Vitor Pamplona por terem me ensinado não apenas computação gráfica, mas por terem compartilhado comigo suas experiências em computação. Agradeço ao André Spritzer pelas revisões de inglês, e também a todos os outros membros do grupo da CG, pessoas com as quais convivi e aprendi muito.

Em especial, agradeço ao Bruno Schneider, meu primeiro professor de computação gráfica, que me inspirou a continuar nesta área.

Agradeço ao meu antigo professor Antonio Tavares da Costa Junior por todo o ensinamento de Matemática e Física, que me permitiu ajudar muitos colegas. Agradeço a toda a minha família pelo apoio, paciência e ajuda durante esses 2 anos.

Finalmente, agradeço ao CNPq pelo auxílio financeiro durante o curso, e ao Instituto de Informática, pela estrutura oferecida durante o mestrado.

SUMÁRIO

LISTA DE FIGURAS

RESUMO

A simulação do movimento de humanos virtuais em mundos sintéticos é necessária a áreas tais como: jogos eletrônicos, filmes, ambientes virtuais colaborativos como Second Life®, simulação de pedestres, sistemas de treinamento e simulação de evacuações em situações de emergência. Contudo, mesmo sendo um importante tópico de pesquisa desde a década de 70, a simulação de comportamentos para personagens virtuais ainda é um desafio.

O principal objetivo deste trabalho é estender a técnica proposta por Dapper et al. (DAPPER, 2007) onde foi apresentado um planejador de movimento para humanos virtuais que fornece trajetórias suaves e dependentes de parâmetros individuais dos agentes, permitindo a navegação de grupos de forma mais eficiente.

Neste trabalho, é proposto um algoritmo para controlar o movimento de grupos em ambientes interativos e uma estratégia para manter a formação durante o deslocamento. O método é baseado em campos potenciais, ou seja, na solução numérica de problemas de valores de contorno envolvendo a equação de Laplace. O método proposto, é composto de duas camadas. Na primeira camada, um mapa para o controle do grupo é criado para possibilitar o controle de cada indivíduo, enquanto que na segunda camada, um algoritmo de planejamento de caminho é utilizado para o movimento do grupo como um todo. A técnica proposta combina o planejamento de movimento para grupos com a navegação baseada em esboços. Os resultados mostram que a técnica é robusta e pode ser utilizada em tempo-real.

Palavras-chave: Planejamento de movimento, planejamento de movimento para grupos, formações, computação gráfica.

Group Motion Planning using Potential Fields

ABSTRACT

The motion simulation of the movement of virtual humans in synthetic worlds is necessary for areas such as: electronic games, movies, collaborative virtual environments such as Second Life®, pedestrian simulation, training systems, simulation of evacuations in emergency situations. However, although being an important research topic since the 70s, the simulation of behaviors for virtual characters is still a challenge.

Dapper et al. (DAPPER, 2007) developed a motion planner that provides smooth trajectories dependent on the individual parameters of the agents.

The main goal of this work is to extend the technique proposed by Dapper, allowing for a more efficient control and navigation of groups. In this work, an algorithm for the control of group motion in interactive environments is proposed, along with a strategy for maintaining the group formation during the motion. The method is based on potential fields, more specifically, on the numeric solution of boundary-value problems involving the Laplace equation. The proposed method has two layers. In the first layer, a map for group control is created in order to allow for the control of every individual, while in the second layer a path planning algorithm is used for the movement of the group as a whole. The proposed technique combines the planning of group movements with a sketch-based navigation. The results show that the technique is robust and can be used in real-time.

Keywords: motion planning, group motion planning, formation, computer graphics.

1 INTRODUÇÃO

A simulação do movimento de humanos virtuais em mundos sintéticos é necessária a áreas tais como: como jogos eletrônicos, filmes, ambientes virtuais colaborativos como Second Life®, simulação de pedestres, sistemas de treinamento de exércitos e simulações de guerra e planejamento de evacuações em situações de emergência. Contudo, mesmo sendo um importante tópico de pesquisa desde a década de 70, a simulação de comportamentos para personagens virtuais ainda é um grande desafio.

Agentes autônomos devem apresentar um comportamento individual baseado em sua personalidade, em seus humores e em seus desejos. Grupos de agentes devem mover-se de maneira coesa, obedecendo certas regras sociais como seguir o mesmo caminho, caminhar em uma mesma velocidade média, manter uma distância regular de seus vizinhos, possuir um objetivo em comum e manter uma formação pré-definida. Lidar com grupos de agentes é uma tarefa bastante complexa devido ao grande número de elementos envolvidos, tais como: eficiência, gerenciamento de interações dinâmicas entre agentes, complexidade e sutileza de comportamentos (TREUILLE; COOPER; POPOVIć, 2006).

Várias áreas têm dado diferentes definições do que é um grupo (MUSSE et al., 2005). Na computação, grupos são frequentemente definidos como uma coleção de indivíduos, no mesmo ambiente físico compartilhando um mesmo objetivo (MUSSE; THALMANN, 2001).

Com essa definição, no domínio da computação gráfica, grupos de personagens são amplamente usados pela indústria de jogos, principalmente em jogos de estratégia em tempo-real (RTS). Nesses jogos, grupos – normalmente exércitos – comportam-se de forma coerente (em formação) e jogadores selecionam unidades e definem tarefas clicando com o mouse. Apesar desses jogos se terem tornado bastante populares e alcançado um grande sucesso comercial, nenhum progresso significativo foi obtido na maneira como os usuários interagem, utilizando o mouse, com os personagens virtuais. O mouse é um dispositivo intuitivo de se manipular, mas a maneira como é utilizado não tem sido muito eficiente. Muitos “clicks” são necessários para especificar uma única tarefa em um grupo, fazendo com que o usuário preocupe-se com o controle e perca o foco da estratégia (GOETZ, 2006).

1.1 Motivação

As motivações para o desenvolvimento deste trabalho surgiram ao tentar responder algumas questões sobre o planejamento de movimento de grupos. Essas questões correspondem a tópicos de pesquisas correntes:

- Movimento natural:

  - Como as pessoas caminham através de construções?
  - Como o caminho varia em função da velocidade?

- Manipulação ou gerenciamento de grupos:
  - Como gerenciar múltiplos grupos?
  - Como simular um comportamento estratégico?

- Viabilidade de alcançar objetivos:
  - Como lidar com objetivos imprecisos?
  - Como lidar com a interrupção dos movimentos e com mudanças nos objetivos?

A eficiência é um fator fundamental, visto que uma cena, ou seja, uma configuração das entidades, deve ser renderizada com freqüência situada em na faixa de 20 a 100 quadros por segundo para ser visualmente aceitável. Assim, tem-se por volta de 0.02 segundos por quadro. Em um sistema interativo, este tempo inclui: a atualização dos estados do sistema; o controle das interações, ou seja, dos dados de entada; o processamento gráfico; o cálculo da física necessária; o cálculo da IA utilizada e o cálculo do planejamento de movimento. Isso implica em um tempo menor que 1 milissegundo para o planejamento do movimento das entidades!

Controle de formação tem sido um dos mais importantes tópicos de pesquisa em sistemas multirrobôs (BALCH; HYBINETTE, 2000). Nessa área, é necessário mover vários robôs dispostos em uma forma geométrica específica. Muitas tarefas complicadas, como salvamento, transporte e construção (SONG; KUMAR, 2002) utilizam formações para que os robôs ou agentes atinjam seus objetivos. A importância de formações nesse contexto é permitir que membros de um grupo concentrem-se em uma pequena porção do ambiente, enquanto seus parceiros cobrem o resto.

Em computação gráfica, as técnicas atuais para lidar com formações não são suficientemente genéricas para integrar de maneira satisfatória o problema do movimento dos grupos com o problema de manter a formação. Por conseguinte, o usuário não pode definir a forma desejada interativamente.

Levando em consideração essas questões, este trabalho apresenta um estudo sobre planejamento de movimento para grupos de humanos virtuais. Como resultado prático, foi desenvolvida uma extensão do algoritmo proposto por Dapper et al. (DAPPER, 2007), baseado em problemas de valores-de-contorno. A extensão proposta permite encontrar caminhos suaves e variados em ambientes dinâmicos, onde esses caminhos dependem das características individuais de cada grupo, as quais podem ser alteradas dinamicamente.

A abordagem proposta neste trabalho permite a interação entre os membros de um grupo de humanos virtuais, enquanto mantém a coesão do grupo durante seu deslocamento. A extensão sugerida é suficientemente genérica para ser utilizada com qualquer planejador de caminhos, não apenas o proposto por Dapper et al. (DAPPER, 2007).

1.2 Objetivos do Trabalho

O principal objetivo deste trabalho foi estender a técnica proposta por Dapper et al. (DAPPER, 2007), para permitir o controle e a navegação de grupos de forma mais eficiente. Mais especificamente, pretende-se:

- Desenvolver um algoritmo robusto e elegante para controlar a navegação de grupos inspirado em um planejador de caminhos baseado em problemas de valor-de-contorno (BVP);

- Elaborar uma estratégia para lidar com o problema de formações em grupos, que permitisse ao usuário definir qualquer formação e

- Integrar o planejador proposto por Dapper et al. com uma forma de interação em alto nível, baseada em esboços, que permitisse a especificação de uma trajetória para um grupo.

A técnica proposta é robusta no que se refere às formações que podem ser definidas, ao tamanho do grupo, à mudança dinâmica entre formações e ao desvio de obstáculos durante o movimento dos agentes. Além disso, por ter como núcleo a solução numérica de um problema de valor-de-contorno, ela pode ser computada de forma eficiente através do uso de arquiteturas paralelas, tais como multicores e GPUs.

1.3 Organização do Texto

O restante do texto está organizado da seguinte maneira:

- **Capítulo 2** apresenta as abordagens utilizadas para lidar com o problema de planejamento de movimento para humanoides virtuais, individualmente ou em grupo, e os trabalhos relacionados nessa linha de pesquisa;

- **Capítulo 3** apresenta o planejador de caminhos baseado em funções harmônicas proposto por Dapper et al. (DAPPER, 2007);

- **Capítulo 4** relata as contribuições feitas na técnica de Dapper, e sugere novas técnicas para lidar de forma mais eficiente com as limitações dessa técnica. São elas:

  – Otimizações na etapa do relaxamento;

  – Análise formal de possíveis comportamentos através do tamanho da janela local.

- **Capítulo 5** apresenta uma nova técnica para geração em alto-nível de campos potenciais e propõe uma maneira para lidar com grupos. As principais contribuições desse capítulo são:

  – Integração com um planejador de caminhos em alto-nível;

  – Controle de Grupo - é proposta uma nova técnica para lidar com grupos de forma eficiente, garantindo a coesão e possibilitando qualquer tipo de formação.

- **Capítulo 6** discute os resultados obtidos nos experimentos realizados e expõe os ganhos de desempenho que a nova técnica proporciona, assim como os detalhes da implementação.

- **Capítulo 7** discute as considerações finais e apresenta idéias para trabalhos futuros.

2 PLANEJAMENTO DE MOVIMENTOS EM AMBIENTES VIRTUAIS

O problema do planejamento de movimento em um ambiente virtual pode ser definido da seguinte maneira: dados critérios como distância e tempo, qual é a melhor seqüência de ações que um agente deve realizar para alcançar um objetivo? O planejamento de movimentos é utilizado em vários tipos de situações, como o movimento interno das articulações de humanos virtuais e o planejamento das ações que os agentes realizarão durante a navegação em um ambiente.

Ambientes virtuais e jogos, em particular, oferecem um problema ainda mais desafiador do que o problema do planejamento de movimento, devido aos seguintes aspectos (NIEUWENHUISEN; KAMPHUIS; OVERMARS, 2007; LAVALLE, 2006):

- **Complexidade**: as cenas são bastante complexas e, em geral, possuem uma quantidade significativa de obstáculos como salas, corredores e passagens estreitas;
- **Dinamismo**: as cenas tendem a ser dinâmicas, ou seja, os obstáculos podem aparecer e desaparecer a qualquer momento, por exemplo, quando se abre uma porta ou uma ponte;
- **Tempo-real**: os movimentos devem ser computados em tempo-real e podem ser modificados pelo usuário a qualquer momento, por exemplo, o usuário pode mandar uma entidade parar ou prosseguir. Além disso, os objetivos não são precisos e estão em constante mudança;
- **Vários graus de liberdade**: uma aproximação da estrutura do corpo humano pode possuir centenas de graus de liberdade, e o movimento das combinações destas estruturas deve ser planejado;
- **Múltiplas entidades**: freqüentemente, várias entidades movem-se simultaneamente no mesmo ambiente e é necessário que, enquanto se comportam como grupo, não colidam umas contra as outras;
- **Movimento natural**: para dar ao usuário uma sensação de imersão, o movimento deve ser natural, isto é, deve ser visualmente convincente.

Conforme pode-se observar, o planejamento de movimento para uma entidade é uma tarefa bastante complexa, já que existem diversos aspectos considerar. A adição de vários agentes torna o problema ainda mais difícil e, em alguns casos, intratável (NIEUWENHUISEN; KAMPHUIS; OVERMARS, 2007).

Na Seção 2.1 são apresentadas técnicas freqüentemente utilizadas para o planejamento de movimento de um único agente, enquanto que na Seção 2.2 são apresentadas as técnicas existentes para o planejamento de movimento de grupos de agentes.

2.1 Planejamento de Movimento

O movimento de entidades é frequentemente planejado utilizando uma combinação de métodos como *scripts*, grades como A^*, métodos locais reativos como campos potenciais e *roadmaps*. Esses métodos são explicados a seguir.

2.1.1 Métodos Baseados em Scritps

Nos métodos baseados em *scripts*, o artista descreve explicitamente o caminho que poderá ou que deverá ser seguido pelos agentes, usando algum tipo de linguagem. Isso normalmente é feito durante a etapa de modelagem do ambiente. Criar *scripts* dessa maneira é um processo que consome bastante tempo do artista. Além disso, é provável que ocorram comportamentos repetitivos e indesejáveis para os agentes, pois os caminhos gerados são restritos e pré-definidos, o que pode ser facilmente observado pelos usuários (NIEUWENHUISEN; KAMPHUIS; OVERMARS, 2007). Além disso, o ambiente pode não ser completamente conhecido durante a etapa de modelagem.

O uso de *scripts* torna-se bastante complicado quando muitas entidades podem mover-se no mesmo ambiente, negociando o espaço, ou quando esse ambiente for grande, o que é o caso da maioria dos ambientes virtuais e dos jogos atuais.

2.1.2 Métodos Baseados em Grades

Os métodos baseados em grades geralmente discretizam o ambiente em uma malha de células e planejam o movimento que as entidades irão realizar, utilizando algum algoritmo de busca sobre as células livres, como os algoritmos baseados em A^* (DELOURA, 2000; RUSSELL; NORVIG, 2003).

Se o ambiente for pequeno, essa técnica é bastante eficiente. Porém, quando o ambiente é complexo e grande, e possui muitas entidades que deverão se mover, estes métodos tornam-se custosos, exigindo uma grande quantidade de memória e de processamento.

Através de heurísticas específicas para reduzir a busca, pode-se diminuir o tempo de processamento. Porém, isso pode produzir muitos caminhos errados. Além disso, caminhos criados por esses métodos geralmente são visivelmente não-naturais. Geralmente, é necessária uma fase de pós-processamento para se produzir caminhos suaves, aumentando assim o custo computacional. A Figura 2.1 mostra um exemplo de um ambiente discretizado onde uma busca forneceu um caminho entre a posição em que o agente se encontra (célula verde) e a posição do objetivo (célula azul). As células em vermelho representam o caminho.

2.1.3 Métodos Baseados em *Roadmaps*

Durante as últimas três décadas, vários planejadores de caminhos foram desenvolvidos na área da robótica, entre eles, o método baseado em *Roadmaps*. Esses métodos têm sido estudados por muitos autores (KAVRAKI; LATOMBE, 1994; KAVRAKI; KOLOUNTZAKIS; LATOMBE, 1996; SVESTKA; OVERMARS, 1998; HOLLEMAN; KAVRAKI, 2000; GERAERTS; OVERMARS, 2004) e aplicados em diversos tipos de tarefas, tais como: navegação de agentes em ambientes complexos (BAYAZIT; LIEN; AMATO, 2003;

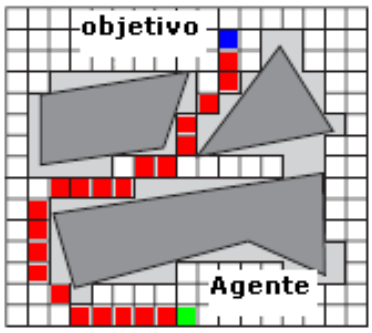
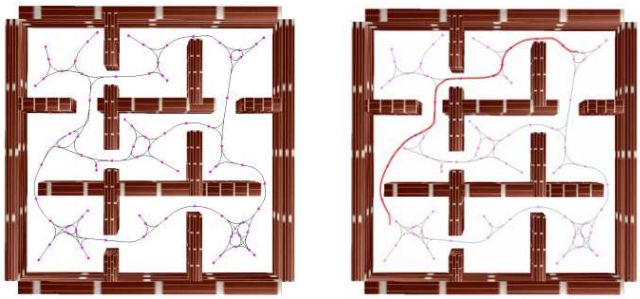


Figura 2.1: Abordagem baseada em grade.

KAVRAKI; LATOMBE, 1994; KAVRAKI; KOLOUNTZAKIS; LATOMBE, 1996), movimentos coordenados de robôs (SVESTKA; OVERMARS, 1998), movimento de articulações de robôs com vários graus de liberdade (HOLLEMAN; KAVRAKI, 2000), entre outros (GERAERTS; OVERMARS, 2004).

Em fase de pré-processamento, esses métodos constroem um mapa das possíveis configurações dos movimentos de um robô no ambiente, chamado *roadmap*. Esse mapa consiste em uma rede de curvas e vértices que resumem a informação do ambiente (KAVRAKI; LATOMBE, 1994).

Como exemplo, a Figura 2.2-a mostra um *roadmap* criado em um ambiente retangular. Quando uma consulta por um caminho particular é feita, o *roadmap*, através de uma simples e rápida busca no mapa, utilizando algoritmos clássicos de busca em grafo, retorna esse caminho. A Figura 2.2-b mostra um caminho encontrado no *roadmap*. Essa abordagem possui um baixo custo computacional mesmo em ambientes complexos, porém, pode produzir caminhos de baixa qualidade, necessitando de um custo computacional a mais para sua suavização.



(a) (b)

Figura 2.2: Abordagem baseada em *roadmaps*. (a) criação de um roadmap. (b) Detecção de um caminho arbitrário. (NIEUWENHUISEN; KAMPHUIS; OVERMARS, 2007)

Roadmaps probabilísticos também foram amplamente utilizados para planejar o movimento de agentes (KAVRAKI et al., 1994; KAVRAKI; KOLOUNTZAKIS; LATOMBE, 1996). Isso consiste em amostrar aleatoriamente alguns pontos na região livre do ambiente, e criar o *roadmap* a partir desses pontos.

2.1.4 Métodos Baseados em Potenciais

Os métodos baseados em potenciais foram inicialmente propostos por Khatib (KHATIB, 1980) e definem campos de forças potenciais entre os agentes no ambiente, para

produzir os comportamentos desejados.

Khatib (KHATIB, 1980), na década de 80, considerou que, ao invés de procurar por um caminho de qualidade e ajustar o movimento do agente para se adequar a ele, um bom planejador deveria prover um campo potencial, ou um campo de força, que se expande por todo o ambiente, produzindo caminhos alternativos. O campo potencial é concebido para incorporar obstáculos e objetivos, e deve guiar o agente sempre indicando a melhor direção a ser seguida, semelhante à simulação física de um corpo sujeito a forças externas. Os obstáculos exercem forças repulsoras que impedem o agente de colidir e os objetivos exercem forças atrativas que "puxam" os agentes em sua direção.

A técnica de campos potenciais, quando aplicada localmente, é denominada "técnica reativa", onde o objetivo é adaptar um movimento previamente computado aos obstáculos encontrados próximo às entidades, os quais não foram tratados previamente durante o planejamento. Esses obstáculos podem ser outras entidades, ou obstáculos dinâmicos (REIF; WANG, 1995; LAMIRAUX; BONNAFOUS; LEFEBVRE, 2004). A Figura 2.3 mostra um exemplo de campos potenciais, onde se pode observar que o potencial final é a superposição do potencial associado aos obstáculos com o potencial associado ao objetivo.

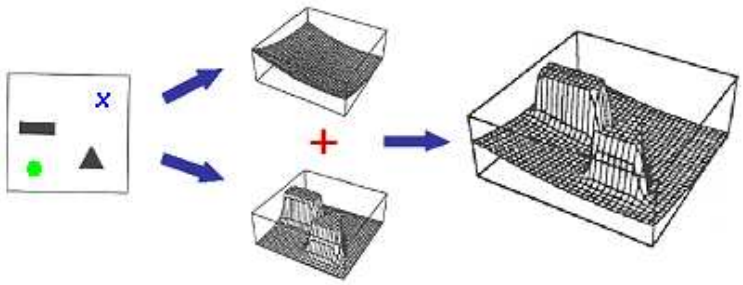


Figura 2.3: Abordagem utilizando Campos Potenciais.

Geralmente, a utilização dessa técnica localmente pode levar a mínimos locais, ou seja, as entidades não conseguem saber para onde se mover, ficando presas em algum local no ambiente. A Figura 2.4 ilustra um exemplo de mínimo local. Isso se dá porque a força resultante das ações de repulsão exercidas pelos obstáculos com a ação de atração exercida pelo objetivo é nula.

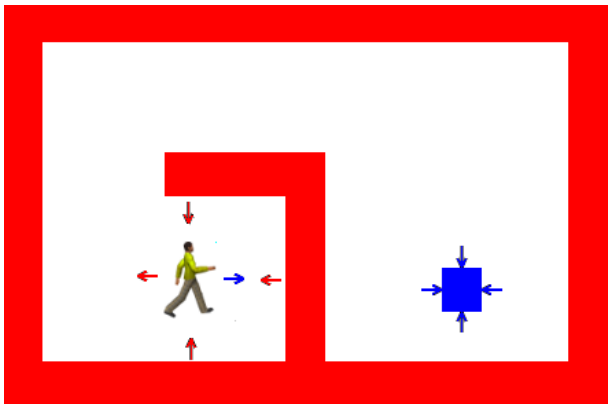


Figura 2.4: Exemplo de mínimo local. As setas vermelhas indicam as forças de repulsão dos obstáculos, enquanto a seta azul indica a força de atração do objetivo. As forças se anulam e o agente fica preso.

2.2 Planejamento de Movimento Para Grupos

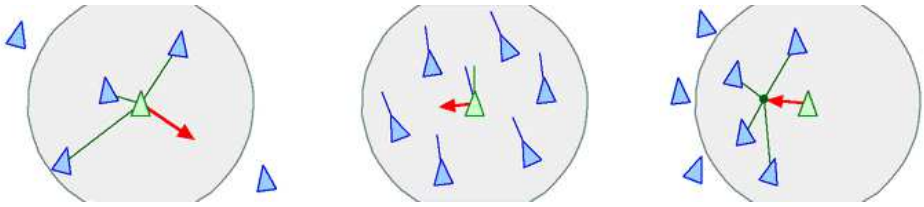
Planejamento de caminhos, navegação e simulações de grupos têm sido amplamente estudados em diversas áreas, como na computação gráfica, na indústria de jogos e, inclusive, na comunidade da robótica. Uma exposição de trabalhos relevantes sobre esses temas é feita a seguir.

2.2.1 Navegação de Grupos em Ambientes Virtuais

Na computação gráfica e na indústria de jogos, as abordagens mais comumente utilizadas para simular o movimento de entidades comportam-se como um grupo são aquelas baseadas em *flocking*, introduzidas por Reynolds (REYNOLDS, 1987). Em seu trabalho clássico (REYNOLDS, 1987), Reynolds propôs um modelo para descrever o comportamento de pássaros em um grupo, apelidados de *boids*, utilizando apenas regras locais individuais como:

- um *boid* deve sempre distanciar-se de *boids* vizinhos de forma a evitar colisões (Figura 2.5-a);
- um *boid* deve alinhar o sentido de seu movimento com aquele dos *boids* vizinhos (Figura 2.5-b);
- um *boid* deve aproximar-se da posição média dos *boids* vizinhos de forma a manter a coesão do grupo (Figura 2.5-c).

A Figura 2.5 mostra estas regras locais. O comportamento final é dado através da



(a) (b) (c)

Figura 2.5: (a) Separação. (b) Alinhamento. (c) Coesão.

soma vetorial de cada um dos comportamentos individuais. Embora o movimento obtido através dessa técnica seja natural, não há controle sobre o caminho a ser seguido nem há garantia de que uma entidade alcançará seu objetivo final. Por isso, essa técnica é frequentemente utilizada em conjunto com outras técnicas para que seja possível garantir que todos os agentes atinjam seus objetivos. Freqüentemente, utiliza-se *flocking* no planejamento local de movimentos, enquanto utiliza-se alguma outra técnica para planejar o movimento de maneira geral (BAYAZIT; LIEN; AMATO, 2003, 2002).

Posteriormente, Reynolds estendeu sua técnica para incluir comportamentos autônomos reativos, com o intuito de obter resultados mais convincentes (REYNOLDS, 1999). Recentemente, Reynolds (REYNOLDS, 2006) implementou uma forma de simulação e animação de multiagentes com alto desempenho na arquitetura do PlayStation® 3. Essa técnica permitiu uma implementação escalável em multiprocessadores de uma simulação de multidões composta por muitos agentes de forma bastante eficiente, obtendo uma boa taxa de quadros por segundo com milhares de agentes. A Figura 2.6 ilustra o tipo de cenário que pode ser tratado com o uso dessa técnica.

Figura 2.6: Cenário de simulação de multiagente no PlayStation® 3 tratado com a técnica proposta por Reynolds (REYNOLDS, 2006).

Também na indústria de jogos, Nieuwenhuisen et al. (NIEUWENHUISEN; KAMPHUIS; OVERMARS, 2007) concentraram seus esforços na solução do problema da qualidade da navegação dos personagens em um grupo em ambientes virtuais, e propuseram um método baseado em *roadmaps* probabilísticos. Esta técnica provou ser eficiente quando aplicada à jogos.

Sanchez e Latombe (SANCHEZ; LATOMBE, 2002) e Svestka e Overmars (SVESTKA; OVERMARS, 1998) propuseram técnicas eficientes para planejar o movimento de grupos de agentes. Nessas técnicas, diferentes entidades são tratadas como um amplo sistema hierárquico. Além disso, cada entidade possui dois graus de liberdade (considerando que estes são definidos por suas posições em uma superfície planar) e portanto, o sistema total possui $2n$ graus de liberdade para tratar de n entidades. Quando n é grande, o tempo de processamento torna-se muito elevado. Esta técnica parte do pressuposto que o número de entidades é previamente conhecido. Visando tornar essa técnica ainda mais eficiente, Leroy et al. (LEROY; LAUMOND; SIMéON, 1999) propuseram novos algoritmos geométricos, possibilitando o planejamento de movimento para uma grande quantidade de entidades através da consulta de um diagrama de coordenação, construído a partir da *bounding box* dos obstáculos presentes no ambiente.

Bayazit et al. (BAYAZIT; LIEN; AMATO, 2003) combinaram técnicas de *roadmaps* com a técnica de *flocking*. As entidades utilizam os mapas criados pelo planejador para guiar seus movimentos e alcançar um objetivo, enquanto a técnica de *flocking* é utilizada para as entidades se manterem coesas e se comportarem como um grupo, também evitando colisões locais. A Figura 2.7 ilustra essa técnica.

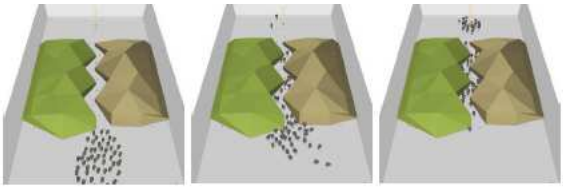


Figura 2.7: Técnica proposta por Bayazit et al (BAYAZIT; LIEN; AMATO, 2003). Agentes passando por uma passagem estreita.

Li e Chou (LI; CHOU, 2003) desenvolveram uma abordagem que permite a reestruturação dinâmica das entidades, de tal forma que planejadores centralizados de movimentos pudessem ser melhorados. Li e Chou propuseram um planejador que utiliza uma árvore hierárquica esférica para agrupar as entidades. Mas com essa técnica, não é possível tratar da coesão do grupo.

Pesquisas em simulação de multidões também investigaram o movimento de grupos de entidades em ambientes virtuais. Essa área vem recebendo muita atenção nas últimas décadas (MUSSE; THALMANN, 2001; ULICNY; THALMANN, 2001). Helbing (HELBING, 1994; HELBING; MOLNAR, 1995) propôs um modelo matemático para simular o comportamento do tráfego de pedestres, baseado nos modelos de difusão. A Figura 2.8 ilustra os resultados descritos por esse modelo.

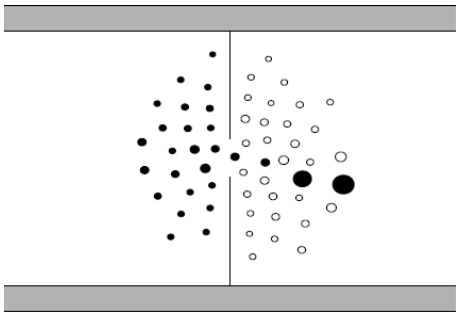


Figura 2.8: Modelo físico para simulação de comportamento de pedestres proposto por Helbing (HELBING, 1994).

Sud et al. (SUD et al., 2006) desenvolveram uma técnica para planejamento de caminhos em tempo real para múltiplos agentes virtuais em um ambiente dinâmico. Para isso, foi introduzido uma nova estrutura de dados chamada “grafo para navegação de multi-agentes”(MaNG) que é computada de forma eficiente utilizando Diagramas de Voronoi na GPU. As técnicas baseadas no Diagrama de Voronoi são, geralmente, utilizadas para ambientes estáticos e neste trabalho foi feita uma extensão para suportar ambientes dinâmicos. A vantagem dessa técnica é que não é necessário uma estrutura de dados para cada agente individualmente. A Figura 2.9 ilustra esse trabalho.

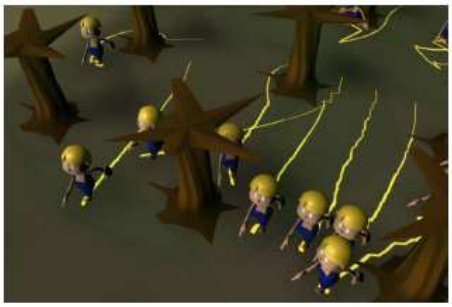


Figura 2.9: Agentes passando em um ambiente repleto de obstáculos (SUD et al., 2006).

Treuille et al. (TREUILLE; COOPER; POPOVIC, 2006) propuseram um modelo para planejamento de movimento em tempo real para populações, sem o uso de dinâmica para cada agente. Treuille considera o movimento como uma minimização de energia para cada personagem, e adota uma perspectiva contínua do sistema. Essa formulação produz um conjunto dinâmico de potenciais e campos de velocidade sobre o domínio, que guia o movimento individual de cada agente. O problema dessa técnica é o fato de lidar apenas com personagens com objetivos em comum, não tratando dos objetivos individuais. Uma tela da aplicação desenvolvida pode ser vista na Figura 2.10.

Recentemente, Berg et al. (BERG et al., 2008) propuseram uma técnica para planejamento de caminho e navegação interativa de agentes em ambientes virtuais na presença

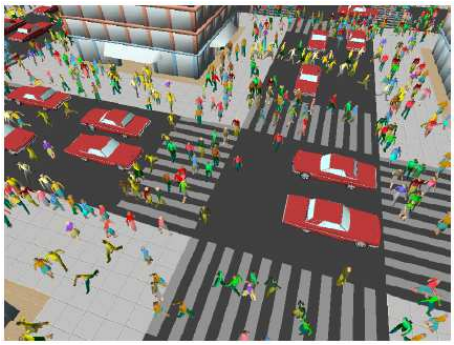


Figura 2.10: Agentes guiados pela técnica proposta por Treuille et al. (TREUILLE; COOPER; POPOVIC, 2006).

de multidões e obstáculos móveis, fazendo uso novamente de *roadmaps* probabilísticos. Uma tela da aplicação desenvolvida pode ser vista na Figura 2.11 ilustra esse trabalho.

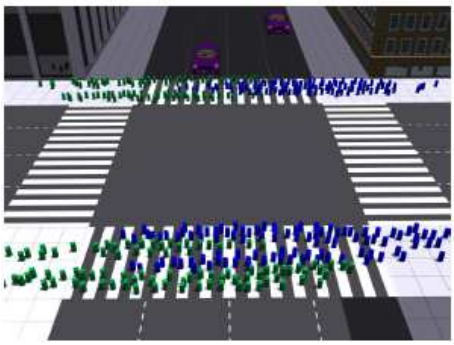


Figura 2.11: Vários agentes interagem ao atravessar a rua (BERG et al., 2008).

A seguir, são comentadas algumas técnicas que abordam o problema da coerência do grupo, geralmente resolvido com a adição de formações.

2.2.2 Coerência do Grupo e Controle de Formação

Todas as técnicas mencionadas apresentam solução para o problema de planejamento de movimento para grupos. Porém, a maioria delas falha quando existe uma exigência do grupo se comportar de forma coesa, sem haver quebra ao passar por obstáculos ou passagens estreitas. Foram realizadas algumas pesquisas para solucionar esse problema, e algumas técnicas relevantes surgidas serão descritas a seguir.

No domínio da computação gráfica, Kamphuis e Overmars (KAMPHUIS; OVERMARS, 2004b) propuseram uma abordagem para planejar caminhos de um grupo de entidades onde era possível manter a coerência entre elas. Essa técnica se baseia-se em um algoritmo multifase, onde na primeira fase, é planejado um caminho para um retângulo deformável, que delimita a área de atuação do grupo. Na segunda fase, o movimento das entidades internas a esse retângulo deformável é computado através de potenciais sociais. Posteriormente, os caminhos local e global são combinados para gerar um caminho final para as entidades.

Em seguida, Kamphuis e Overmars (KAMPHUIS; OVERMARS, 2004a) propuseram melhorias na técnica previamente proposta para obter caminhos de melhor qualidade modificando a forma retangular para corredores circulares deformáveis. A Figura 2.12 ilustra o caminho gerado para o movimento do grupo.

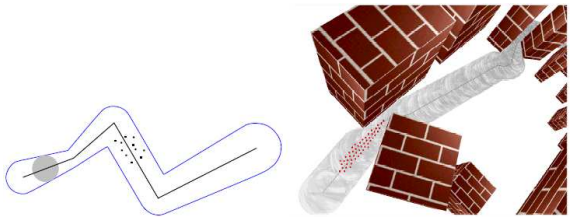


Figura 2.12: Técnica proposta por Kamphuis e Overmars (KAMPHUIS; OVERMARS, 2004a).

Pottinger (POTTINGER, 1999a,b), em dois breves artigos, propôs uma maneira alternativa para implementar formações pré-definidas em jogos. A abordagem de Pottinger trata o problema de agrupamento de unidades, o problema da detecção de colisões e o problema da movimentação do grupo no ambiente. Contudo, essa abordagem produz resultados não-naturais.

Li e Gupta (LI; GUPTA, 2005) propuseram uma técnica para lidar com o problema de formações de um grande grupo de agentes afim de se moverem rigidamente em formação em um ambiente 2D, utilizando formas elásticas lineares pré-definidas, modeladas através de um método conhecido como método do *elemento de contorno* (LI; GUPTA, 2005).

Na comunidade de robótica, pesquisadores têm focado seus eforços no controle de grupos de robôs. Uma forma bastante utilizada para garantir a coerência do grupo é através de robôs que se movem em formação. Por exemplo, Balch e Hybinette (BALCH; HYBINETTE, 2000) propuseram o uso de potenciais sociais para gerar formações escaláveis para grupos de robôs móveis. Eles definiram algumas forças para atuar nos robôs, para que estes sejam guiados e evitem a colisão com obstáculos. Essas forças funcionariam como lugares "atratores" que garantiriam que os robôs mantivessem a formação. A Figura 2.13 mostra os resultados obtidos por Balch e Hybinette em uma simulação.

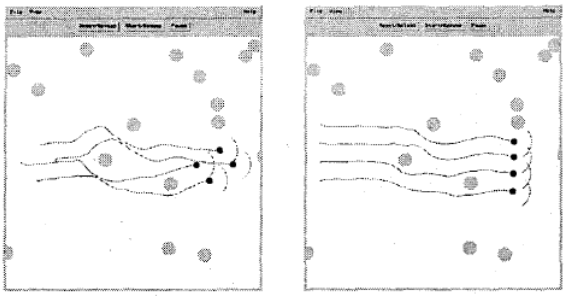


Figura 2.13: Técnica proposta por Balch e Hybinette (BALCH; HYBINETTE, 2000). As círculos cinza claro representam os obstáculos, enquanto os círculos pretos representam os robôs, mantendo a formação.

Schneider e Wildermuth (SCHNEIDER; WILDERMUTH, 2003) propuseram uma abordagem baseada em campos potenciais para lidar com o movimento coordenado de um grupo de robôs em formação. Cada robô estaria sujeito à diferentes *forças virtuais* pertencentes aos outros robôs do mesmo grupo. Os robôs estariam sujeitos também a forças emitidas pelos obstáculos. Estas forças, quando combinadas, serviriam para mover cada robô para sua posição desejada, mantendo uma formação fixa.

Utilizando sensores locais, Fredslund e Mataric (FREDSLUND; MATARIC, 2002) apresentaram uma técnica para a obtenção de um comportamento global baseado em uma

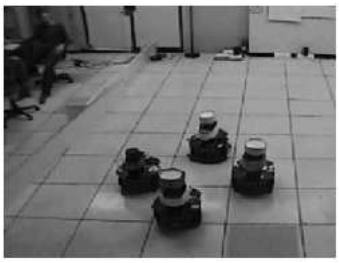


Figura 2.14: Robôs navegando utilizando a técnica proposta por Fredslund e Mataric (FREDSLUND; MATARIC, 2002).

comunicação mínima entre os robôs de um grupo para se alcançar um objetivo global. Esse método possibilita que um robô siga um "amigo" previamente estabelecido e em uma orientação específica, e compartilhe informações através de seus sensores. A Figura 2.14 mostra um robô guiado por essa técnica.

Este trabalho propõe uma nova técnica (vide Seção 5) para o movimento de agentes em ambientes virtuais que garante a coesão.

2.3 Considerações sobre Controle de Grupos

A Tabela 2.1 abaixo compara as técnicas tradicionais mencionadas anteriormente com a técnica proposta neste trabalho. Os elementos de comparação estabelecidos são: a possibilidade de utilizar formação, a possibilidade de manter a coesão do grupo durante a navegação, a presença de um movimento natural em passagens estreitas e o desempenho. Percebe-se que a técnica aqui proposta reúne características que as técnicas existentes não possuem, sem a necessidade de utilizar "remendos" (POTTINGER, 1999a), como a maioria das técnicas utilizadas em jogos e ambientes virtuais, e ainda possibilita que algumas características sejam configuradas (habilitadas ou não) de acordo com a necessidade.

Tabela 2.1: *Comparação entre as Técnicas*

	Formação	Coesão	Passagem Estreitas	Desempenho
Pottinger (POTTINGER, 1999b)	sempre	não	natural	ótimo
Balch e Hybinette (BALCH; HYBINETTE, 2000)	sempre	não	sem informação	ótimo
Li e Chou (LI; CHOU, 2003)	sempre	não	não-natural	ótimo
Kamphuis (KAMPHUIS; OVERMARS, 2004b)	não	sempre	não-natural	ótimo
Berg et al. (BERG et al., 2008)	não	não	não-natural	bom
Silveira et al.	sim(configurável)	sim(configurável)	natural	bom

3 PLANEJAMENTO DE CAMINHOS UTILIZANDO FUNÇÕES HARMÔNICAS

A navegação de um humano virtual, um robô ou um ator sintético é um processo que normalmente se dá em duas etapas. Na primeira etapa, faz-se um planejamento para determinar um caminho; e na segunda etapa, o agente segue o caminho determinado na primeira etapa. A primeira etapa lida com a combinação de conceitos como eficiência, tratamento de riscos, computabilidade, entre outros. A segunda etapa possui uma série de rotinas ou correções que os agentes devem executar para adaptar seus movimentos quando os caminhos previamente definidos não puderem ser seguidos devido às mudanças imprevistas no ambiente, ou à necessidade de suavização do caminho gerado.

Como comentado anteriormente, dependendo do método de campos potenciais utilizado, o robô ou agente pode se deparar com mínimos locais. Uma maneira de produzir um campo potencial livre de mínimos locais é através da solução numérica de uma equação diferencial parcial apropriada com condições de contorno convenientes, ou seja, um problema de valor de contorno (BVP) como o apresentado na seção seguinte.

3.1 Planejador de Caminhos baseado em Problema de Valores de Contorno

Nesta seção, é apresentado um método de planejamento de caminho baseado em problemas de valor de contorno, inicialmente proposto por Prestes (PRESTES, 2003) e adaptado por Dapper (DAPPER, 2007). A primeira proposta para o planejamento de caminho através de BVP foi feita por Connolly et al. (CONNOLLY; GRUPEN, 1993) através do uso de funções harmônicas. Por definição, uma função harmônica $p(\mathbf{r})$ em um domínio $\Omega \subset \Re^n$, é uma função que satisfaz a equação de Laplace:

$$\nabla^2 \, p(\mathbf{r}) \; = \; \sum_i \; \frac{\partial^2 p(\mathbf{r})}{\partial x_i^2} \; = \; 0 \tag{3.1}$$

onde $\mathbf{r}$ são as coordenadas do ambiente.

A equação de Laplace é importante em muitas áreas da ciência, tais como gravitação, eletro-magnetismo e dinâmica de fluídos, pois descreve campos de forças usados em cada uma dessas áreas. Na expressão matemática dos campos de forças, aparecem superfícies equipotenciais, definidas pela equação de Laplace. Linhas de fluxo, normais a essas superfícies ou linhas, definem caminhos no qual uma partícula, massa, ou carga elétrica move-se, "caindo" de um potencial maior, para um menor.

Recentemente, Trevisan (TREVISAN et al., 2006) propôs um *framework* para navegação exploratória baseado em uma família de funções potenciais que não possuem

mínimos locais, chamado planejador BVP. Essa família de funções é gerada através da solução numérica de um problema de valor de contorno usando as condições de contorno de Dirichlet e a seguinte equação:

$$\nabla^2 \, p(\mathbf{r}) + \epsilon \mathbf{v}.\nabla p(\mathbf{r}) \; = \; 0 \tag{3.2}$$

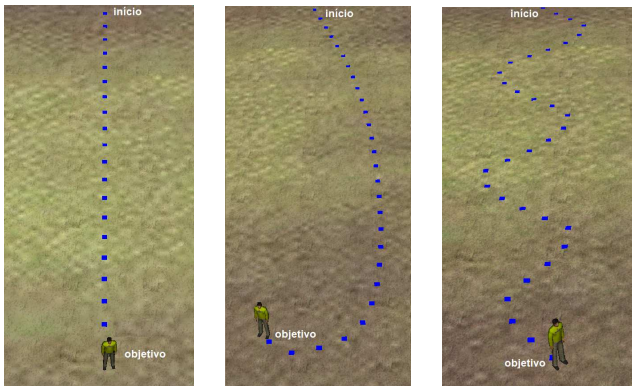
onde $\mathbf{v}$ é um vetor unitário e ϵ é um valor escalar. Em tarefas exploratórias, o uso dos termos ϵ e $\mathbf{v}$ ajuda o robô a realizar uma exploração eficiente minimizando o tempo necessário para adquirir informações em ambientes esparsos. Isso é feito distorcendo o campo potencial através dos parâmetros ϵ e $\mathbf{v}$ que indicam ao robô uma direção preferencial para a exploração. Este planejador produz caminhos que minimizam a probabilidade de colisão do agente com os obstáculos, isto é, o agente irá seguir um caminho equidistante das paredes, como mostrado na Figura 3.1(a).

Em tarefas de navegação, a distorção produzida pelo uso dos parâmetros especificados é empregada para gerar comportamentos diferentes e naturais para humanóides virtuais, representados através do caminho a ser seguido por cada agente durante tarefas de navegação (DAPPER et al., 2006; DAPPER; PRESTES; NEDEL, 2007). Nesse contexto, o vetor $\mathbf{v}$ chamado *vetor comportamental* pode ser visto como uma força externa que puxa o agente para sua direção sempre que possível, enquanto o parâmetro ϵ pode ser entendido como a *magnitude* ou *influência* desse vetor no comportamento do agente. Os valores permitidos para os parâmetros ϵ e $\mathbf{v}$ possibilitam gerar uma quantidade expressiva de seqüências de ações que humanóides virtuais podem considerar durante a navegação para alcançar um alvo, como pode ser visto na Figura 3.1 (b) e (c). Além do mais, eles podem ser utilizados para individualizar o comportamento dos agentes, simulando características de personalidades ou motivações internas, como raiva e pressa, entre outras.

Duas seqüências de ações não são estaticamente definidas pelo mesmo par ϵ e $\mathbf{v}$. Elas variam de acordo com as informações obtidas pelo agente, permitindo-o reagir dinamicamente contra eventos inesperados, como obstáculos dinâmicos. Além disso, esse par não está restrito a permanecer constante durante a execução das tarefas. Eles podem variar à medida que os agentes se deslocam no ambiente, para obter um comportamento desejado. A Figura 3.1(c) mostra uma situação onde o vetor de comportamento $\mathbf{v}$ varia de acordo com uma função senoidal.

Quando $\epsilon = 0$, a Equação 3.2 se reduz a $\nabla^2 \; p(\mathbf{r}) = 0$ o que corresponde à equação de Laplace e tem-se o planejador baseado em funções harmônicas. O comportamento produzido por esse planejador é bastante estereotipado, e nem sempre é adequado para simular movimentos de humanoides virtuais.

A abordagem utilizada para resolver numericamente um BVP é considerar que o espaço da solução está discretizado em uma grade regular com células do mesmo tamanho. Cada célula (i, j) está associada a uma região quadrada do ambiente real, de tamanho unitário, e armazena um valor potencial $p^t_{i,j}$ no instante t. As condições de contorno de Dirichlet associam às células que contêm obstáculos no mundo real um valor de alto potencial, enquanto as células que contêm o objetivo armazenam um valor de baixo potencial. O alto valor do potencial evita que o agente vá em direção aos obstáculos enquanto o baixo valor do potencial gera uma força de atração que "puxa" os agentes. O potencial das outras células é computado usando o método de relaxamento de Gauss-Seidel. O método de Gauss-Seidel consiste em substituir o valor do potencial de cada célula livre pela média simples de seus vizinhos simultaneamente. Porém, são utilizados vizinhos pertencentes a iterações diferentes. Outros métodos de relaxamento existentes, como SOR e ADI, podem acelerar o processo de relaxamento, porém, Prestes (PRESTES, 2003) mostrou que



(a) (b) (c)

Figura 3.1: Diferentes caminhos seguidos por agentes usando a Equação 3.2: (a) caminho produzido por potenciais harmônicos, isto é, com $\epsilon = 0$; (b) com $\epsilon = -1.0$ e $\mathbf{v} = (1, 0)$; (c) com $\epsilon = -1.0$ e $\mathbf{v} = (1, \sin(0.6t))$

.

se o relaxamento parcial for utilizado para gerar os comportamentos computados, então o método de Gauss-Seidel deve ser usado devido à estabilidade dos resultados.

Considerando a Equação 3.2 discretizada, e considerando o tamanho da discretização como 1 unidade, o potencial das células livres é atualizado através da seguinte equação:

$$p_c = \frac{p_b + p_t + p_r + p_l}{4} + \frac{\epsilon((p_r - p_l)v_x + (p_b - p_t)v_y)}{8} \tag{3.3}$$

onde $p_c = p_{i,j}^{t+1}$, $p_b = p_{i,j+1}^{t}$, $p_t = p_{i,j-1^{t+1}}$, $p_r = p_{i+1,j}^{t}$, $p_l = p_{i-1,j}^{t+1}$ e $\mathbf{v} = (v_x, v_y)$. Estas células podem ser vistas na Figura 3.2.

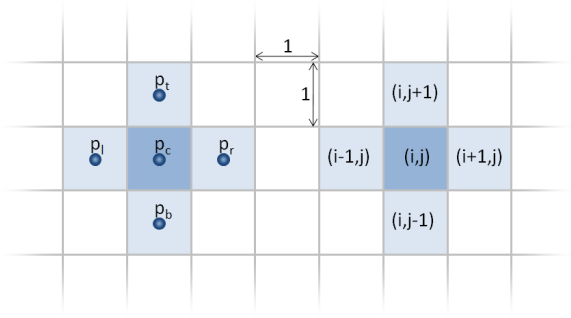


Figura 3.2: Localização das células em uma grade regular.

O parâmetro ϵ deve estar no intervalo $(-2, 2)$, pois valores fora desse intervalo geram oscilações e caminhos instáveis que não garantem que o agente irá alcançar o objetivo ou irá evitar colisões. Isso acontece porque as condições de contorno, que asseguram que o agente é repelido pelos obstáculos e atraído pelos alvos, são violadas. Para esclarecer melhor isso, pode-se reescrever a Equação 3.3 como

$$p_c = \frac{p_l(1 - w_x) + p_t(1 - w_y) + p_r(1 + w_x) + p_b(1 + w_y)}{4} \tag{3.4}$$

onde $w_x = \epsilon v_x/2$ e $w_y = \epsilon v_y/2$. Quando $w_x, w_y \in [-1, 1]$, tem-se uma média ponderada e $p_{min} \leq p_c \leq p_{max}$, onde p_{min} e p_{max} são os valores mínimos e máximos de p em sua

vizinhança. Logo, existe sempre um valor próximo de p menor ou maior que p_c. Como o obstáculo e o alvo possuem os valores p_{max} e p_{min}, respectivamente, o gradiente em qualquer célula do espaço-livre irá apontar para a região de menor valor de potencial longe dos obstáculos. Em outras palavras, o agente sempre será repelido pelos obstáculos e atraído pelo alvo.

Quando w_x e w_y estão fora do intervalo permitido, o pressuposto de que os obstáculos irão repelir o agente não é mais garantido. Considere as células associadas ao potencial p_l, p_t e p_r tendo obstáculos no ambiente real enquanto p_b é atribuído às células no espaço-livre. De acordo com as condições de contorno de Dirichlet, $p_l = 1$, $p_t = 1$, $p_r = 1$ e $0 \leq p_b \leq 1$. Substituindo-se valores na Equação 3.4, tem-se:

$$p_c = \frac{1}{4}(3 + p_b + p_b w_y - w_y)$$

Dependendo dos valores de p_b e w_y, p_c pode estar fora do intervalo $[0, 1]$. A Figura 3.3 mostra o valor p_c para diferentes pares (p_b, w_y). Em vários casos, p_c é maior que p_{max}, e portanto, o gradiente nessa célula apontará na direção de um obstáculo.

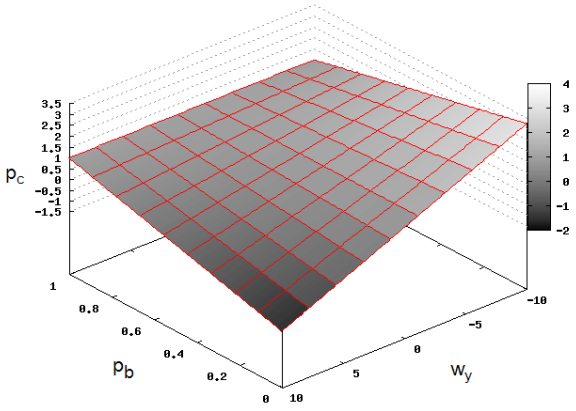


Figura 3.3: Valores de p_c obtidos variando w_y e p_b.

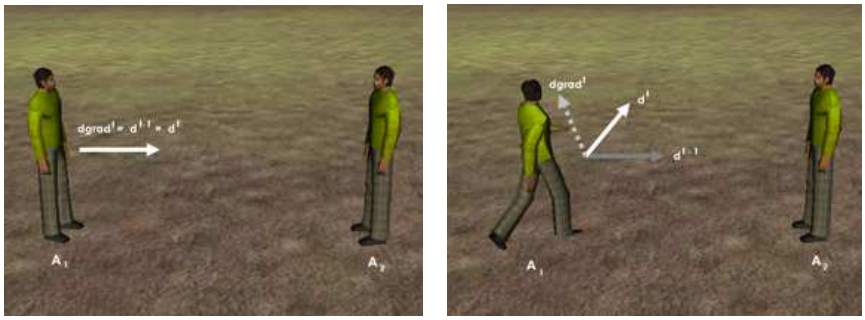
3.1.1 Movimento do Agente - Definindo Movimentos Realísticos e Naturais

Após a computação do potencial, o agente se move seguindo a direção definida pelo gradiente descendente de seu potencial na atual posição (i, j), definido por

$$(\nabla \mathbf{p})_{(i,j)} = \left(\frac{p_{i+1,j} - p_{i-1,j}}{2}, \frac{p_{i,j+1} - p_{i,j-1}}{2} \right)$$

e considerando o tamanho da discretização sendo 1 unidade.

Como o método é formalmente completo, se existir um caminho em direção ao objetivo, ele será encontrado. Esse processo é uma forma intuitiva de controlar o movimento do agente. Contudo, ele pode não produzir comportamentos realísticos, como os observados no mundo real. Uma das razões para isso acontecer, é que o agente muda a sua direção baseado apenas no gradiente descendente de sua região. Como exemplo, se o campo de visão do agente for pequeno, o seu tempo de reação será muito curto para reagir a obstáculos dinâmicos. Então, esses obstáculos produzirão uma forte força de repulsão que



(a) (b)

Figura 3.4: Definindo o movimento do agente A_2. (a) situação antes do agente A_2 entrar no campo de visão de A_1. (b) Se o agente A_1 seguir na direção definida pelo gradiente descendente (*dgrad*), ele irá mudar sua direção em quase $\pi/2$, o que é indesejável. Contudo, se o agente utiliza um vetor d, com orientação φ^t, sua trajetória será uma curva suave, o que é mais realística e natural.

mudará bruscamente a direção do agente. Como pode ser visto na Figura 3.4, se for utilizado somente o gradiente descendente (*dgrad*), o agente mudará a sua direção em torno de $\pi/2$ rad.

Dapper et al. (DAPPER; PRESTES; NEDEL, 2007) propuseram a solução para esse problema ajustando a posição atual do agente de

$$\Delta\ \mathbf{d} = \upsilon(\cos(\varphi^t), \sin(\varphi^t)) \tag{3.5}$$

onde υ define a velocidade máxima do agente e φ^t é

$$\varphi^t = \eta\ \varphi^{t-1} + (1-\eta)\ \zeta^t$$

com $\eta \in [0, 1)$ e ζ representando a orientação do gradiente descendente na posição atual.

Quando $\eta = 0$, o agente ajusta sua orientação usando somente a informação do gradiente. Se $\eta = 0.5$, a direção anterior do agente (φ^{t-1}) e a direção do gradiente descendente influenciam igualmente o cálculo da nova direção do agente. A Figura 3.4 (b) mostra o vetor $\mathbf{d}^t$ com a orientação φ^t utilizando $\eta = 0.5$. O parâmetro η pode ser visto como um fator de inércia que tende a manter constante a direção do agente à medida que $\eta \to 1$. Quando $\eta \to 1$, o agente reage lentamente aos eventos inesperados, aumentando a probabilidade de colisão com os obstáculos.

Apesar da Equação 3.5 produzir bons caminhos suaves em ambientes com poucos obstáculos, quando um ambiente possui muitos obstáculos espalhados, o comportamento do agente não é realístico, e colisões podem ocorrer. Para resolver esse problema, foi incorporado um controle de velocidade, utilizando a equação

$$\Delta\ \mathbf{d} = \upsilon\ (\cos(\varphi^t), \sin(\varphi^t))\ \Psi(|\varphi^{t-1} - \zeta^t|) \tag{3.6}$$

onde a função $\Psi : \Re \to \Re$ é

$$\Psi(x) = \begin{cases} 0 & \text{if } x > \pi/2 \\ \cos(x) & \text{caso contrário} \end{cases} .$$

Se $|\varphi^{t-1} - \zeta^t|$ for maior do que $\pi/2$, então existe uma chance de colisão, e a função retorna 0, fazendo com que o agente pare. Caso contrário, a velocidade do agente muda

proporcionalmente ao risco de colisão. Em regiões com muitos obstáculos, os agentes tendem a mover-se lentamente. Se um agente se depara com outro, um deles irá parar e esperar até que o outro passe. Além do mais, o controle da velocidade permite a simulação do comportamento dos agentes através da variação da magnitude da velocidade, ou seja, é possível simular um agente chateado fazendo-o mover-se lentamente, e um agente que está ansioso, fazendo-o mover-se rapidamente.

3.1.2 Visão Geral da Arquitetura

Como discutido na última seção, o planejador BVP é o núcleo da arquitetura proposta por Dapper et al. (DAPPER, 2007). Essa arquitetura requer a discretização do ambiente em uma grade regular, na forma de um mapa global e um conjunto de mapas locais, cada um associado a um agente diferente.

3.1.2.1 Mapa Global do Ambiente

O ambiente é representado por um conjunto de malhas homogêneas, $\{\mathcal{M}_k\}$, onde cada malha $\mathcal{M}_k$ possui $L_x \times L_y$ células, denotadas por $\{\mathcal{C}^k_{i,j}\}$. Cada célula $\mathcal{C}^k_{i,j}$ corresponde a uma região quadrada centrada nas coordenadas do ambiente $r = (r_i, r_j)$ e armazena um valor de potencial $\mathcal{P}^k_{i,j}$. O potencial associado à malha $\mathcal{M}_k$ é computado através das funções harmônicas (CONNOLLY; GRUPEN, 1993) e é utilizado pelos agentes para alcançar o alvo $\mathcal{O}_k$. Para delimitar o espaço de navegação dos agentes, foi considerado que o ambiente é envolvido por obstáculos.

3.1.2.2 Mapa Local de um Agente

Cada agente a_k possui um mapa m_k que armazena a informação local atual sobre o ambiente obtido por seus sensores. Esse mapa é centrado na posição atual do agente e representa uma pequena fração do mapa global, tipicamente próximo de 10% da área total coberta pelos mapas globais.

O mapa m_k possui $l^k_x \times l^k_y$ células, denotadas por $\{c^k_{i,j}\}$, onde cada célula corresponde a uma região quadrada centrada nas coordenadas do ambiente $r = (r_i, r_j)$ e armazena um valor potencial igual a $p^k_{i,j}$. Essas são divididas em três regiões: a zona de atualização (*u-zone*); a zona livre (*f-zone*) e a zona de borda (*b-zone*), como mostrado na Figura 3.5.

A área associada a cada célula do mapa local é menor que a área associada a uma célula no mapa global. Isso produz um movimento refinado. Quanto menor o tamanho de uma célula, melhor a qualidade do movimento. Enquanto isso, o mapa global é utilizado somente para permitir a navegação do agente em todo o cenário.

3.1.2.3 Atualizando os Mapas Locais e Globais

Para cada agente a_k, um objetivo $\mathcal{O}_{objetivo(k)}$[1], um valor particular $\mathbf{v}_k$ que controla o seu comportamento e um ϵ_k devem ser definidos. Um objetivo e um mesmo par $\mathbf{v}$ e ϵ podem ser associados a vários agentes. Se $\mathbf{v}_k$ ou ϵ_k forem dinâmicos, então a função que os controla deve ser especificada.

Para navegar no ambiente, um agente a_k utiliza os seus sensores percebendo o mundo e atualizando o seu mapa local com a informação dos obstáculos e dos outros agentes. O sensor do agente estabelece um cone de visão com abertura α. A Figura 3.5 mostra um esboço do mapa local de um agente. As células $c^k_{i,j}$ na região *u-zone* que estão dentro do cone e correspondem a obstáculos a ou outros agentes, possuem o seu valor potencial

[1] A função *objetivo()* mapeia o agente k em seu objetivo atual

igual a 1. Na Figura 3.6, como o agente 1 está dentro da região *u-zone* do mapa local do agente 2, mas fora de seu cone de visão, não é mapeado como obstáculo no mapa local do agente 2. Esse procedimento assegura que obstáculos dinâmicos ou estáticos que estão atrás do agente não interferirão em seu movimento futuro. Para cada agente a_k, o gra-

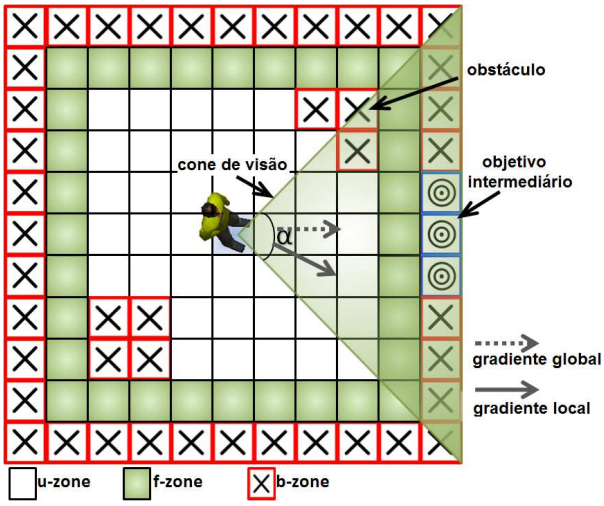


Figura 3.5: Mapa local do agente. Células brancas, verde-claras e vermelhas compreendem a zona de atualização, livre e de bordas, respectivamente. Células vermelhas, azuis e a célula central correspondem aos obstáculos, ao objetivo intermediário e a posição do agente, respectivamente.

diente descendente global na célula no mapa global $\mathcal{M}_{objetivo(k)}$ que contém sua posição atual é calculada. A direção do gradiente é utilizada para gerar um objetivo intermediário na borda do mapa local, estabelecendo valores de potencial igual a 0 para algumas células na região *b-zone*, enquanto outras células na região *b-zone* são consideradas como obstáculos, com seu valor potencial igual a 1. Na Figura 3.6, cada agente calcula seu gradiente global para projetar um objetivo intermediário em seu mapa local. Como o mapa local do agente é delimitado por obstáculos, o agente é "puxado" na direção do objetivo intermediário usando a direção oposta ao seu gradiente. O objetivo intermediário auxilia o agente a_k a alcançar seu alvo $\mathcal{O}_{objetivo(k)}$ enquanto permite produzir um movimento particular. Em alguns casos, o objetivo $\mathcal{O}_{objetivo(k)}$ está dentro do cone de visão e dentro da região *u-zone*, e conseqüentemente, as células do mapa local ocupadas pelo objetivo são atualizadas com o valor potencial 0. O objetivo intermediário é sempre projetado, mesmo quando o $\mathcal{O}_{objetivo(k)}$ for mapeado na região *u-zone*. Caso contrário, o agente poderá facilmente ficar preso porque levaria em consideração somente as informações locais sobre o ambiente, da mesma maneira que os campos potenciais tradicionais (KHATIB, 1980).

As células pertencentes à região *f-zone* são sempre consideradas livres de obstáculos, mesmo quando existem obstáculos dentro dessa região. A ausência dessa região pode fazer com que a conexão entre a célula em que o agente se encontra e o objetivo intermediário devido ao mapeamento de obstáculos na frente do objetivo intermediário seja perdida. Quando isso ocorre, o agente fica perdido porque não existe informação vinda do objetivo intermediário para produzir um caminho para alcançá-lo. As células da região *f-zone* resolvem esse problema, sempre permitindo a propagação da informação sobre o objetivo para as células onde o agente se encontra.

Após coletar as informações e fazer o mapeamento, o agente k atualiza o valor do

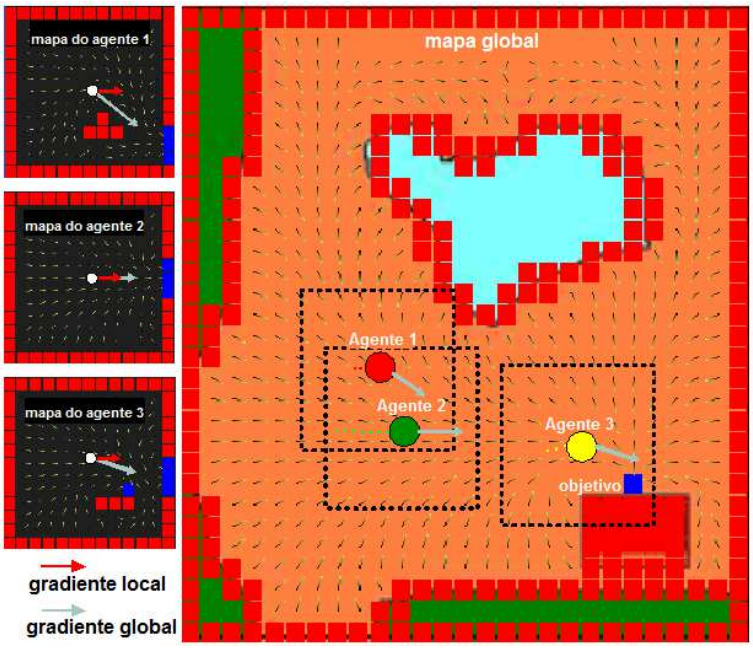


Figura 3.6: Visão geral da implementação de Dapper et al. (DAPPER et al., 2006)

potencial das suas células utilizando a Equação 3.3 com seu par $\mathbf{v}^k$ e ϵ^k. A seguir, a posição do agente é atualizada de acordo com a Equação 3.6 utilizando o gradiente descendente computado através do campo potencial armazenado em seu mapa local na posição $p_x = \lceil l_x^k/2 \rceil$ e $p_y = \lceil l_y^k/2 \rceil$.

3.1.3 Algoritmo

O algoritmo que implementa os conceitos apresentados na seção anterior está descrito a seguir, no Algoritmo 1.

Os dois passos iniciais são executados em uma fase de pré-processamento. Em relação ao segundo laço, cada agente executa independentemente, e de forma não sincronizada, as ações de 6 a 11. Esse algoritmo considera que cada agente deve alcançar apenas um objetivo. Contudo, o agente pode ser encarregado de alcançar vários objetivos enumerados. Nesse caso, o passo 11 deve ser alterado pelo Algoritmo 2.

Algorithm 1 Planejador de Caminho BVP

1: Computar todos os mapas globais do ambiente.{*um para cada objetivo* o_k *existente.*}

2: **for** cada agente a_k **do**
3: Definir o vetor comportamental $\mathbf{v}_k$ e ϵ_k.{*Cada variável pode ser estática ou dinâmica. Se uma variável for dinâmica, então a função que a controla deve ser especificada.*}
4: **end for**
5: **for** cada agente a_k **do**
6: Obter a informação do ambiente, detectando obstáculos estáticos e dinâmicos.
7: Atualizar seu mapa local com informação sobre os obstáculos e outros agentes.
8: Computar o gradiente descendente global e gerar o objetivo intermediário.
9: Atualizar o campo potencial.
10: Computar o gradiente local e seguir o sentido oposto de acordo com a equação 3.6.

11: **while** não alcançar o objetivo $o_{objetivo(k)}$. **do**
12: Repetir os passos de 6 a 11.
13: **end while**
14: Parar de mover.
15: **end for**

Algorithm 2

while não alcançar o objetivo $o_{objetivo^i(k)}$. **do**
Repetir os passos de 6 a 11 do Algoritmo 1.
end while
if $objetivo^i(k) = objetivo^{ultimo}(k)$ **then**
Parar de mover.
else
repetir o processo com o próximo objetivo $o_{objetivo^{i+1}(k)}$
end if

4 EXTENSÕES

Um dos objetivos iniciais deste trabalho foi resolver algumas limitações da técnica apresentada na seção anterior, como o alto custo computacional, propondo algumas melhorias.

Para que a técnica pudesse ser aplicada de forma efetiva em aplicações em tempo-real, procurou-se estudar detalhes sobre a etapa de relaxamento, e propor maneiras de acelerá-la.

Através de experimentos realizados, percebeu-se que os agentes poderiam exibir comportamentos interessantes, através da maneira como os agentes interagem com o ambiente. Essas situações foram melhor exploradas e são discutidas nas subseções seguintes.

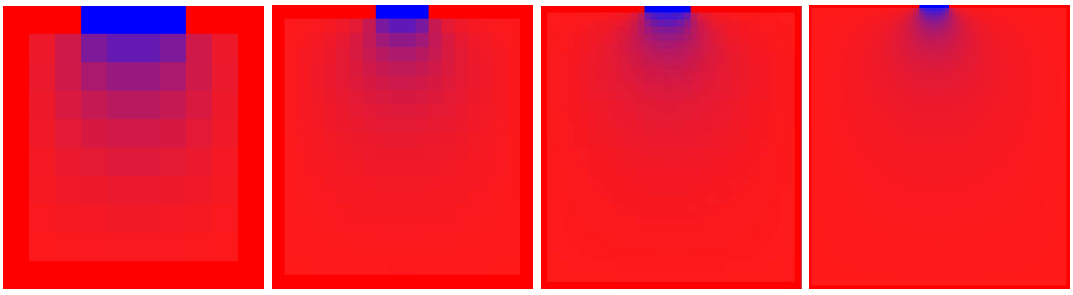
4.1 Acelerando o Relaxamento

Prestes (PRESTES, 2003) mostrou que, apesar do método SOR convergir mais rapidamente do que o método de Gauss-Seidel, para um número pequeno de iterações, o método de Gauss-Seidel gera trajetórias mais suaves do que o SOR em tarefas de exploração. Utilizar apenas resultados parciais para a equação de Laplace é importante para se conseguir controlar vários agentes em tempo real. Também é importante em ambientes dinâmicos, onde o espaço de configurações muda constantemente (sendo alteradas as condições de contorno e um novo cálculo para o campo potencial precisa ser realizado). Percebeu-se que a maior parte de todo o processamento é gasta com a etapa de relaxamento, ou seja, com a solução da Equação 3.3.

Com a finalidade de diminuir a quantidade de iterações necessárias ao relaxamento dos mapas locais de cada agente, foram analisadas algumas maneiras diferentes de atualizar os mapas locais. A etapa de relaxamento feita por Dapper et al. (DAPPER, 2007) consistia em iniciar o mapa com potenciais iguais a 1 nas células marcadas como obstáculos, potenciais iguais a 0 nas células marcadas como objetivo, e potenciais escalares intermediários entre 0 e 1 nas células marcadas como livre. Dapper et al. (DAPPER, 2007) utilizavam sempre um valor constante 0.5 como potencial escalar nas células livres.

Durante a etapa de relaxamento, a Equação 3.3 é avaliada várias vezes para cada célula do mapa local de um agente, até a convergência. Após a convergência, conforme pode ser visto na Figura 4.1, os valores entre as células marcadas como objetivos e as células marcadas como obstáculos não crescem linearmente. Há uma grande quantidade de células vermelhas, o que indica que há uma grande quantidade de células com valores muito próximos de 1. Isso implica que a escolha do valor 0.5 como valor inicial das células livres não é uma boa escolha. Alguns experimentos foram realizados com a finalidade de obter valores que acelerassem a convergência.

Na Figura 4.2, pode-se observar a quantidade de iterações gastas para realizar o rela-



(a) (b) (c) (d)

Figura 4.1: Mapas após o relaxamento. A cor vermelha indica valores de potenciais escalares próximos de 1 (obstáculo), enquanto valores azuis indicam valores de potenciais escalares próximos de 0 (objetivo). Pode-se perceber que os valores escalares não crescem linearmente. Em (a) é utilizado uma mapa com dimensões 10×10, em (b) 20×20, em (c) 40×40 e em (d) 80×80.

xamento de um mapa cujas células livres inicializam com determinados valores escalares, ou com valores escalares aleatórios, em um determinado intervalo. Percebe-se que para mapas pequenos (até 25×25 células), inicializar as células livres com valores próximos de 0.9 é uma boa escolha, e acelera a convergência. Para mapas grandes (maiores que 50×50 células), inicializar as células livres com valores bem próximos de 1 acelera de forma significativa a convergência. Percebe-se também que inicializar as células livres com valores aleatórios ao invés de valores escalares iguais não gera benefício significativo, mesmo quando os números aleatórios pertencem a algum intervalo bem definido.

A etapa de relaxamento através do algoritmo de Gauss-Seidel é um processo em que a cada iteração o potencial aproxima-se do potencial esperado após a convergência. Portanto, quanto mais próximos os potenciais escalares de cada célula estão dos potenciais esperados após a convergência, mais rápidamente se dá o processo de relaxamento. Sabe-se que, a cada passo do agente, o espaço de configuração é basicamente o mesmo, ou seja, são feitas poucas mudanças no mapa local. Os obstáculos presenciados pelo agente ao se mover são mapeados em uma posição ligeiramente deslocada em relação a sua posição anterior. O processo de convergência do potencial é acelerado devido aos valores dos potenciais na maioria das células já estarem próximos dos valores esperados para a convergência com a nova configuração. Na Figura 4.3, pode-se ver uma situação onde são aproveitados os potenciais do passo anterior. Inicialmente, foram gastas 116 iterações para o relaxamento em um mapa com 20×20 células, considerando um erro de 10^{-3}. Mantendo a configuração do potencial anterior, foram gastas apenas 86 iterações para a convergência do relaxamento, uma redução neste caso de 26%.

Experimentos utilizando os valores presentes no *depth buffer*[1] também foram realizados. Todos os obstáculos presentes no mapa local de um agente foram renderizados e foi capturado o *depth buffer* desta cena. O processo do relaxamento atuava então, nos valores lidos do *depth buffer*. Os resultados encontrados não se mostraram vantajosos aos obtidos no experimento mencionado acima, e além disso, um tempo a mais era gasto com a leitura do *depth buffer*. Na Figura 4.4 pode-se ver como exemplo, uma cena renderizada, coberta

[1] *Depth buffer* é uma região de memória utilizada para armazenar os valores de profundidade de cada pixel renderizado.

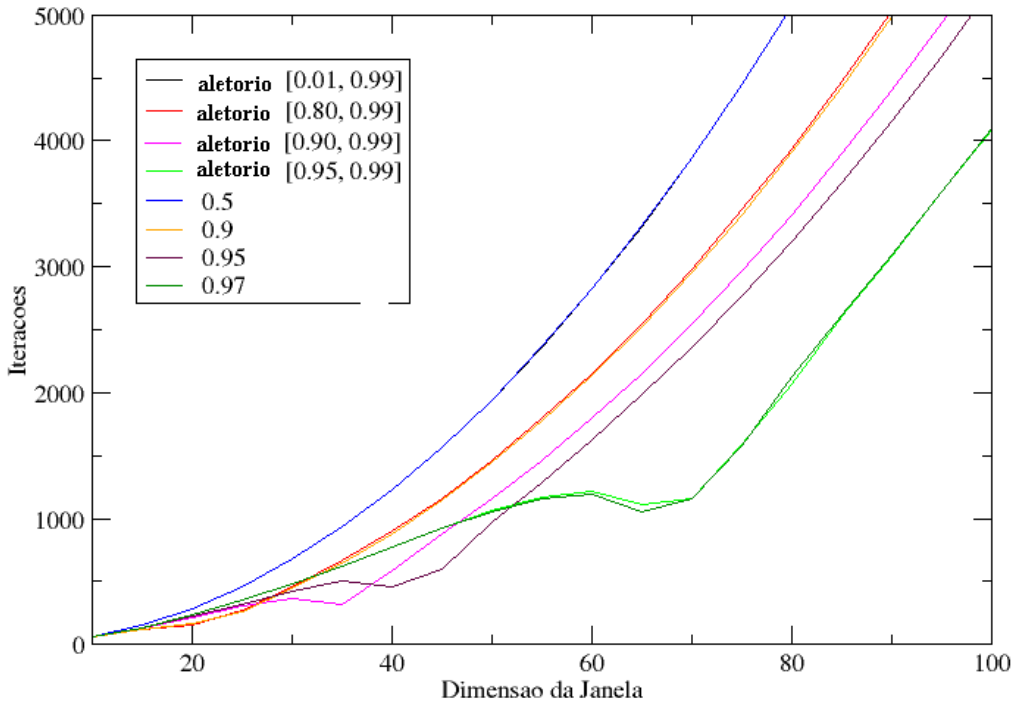


Figura 4.2: Comparação do relaxamento em um mapa inicializado com todos os potenciais escalares iguais, e mapas inicializados com potenciais escalares aleatórios.

de obstáculos, e o seu *depth buffer*.

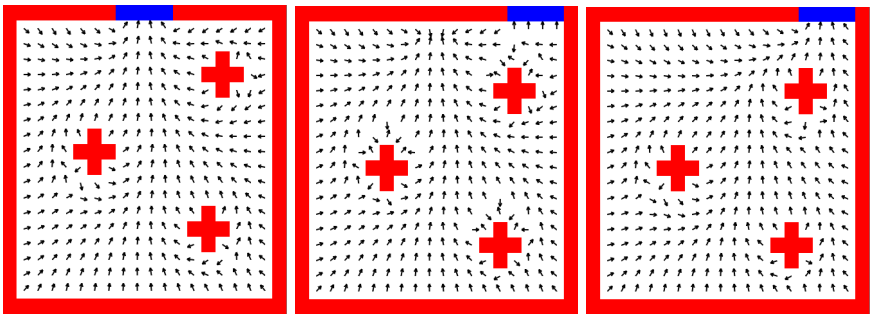
Ainda com o intuito de minimizar o tempo gasto com o processo de relaxamento, e levando em conta sua natureza altamente paralelizável, foi implementado o relaxamento na GPU. O relaxamento foi implementado usando o algoritmo Gauss-Seidel na GPU e comparado com a implementação na CPU. A Figura 4.5 ilustra os resultados obtidos em um ATHLON 2.2 GHz com 2 GB de memória RAM e uma placa gráfica GeForce 8800 GTX com 768 MB de memória. O relaxamento foi feito até a convergência, com um erro da ordem de 10^{-3}. Pode-se perceber com esta comparação que o processo de relaxamento paralelizado nos dá uma escalabilidade linear.

Percebe-se então que a forma utilizada no trabalho de Dapper et al. (DAPPER, 2007) para lidar com os mapas dos agentes não era a melhor forma possível, gerando margem para otimizações significativas, visto que o gargalo da técnica é justamente a etapa da convergência do relaxamento. A Seção 6.8 apresenta os resultados comparativos mostrando que a atribuição prévia de um alto valor de potencial às células livres do ambiente leva a um aumento significativo no desempenho da estratégia proposta por Dapper.

4.2 Definindo Comportamentos Através da Janela Local

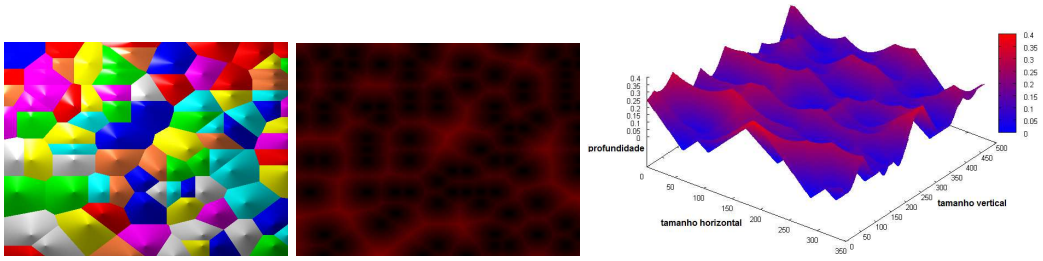
Resultados interessantes podem ser produzidos na forma como o agente interage com outros agentes em um ambiente, simplesmente variando o tamanho de seu mapa local. Quanto mais informação sobre o ambiente estiver disponível, mais o agente tenderá a mudar seu comportamento para evitar regiões com uma grande concentração de obstáculos.

A Figura 4.6 mostra duas situações onde o agente encontra um grupo em seu caminho.



(a) (b) (c)

Figura 4.3: (a) Mapa inicial. O agente segue na direção indicada pelo objetivo (células azuis). Foram gastas 116 iterações para o relaxamento com um erro de 10^{-3} unidades. (b) O agente caminha um passo na direção $(0, 1)$, o que provoca um deslocamento dos obstáculos, e conseqüentemente, uma mudança na posição dos objetivos. Somente as células modificadas são atualizadas e os potenciais das células não atualizadas são aproveitados. (c) Após o relaxamento são necessárias apenas 86 iterações. Ambos os mapas possuem 20×20 células.



(a) (b) (c)

Figura 4.4: (a) Ambiente com obstáculos dispostos arbitrariamente e (b, c) seu *depth buffer*.

Ao se deparar com o grupo, existem duas possibilidades: o agente continuar o seu caminho, passar entre o grupo, ou evitar o grupo, desviando o seu percurso e o contornando. Esse comportamento pode ser visto frequentemente na vida real. Se alguém está com pressa, ou não está em um bom dia e avista um aglomerado de pessoas, a tendência dessa pessoa é desviar sua trajetória, evitando uma interação com o grupo. Caso contrário, essa pessoa pode passar entre os indivíduos da multidão, interagindo melhor com o grupo. O comportamento de uma pessoa muda conforme seu estado emocional ou suas necessidades. No caso do agente, esse comportamento pode ser obtido variando o tamanho do mapa local.

Na Figura 4.6(a), o mapa local do agente possui 25×25 células, enquanto na Figura 4.6(b) o mapa local do agente possui 50×50 células. Cada célula possui 1×1 unidades do ambiente real. No primeiro caso, o agente passa entre os indivíduos do grupo, enquanto que no segundo caso, o agente evita o grupo, contornando-o. Baseado nesses experimentos, é possível notar que o tamanho do mapa exerce uma influência direta no modo como os agentes se comportam no ambiente.

Para explicar essa influência de forma objetiva, foram realizados experimentos variando parâmetros como o tamanho do mapa local, a densidade de obstáculos de uma

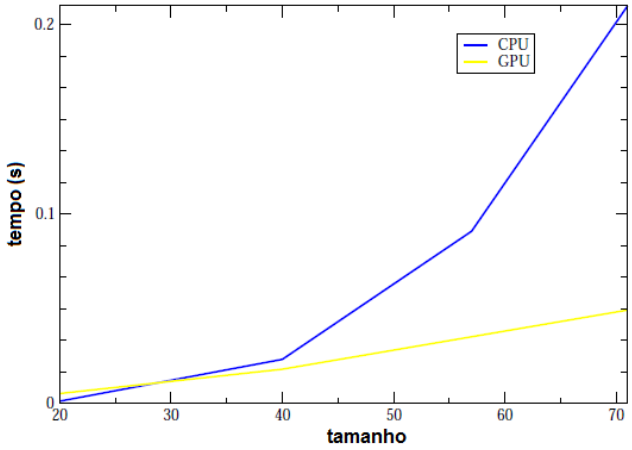
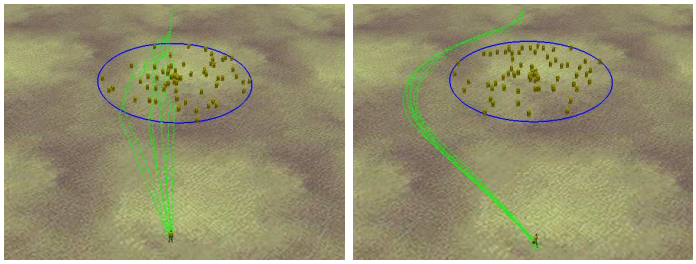


Figura 4.5: Comparação entre a implementação na CPU e GPU. A curva amarela mostra a escalabilidade linear da Equação 3.3. O mapa é relaxado permitindo um erro da ordem de 10^{-3}. Em sistemas em tempo-real, uma aproximação do gradiente pode ser utilizada, e conseqüentemente, uma menor quantidade de iterações. O tempo coletado na GPU inclui o tempo gasto com o envio e leitura dos dados na GPU.

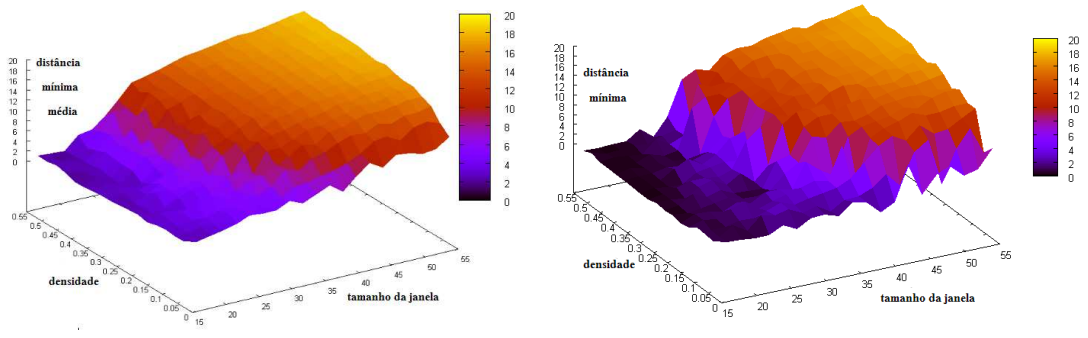


(a) (b)

Figura 4.6: (a) Mapa local com 25×25 células. O agente sempre caminha dentro da região que contém os obstáculos. (b) mapa local com 50×50 células. O agente sempre caminha fora da região composta por obstáculos. Em ambas situações, a densidade da região que contém os obstáculos é 1% e seu raio é 10 unidades.

região, e o tamanho da região de obstáculos, com o objetivo de extrair a relação entre esses parâmetros e identificar quais parâmetros farão com que o agente passe dentro ou fora da região de obstáculos. Em todos os experimentos, o agente sempre começa no mesmo local com a mesma orientação e segue até o objetivo, também fixo e sempre na mesma posição.

Foi assumido que o mapa local corresponde a uma região quadrada com o lado medindo L, onde $15 \leq L \leq 55$ células. Foi considerado também que o agente irá encontrar um grupo composto por outros agentes como obstáculos estáticos, distribuídos aleatoriamente em uma região circular de raio r, onde $5 \leq r \leq 30$ unidades. Esses obstáculos ocupam a região em um percentual γ que corresponde de 3% a 50% da região circular. Como para um mesmo valor de densidade podem haver diferentes disposições dos obstáculos, para cada tupla (r, L, γ) foram realizados 10 experimentos. Desses 10 experimentos, foi obtida a média das distâncias mínimas que o agente passa do centro da região de obstáculos. A Figura 4.7 mostra a relação entre os parâmetros para uma região de obstáculos com raio igual a 10 unidades. Nesse caso, pode-se observar que alguns pares



(a) (b)

Figura 4.7: Relação entre uma dimensão L do mapa local com a densidade γ de uma região com raio r igual a 10 unidades. $L \in [15, 55]$, $gamma \in [3\%, 51\%]$. Em (a) foi traçada a distância média após 10 experimentos, e em (b) foi traçada a distância mínima obtida em um experimento arbitrário.

(L, γ) fazem com que os agentes sempre passem dentro da região de obstáculos quando a média das distâncias mínimas é menor do que 10 (região azul).

Foi observado que para algumas tuplas (r, L, γ) o agente irá sempre passar dentro ou sempre passar fora da região de obstáculos. Existe uma região intermediária no gráfico onde os agentes, ora passam dentro, ora passam fora. A Figura 4.8 mostra os resultados apresentados na Figura 4.7 sob outra perspectiva. Nessa situação, foi particionado o espaço de parâmetros em três diferentes regiões, ρ_0, ρ_1 e ρ_2 de acordo com os valores da distância mínima produzida por cada par (L, γ). Se o par (L, γ) gerar, em todos os experimentos, uma distância mínima menor do que o raio r, esse par é classificado como um ponto pertencente à região ρ_0. Se o par gerar, em todos os experimentos, uma distância mínima maior do que o raio r, esse par é classificado como um ponto pertencente à região ρ_2. Caso contrário, o par pertence à região ρ_1, isto é, em alguns experimentos foram obtidas distâncias mínimas menor que o raio r, e em outros experimentos foram obtidas distâncias mínimas maiores do que r.

Um resultado interessante dessa classificação pode ser visto. A região intermediária possui a forma de uma função exponencial, definida pela tupla (L, γ, r)

$$\gamma = e^{-(L-\beta)/r} + \alpha \tag{4.1}$$

onde α e β são constantes.

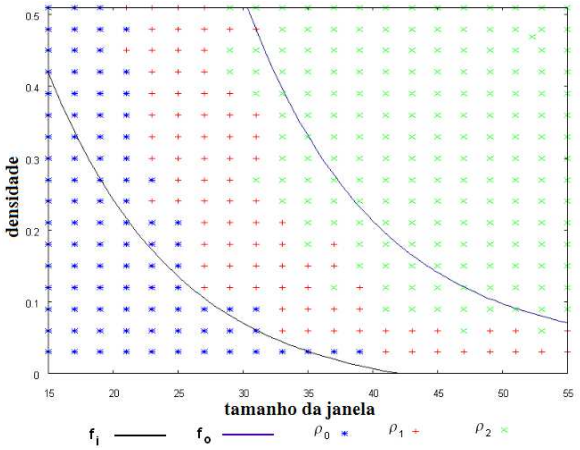
Utilizando uma função exponencial para fazer o ajuste da região, é encontrada uma expressão que relaciona a dimensão do mapa local, a densidade da região de obstáculos e o raio da região de obstáculos com o comportamento do agente. A relação entre esses valores é dada pela equação:

$$f(L, \gamma, r) = e^{-(L-\beta)/r} + \alpha - \gamma \tag{4.2}$$

onde os parâmetros e constantes são os mesmos da Equação 4.1. A partir daí, é possível classificar o espaço de parâmetros ρ_0, ρ_1 e ρ_2 através das seguintes funções:

$$f_i(L, \gamma, r) = e^{-(L-\beta_i)/r} + \alpha_i - \gamma$$

$$f_o(L, \gamma, r) = e^{-(L-\beta_o)/r} + \alpha_o - \gamma$$



(a)

Figura 4.8: (a) Ajuste com uma função exponencial da região de obstáculos com raio igual a 10 unidades. A região azul garante que o agente sempre passa dentro da região de obstáculos. A região verde garante que o agente sempre passa fora da região de obstáculos. A região vermelha define uma região onde não é possível prever o comportamento do agente.

onde $\beta_i < \beta_o$ e $\alpha_i = -\alpha_o$.

Para valores específicos de α e β, essas funções delimitam o espaço de parâmetros em regiões onde um agente irá ter um comportamento determinado, sempre passando fora da região de obstáculo, ou sempre passando dentro da região de obstáculos. Se $f_i(L, \gamma, r) < 0$ então a tupla (L, γ) pertence a ρ_0, isto é, o agente sempre passará na região entre os outros agentes. Se $f_o(L, \gamma, r) > 0$ então a tupla (L, γ) pertence a região ρ_2, isto é, o agente sempre passará na região ao redor dos outros agentes. Quando $f_i(L, \gamma, r) > 0$ e $f_o(L, \gamma, r) < 0$ não é possível predizer o comportamento do agente, ou seja, o agente pode ocasionalmente passar dentro ou fora da região de obstáculos, dependendo da disposição dos obstáculos. Portanto, o par (L, γ) pertence a ρ_1. A Figura 4.8 mostra um caso particular de uma região de obstáculos com raio igual a 10 unidades, e suas funções f_o e f_i, com $a_o = 0.3$, $b_o = 23$, $a_i = -0.3$ e $b_i = 7$.

Pode-se concluir então que, se existe um mapa local pequeno, $L \ll r$, o agente tende a passar dentro da região de obstáculos, a menos que a densidade da região de obstáculos seja grande. Se o agente possui um mapa local grande, $L \gg r$, ele considerará a região de obstáculos como um único obstáculo, passando fora dela, a menos que a densidade da região de obstáculos seja muito pequena. Para uma mesma região de obstáculos, o tamanho do mapa local do agente irá determinar a maneira que o agente irá passar por ela.

5 ESPECIFICANDO CAMINHOS EM ALTO-NÍVEL

Em vários jogos e ambientes virtuais interativos, o usuário está sujeito a interagir com o sistema para definir a navegação dos personagens individualmente, ou em grupos.

Em geral, um planejador de caminhos é utilizado para auxiliá-lo nesse processo, computando um caminho livre de colisões entre uma posição inicial e uma posição final no ambiente. A maioria desses planejadores funciona em tempo-real e produz o caminho mínimo unindo essas posições. Esse caminho nem sempre é adequado, considerando que os agentes devem exibir um comportamento realístico e natural. Além disso, dependendo do comportamento desejado, esse caminho pode não ser obtido de uma maneira fácil. Um comportamento arbitrário pode necessitar a especificação de vários objetivos intermediários a serem seguidos pelos personagens, o que exigiria vários "clicks", se o dispositivo de interação fosse o mouse (GOETZ, 2006). Se esses objetivos intermediários não forem cuidadosamente escolhidos, eles podem gerar comportamentos indesejáveis, como trajetórias contínuas por partes.

O planejador BVP utiliza campos potenciais baseados na equação de Laplace como planejador global, como discutido na Seção 3.1. Definir um objetivo e gerar o campo potencial é um processo pré-computado, em uma fase *offline*. Em um ambiente interativo, o usuário frequentemente precisa manipular os agentes, e, assim, definir novos objetivos a serem seguidos por esses. Isto exige que o potencial seja re-calculado, o que, para um mapa razoavelmente grande, pode ser muito custoso, tornando o uso da técnica inviável em ambientes interativos. Para substituir o planejador global, foi utilizada uma abordagem baseada no trabalho de Dietrich et al. (DIETRICH; NEDEL; COMBA, 2008), para especificar caminhos em alto-nível, onde o usuário define a trajetória desejada apenas desenhando o caminho a ser seguido pelo agente.

A integração do planejador BVP com o trabalho de Dietrich et al. (DIETRICH; NEDEL; COMBA, 2008) é direta. Basicamente, o usuário esboça o caminho que será seguido pelos agentes no ambiente como um comandante de um exército ou um técnico de algum esporte esboçaria estratégias a serem executadas pelo seu time, conforme pode ser visto na Figura 5.1. Isto é feito utilizando o mouse.

Baseado nisso, alguns pontos desse caminho são amostrados e projetados no espaço do mundo dos agentes para serem utilizados como objetivo intermediário para cada agente. Esse mapeamento se dá conforme está especificado na Seção 3.1. Os pontos são então mapeados – um por um, na ordem que forem amostrados – para o mapa local do agente, à medida que eles forem sendo alcançados. Cada agente irá, então, em direção a esse objetivo usando o campo potencial presente em seu mapa local enquanto evita obstáculos estáticos e dinâmicos que podem surgir durante a navegação. Como vários agentes podem atuar no ambiente, para ser possível obter um comportamento de grupo, um mesmo objetivo intermediário deve ser associado a um subconjunto de agentes, como pode ser visto

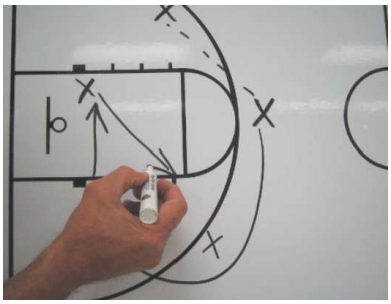


Figura 5.1: Esboços de uma estratégia para um time de basquete.

na Figura 5.2. A interação local entre os agentes se dá através do campo potencial gerado pelo planejador BVP, prevenindo colisões e permitindo que os agentes sigam o mesmo objetivo concorrentemente. Apesar de o mesmo objetivo ser associado a vários agentes, cada agente possui valores específicos para os parâmetros ϵ e **v** (conforme Equação 3.2) que os permitem seguir caminhos de uma maneira particular.

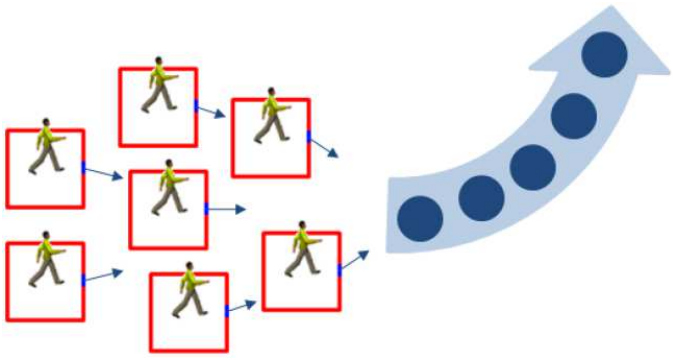


Figura 5.2: Especificando caminhos em alto-nível. Círculos são objetivos intermediários, ou seja, pontos amostrados, obtidos a partir do caminho esboçado. No mapa de cada agente, um desses pontos é mapeado como objetivo intermediário.

Outros comportamentos interessantes podem ser obtidos também a partir dessa abordagem. O usuário pode desenhar caminhos com ramificações, onde cada ramificação pode ser seguida por um subconjunto de agentes. Como exemplo, em um exército, composto por infantaria e mergulhadores, os mergulhadores podem seguir o caminho sob a água enquanto a infantaria segue o caminho sobre a terra.

5.1 Gerando Caminhos com Campos Potenciais

Inspirado na abordagem descrita na seção anterior, foi desenvolvida uma técnica para gerar um campo potencial que produza o caminho de alto-nível especificado pelo usuário. A Figura 5.3 ilustra o campo vetorial gerado, que é usado pelo agente para seguir o caminho em azul, definido pelo usuário. Esta técnica funciona como segue.

Considere um ambiente representado por uma malha m^k homogênea e uniforme. A malha m^k possui um conjunto $n_x \times n_y$ de células. Cada célula $c^k_{i,j}$ corresponde a uma região quadrada centrada nas coordenadas $r = (r_i, r_j)$ do ambiente e armazena um valor escalar que representa seu potencial p^k. Da mesma maneira que os mapas descritos na Seção 3.1.2.1, as células que correspondem à borda de m^k são marcadas como obstáculos e seus potenciais p^k recebem o valor 1. As células restantes são marcadas como

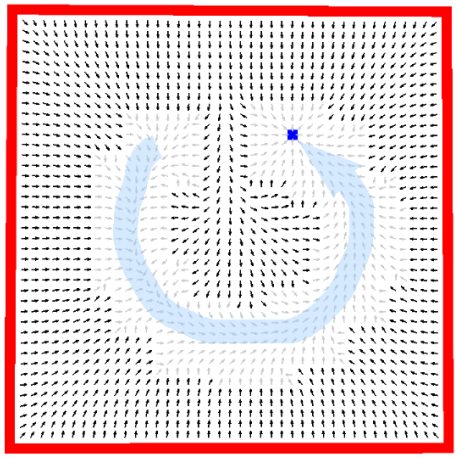
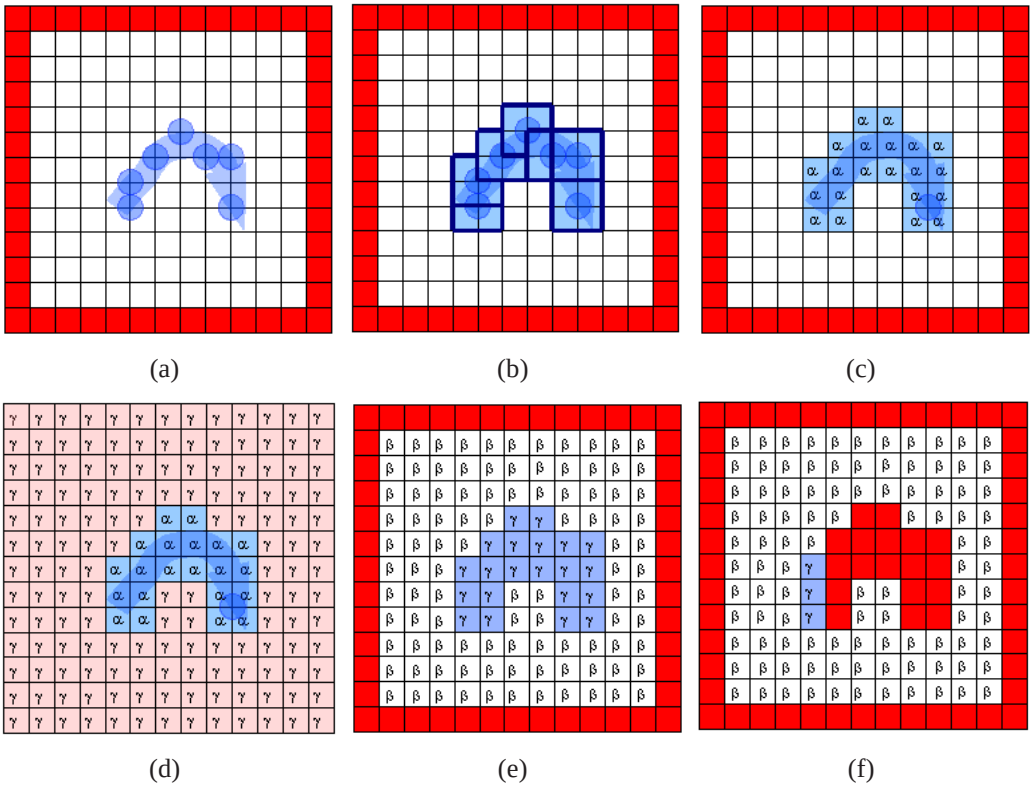


Figura 5.3: Gerando um caminho. Campo potencial gerado pelo usuário.

células livres, tendo os seus potenciais $p^k \in (0, 1)$. Esse procedimento pode ser visto na Figura 5.4-a,b.

O usuário é livre para desenhar uma curva qualquer, que os agentes deverão seguir. Em seguida, um conjunto de pontos pertencentes ao ambiente que representa essa curva, é extraído e denotado por $C = \{c^k_{i,j}\} \subset m^k$. Em seguida, uma região em volta da curva é extraída. Ela é definida por um conjunto de células $u^k = \{c \in m^k \mid \min_{c' \in C} dist(c, c') < \zeta\}$, onde $dist(c, c')$ computa a distância Euclidiana entre as células c e c' em função de seus índices na malha e ζ é um valor escalar que define a largura da região em volta da curva. As células $\omega^k = \{m^k - u^k\}$ são as células restantes da malha que não estão sobre o caminho esboçado. Portanto, as células da malha m^k podem ser divididas em u^k e ω^k.



(a) (b) (c)

(d) (e) (f)

Figura 5.4: Gerando o potencial correspondente a um caminho específico.

O processo de relaxamento é feito em duas etapas. Na primeira etapa, é feito o relaxamento considerando apenas as células $c \in u^k$, enquanto na segunda etapa, o relaxamento é feito considerando apenas as células $c \in \omega^k$.

5.1.1 Primeira Etapa: Gerando o Potencial Sobre a Curva

Na primeira etapa é gerado o potencial sobre a curva desenhada pelo usuário. Nesta etapa, o potencial de cada célula $c^k \in u^k$ é atualizado com o valor α. A célula ocupada por $q_{|\{C\}|} \in C$ (último ponto amostrado) é marcada como objetivo o^k, tendo o seu potencial atualizado com o valor 0, conforme pode ser visto na Figura 5.4-c. O potencial associado a essas células é então computado pela Equação 3.3. Durante o relaxamento, todas as células $c \in \omega^k$ são marcadas como obstáculos e mascaradas com valores potenciais iguais a γ, sendo $0 < \alpha < \gamma < 1$. Esse procedimento é exemplificado na Figura 5.4-d.

5.1.2 Segunda Etapa: Gerando o Potencial Restante

A segunda etapa é utilizada para gerar o potencial das células restantes, ou seja, $c \in \omega^k$. Durante esta etapa, os valores dos potenciais das células $c \in \omega^k$ são atualizados com o valor β, sendo $0 < \alpha < \gamma < \beta < 1$. As células $c \in u^k$, anteriormente relaxadas, são mascaradas com valores potenciais iguais a γ, e como $\gamma < \beta$, estas células se comportarão como objetivos. O potencial associado a essas células é então computado pela Equação 3.3, gerando o restante do potencial do mapa m^k.

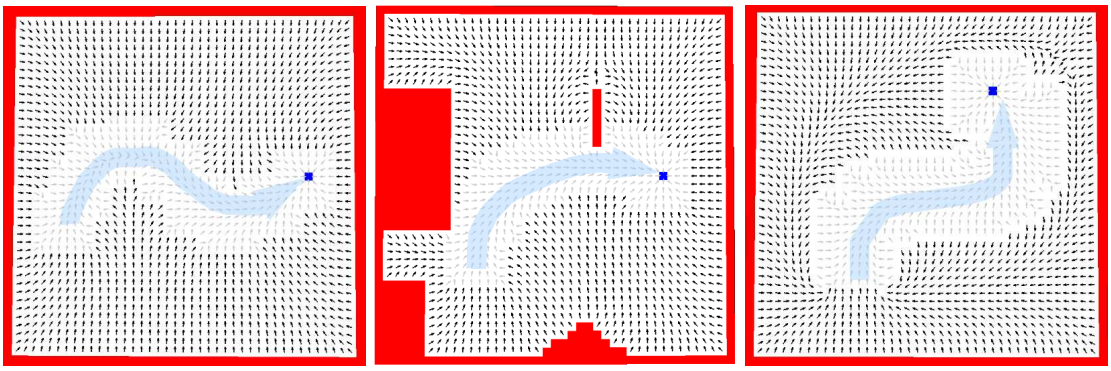
A Figura 5.4-e,f mostra duas formas diferentes de unir os potenciais. Considerar todas as células $c \in u^k$ como objetivo, e portanto, com potencial igual a γ, faz com que todos os agentes que se encontrem no mapa m^k sigam a partir de sua posição em direção a algum ponto na curva desenhada, e uma vez sobre a curva, caminhem seguindo-a até o seu fim.

Em algumas situações, pode-se desejar que o agente caminhe por toda a curva, independentemente da posição onde esse se encontra no mapa m^k. Para conseguir este comportamento, basta, nesta segunda etapa, considerar que algumas das células pertencentes a u^k, próximas ao primeiro ponto $q_1 \in C$, possuam o potencial igual a γ, comportando-se como obstáculos, e as células restantes em u^k, sejam obstáculos, ou seja, possuam seus valores potenciais iguais a 1. Isso faz com que todos os agentes sigam até o início da curva, e após isto, caminhem por toda ela. Esses tipos diferentes de comportamentos podem ser vistos na Figura 5.5-a e c.

A técnica mencionada funciona também em mapas com a presença de obstáculos. Para isso, basta na primeira etapa, mascarar as células $c \in u^k$ que representam obstáculos com valores potenciais iguais a γ; e, na segunda etapa, considerá-las como obstáculos, ou seja, considerar os seus valores potenciais iguais a 1. A Figura 5.5-b mostra essa situação. O potencial gerado pode ser usado para cada agente no ambiente alcançar o objetivo o^k, seguindo a trajetória especificada pelo usuário.

Essa estratégia pode ser utilizada em planejadores baseados em campos potenciais com funções harmônicas. Nas técnicas atuais, é possível gerar um campo potencial que forneça sempre uma direção livre de obstáculos até um objetivo especificado. Com essa nova técnica, é possível dizer como será o caminho desejado, e então, é gerado um campo potencial que forneça esse caminho. Isso possibilita uma maior interação com o planejador, facilitando a geração de caminhos naturais.

Até o momento, as abordagens propostas não lidam com o problema de grupos de agentes. Na seção seguinte, será abordada a técnica para lidar com grupos, de forma eficiente, utilizando os mesmos fundamentos do planejador BPV.

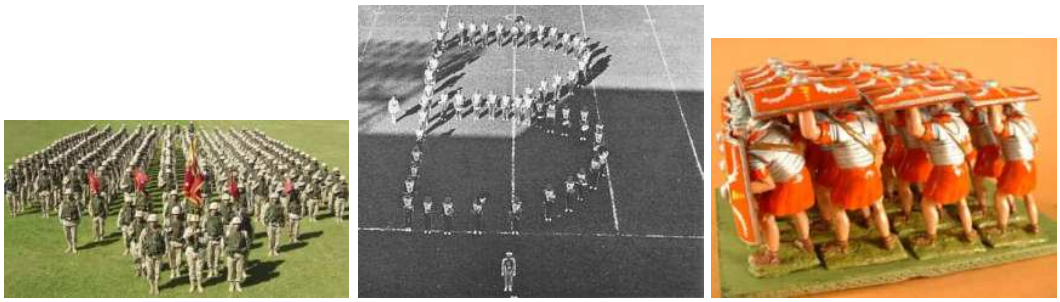


(a) (b) (c)

Figura 5.5: Em (a) o potencial é distorcido de forma arbitrária e em (b), o potencial é distorcido na presença de obstáculos. Em qualquer lugar do mapa, o agente tende a ir pelo caminho mais próximo à curva. Já em (c), o agente obrigatoriamente percorre todo o caminho especificado pela curva.

5.2 Controle de Grupo

Um dos principais objetivos de utilizar grupo é diminuir o custo computacional através do controle do grupo de agentes, ao invés de controlá-los independentemente. Apesar de ser possível fazer com que os agentes se comportem como um grupo utilizando o planejador mencionado, controlar a coerência do grupo não é uma tarefa simples e direta. Para controlar a coerência do grupo, ou seja, a disposição dos agentes enquanto se deslocam, é necessário especificar uma formação a ser mantida pelos agentes. A Figura 5.6-a mostra uma formação convencional de um batalhão. A Figura 5.6-b mostra uma formação em "B". Pode-se perceber que o objetivo de manter uma formação é possibilitar que cada agente volte sua atenção apenas para uma porção do espaço, enquanto os outros membros cobrem o resto. A Figura 5.6-c mostra soldados mantendo uma formação para se protegerem.



(a) (b) (c)

Figura 5.6: Exemplo de formações. (a) Um batalhão. (b) Um grupo de pessoas mantendo uma formação em "B". (c) Exército mantendo uma formação para se proteger.

Além disso, a técnica de Dapper et al. (DAPPER, 2007) não consegue resolver de forma satisfatória alguns problemas, por exemplo: manutenção da coerência ao passar por regiões repletas de obstáculos; movimento coerente do grupo usando uma formação pré-definida, etc. Se os agentes se comportam como um grupo e possuem um objetivo em comum, como tornar a tarefa mais eficiente garantindo a coerência do grupo enquanto seus participantes se deslocam em formação?

Para responder a essa questão, foi elaborada uma nova abordagem para lidar com grupos de agentes, de forma eficiente, que é discutida na próxima seção. A nova abordagem trata da movimentação de grupos livres ou em formação, ou seja, deve-se definir um conjunto de pontos $L = \{(x_i, y_i)\}$, compostos de $|L|$ pontos no espaço, onde $|L| = n$, sendo n a quantidade de agentes. Cada ponto está associado a um agente e é projetado como um objetivo intermediário, como pode ser visto na Figura 5.7. Esta figura mostra uma situação com forma de "V", composta por sete pontos. Cada ponto corresponde a um objetivo particular para um agente específico. Em seguida, o planejador de caminhos deve ser utilizado para mover os pontos da formação. Apesar desse abordagem ser simples, ela possui um **custo linear** em relação à quantidade de agentes, dado que cada agente possui o seu mapa local. A seguir é discutida a abordagem utilizada para resolver esse problema.

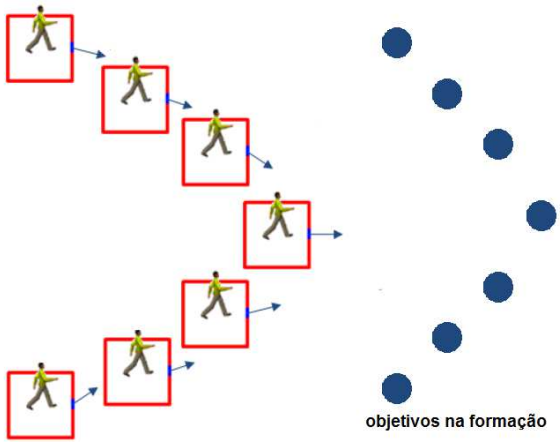


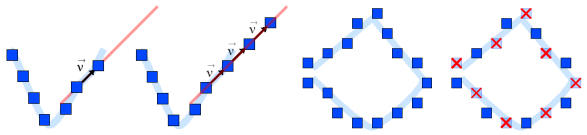
Figura 5.7: Esquema da abordagem para lidar com o problema do controle de formações. A quantidade de objetivos é igual à quantidade de agentes no grupo.

5.2.1 Especificando a Formação

Inicialmente, deve-se saber qual é a forma em que os agentes ficarão dispostos em formação. Essa forma pode ser carregada de um arquivo que armazena uma formação pré-definida ou pode ser interativamente desenhada pelo usuário. O usuário desenha uma forma qualquer, da qual, são amostrados alguns pontos. Esses pontos correspondem a uma lista $L = \{(x_i^l, y_i^l)\}$, formada por $|L|$ pontos do $\Re^2$.

Como um grupo possui n agentes, se $n \leq |L|$ então somente n pontos dessa lista são necessários, já que existe uma relação biunívoca entre os agentes e os pontos da formação. Nesse caso, esses pontos são escolhidos a partir do primeiro ponto, coletando pontos distantes $\lfloor |L|/n \rfloor$ entre si. No outro caso, se $n > |L|$ cada elemento de L é associado aos $|L|$ primeiros agentes no grupo. Os $n - |L|$ agentes restantes têm seus objetivos definidos a partir dos dois últimos elementos na lista L, ou seja, é computado um vetor $\mathbf{v} = (x^l_{|L|} - x^l_{|L|-1}, y^l_{|L|} - y^l_{|L|-1})$ usando os últimos dois pontos da lista. Adicionando esse vetor ao último ponto da lista, é possível gerar novos pontos tentando manter a forma especificada, preservando assim, a topologia da formação. Esta é uma solução de baixo custo para a geração de novos pontos, porém, não é a melhor. A curva final pode diferir em uma das extremidades da curva desenhada pelo usuário. Uma melhor solução, porém um pouco mais onerosa, seria aproximar essa forma por alguma curva, como uma *Catmull-Rom*, e a partir daí, amostrar pontos distantes $\lfloor |L|/n \rfloor$ entre si. Esse processo está exemplificado

na Figura 5.8.



(a) (b)

Figura 5.8: (a) O usuário desenha uma formação em V com menos pontos amostrados do que a quantidade de agentes. O vetor $\vec{v}$ é definido e novos pontos que representarão a formação são criados. (b) O usuário desenha um formação em diamante com mais pontos amostrados do que o necessário. Alguns pontos são eliminados, mantendo os pontos igualmente distantes.

Os pontos amostrados da lista L compõem os pontos da formação, e compreendem uma nova lista $\mathcal{L}$, com $|\mathcal{L}| = n$. Esses pontos representam uma topologia para a formação, e poderão ser vistos como pontos atratores virtuais, que "puxarão" os agentes para eles.

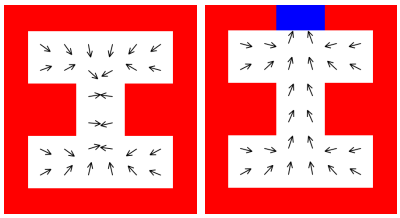
5.2.2 Mapa do Grupo

Cada formação está associada a um grupo, e portanto, a um mapa de $\gamma_x \times \gamma_y$ células. O mapa do grupo é projetado no ambiente virtual e o seu centro $(\sum_{i=1}^{|\mathcal{L}|} x_i^l/|\mathcal{L}|, \sum_{i=1}^{|\mathcal{L}|} y_i^l/|\mathcal{L}|)$ é alinhado com o centro do grupo, definido por $(\sum_{i=1}^{|\mathcal{L}|} x_i^a/|\mathcal{L}|, \sum_{i=1}^{|\mathcal{L}|} y_i^a/|\mathcal{L}|)$, onde (x_i^a, y_i^a) é a posição do agente i no ambiente. Esse mapa existirá sempre que um grupo for utilizado, sendo um mapa para cada grupo, independentemente da existência de formações. Esse mapa será a base para o movimento do grupo.

Cada célula do mapa do grupo armazena um valor de potencial. Como no mapa local de um agente, descrito na Seção 3.1, a borda do mapa e cada célula que corresponde a obstáculo no ambiente irão armazenar um alto valor para o potencial, o valor 1, enquanto as outras células, as células livres, irão armazenar inicialmente um valor de potencial entre 0 e 1. Para um maior desempenho, deve-se utilizar os valores discutidos na Seção 4.1, ou seja, valores próximos de 0.9.

Conforme já discutido, o planejador global fornece uma direção livre de colisão utilizada para identificar quais células na borda do mapa do grupo serão os objetivos intermediários (valores potenciais iguais a 0).

Caso essas células estejam obstruídas, é feita uma propagação no sentido oposto ao sentido do objetivo, procurando células livres para que o mapeamento do objetivo no mapa possa ser feito sempre com sucesso. Esse objetivo irá atrair agentes que estão dentro do mapa do grupo, resultando no movimento dos agentes no grupo. A presença do objetivo intermediário é essencial para tornar possível o movimento dos agentes através de uma passagem estreita, como pode ser visto na Figura 5.9. O algoritmo de Gauss-Seidel é então executado, da mesma maneira discutida na Subseção 3.1.1, para computar o potencial das células que nem são obstáculos, nem são bordas. O campo vetorial deste potencial é extraído e os agentes movimentam-se seguindo esse campo. O movimento do agente é influenciado, então, por duas forças: a força da formação e a força do gradiente. Para evitar comportamentos indesejados ou não-naturais, a influência de cada uma destas forças deve ser devidamente estabelecida.



(a) (b)

Figura 5.9: (a) Sem o objetivo intermediário, os agentes nunca irão passar através de uma passagem estreita. (b) Com os objetivos intermediários, o gradiente das células nas passagens estreitas apontará para o objetivo. Os agentes poderão então passar através de passagens estreitas.

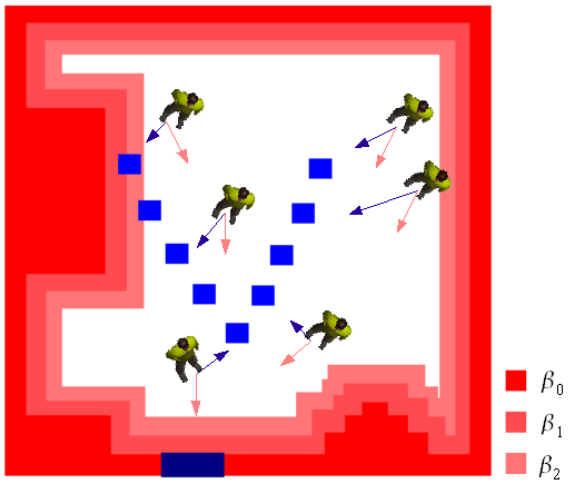
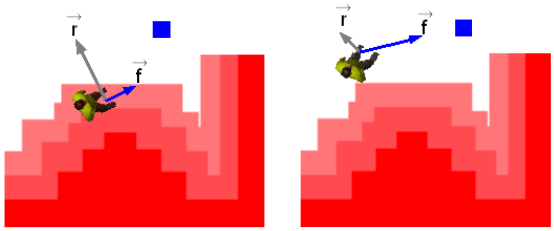


Figura 5.10: Mapa do Grupo. Agentes dentro do mapa estão sob a influência das forças da formação. As setas em azul escuro indicam a direção da força de formação, enquanto as setas vermelho-claras indicam o gradiente do ambiente. A região β_0 é a borda do mapa do grupo que por sua vez, é uma zona de obstáculos usada como condição de contorno do BVP. A borda β_1 é a zona onde o gradiente tem a influência máxima sobre a força de formação e β_2 é a zona onde o gradiente possui uma alta influência sobre a força da formação.

5.2.3 Estabelecendo Regiões de Influência das Forças Resultantes

Células que estão i células distantes dos obstáculos ou das bordas, compreendem a região chamada β_i, com $i \in \{1, 2, \ldots, max\}$. Cada célula da região β_i possui um valor escalar μ_{β_i} que é utilizado para determinar o quanto um agente está próximo de um obstáculo ou de uma borda. Deve-se garantir que o valor escalar associado com cada região aumente na medida em que se aproxima dos obstáculos. Em outras palavras, $\mu_{\beta_1} = 1$, e $\mu_{\beta_1} > \mu_{\beta_2} > ... > \mu_{\beta_{max}}$. Isso pode ser feito durante o processo de relaxamento, quando as regiões β_i são criadas. Uma ilustração do mapa do grupo pode ser vista na Figura 5.10. Quando um agente está em uma célula associada a uma dessas regiões, $(\beta_1, \beta_2, ..., \beta_{max})$, ele sofrerá a influência não somente da força exercida pela sua posição na formação, mas também pelo gradiente descendente calculado em sua posição atual no mapa do grupo. A influência do gradiente associado à região β_i tornar-se-á fraca na medida em que $i \rightarrow max$. A Figura 5.11 mostra a influência das regiões β_i em um agente que encontra-se sobre o mapa do grupo. Um agente sobre uma região β_i sofre uma influência

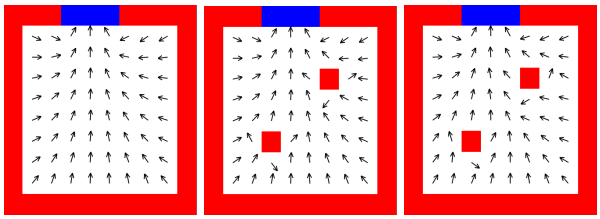
maior do gradiente local, enquanto que um agente fora de uma região β_i sofre uma maior influência da força da formação. Após o processo de relaxamento, cada agente é mapeado



(a) (b)

Figura 5.11: Em (a), um agente dentro de uma região β_i sofre uma maior influência do gradiente **r** do ambiente. Em (b), um agente fora de uma região β_i sofre uma influência maior da força **f** exercida pela formação.

no mapa do grupo como um obstáculo, ou seja, a célula ocupada por ele armazenará um valor potencial igual a 1. Ao mesmo tempo, suas 8 células vizinhas são marcadas como células pertencentes à região β_1. Isso implica que a cada vez que um agente se mover, não é necessário realizar o processo do relaxamento em todo o mapa do grupo, mas apenas restaurar os valores dos potenciais das células anteriormente ocupadas pelo agente e marcar as células atualmente ocupadas pelo agente como obstáculos. Para um movimento mais natural, pode-se suavizar o campo potencial utilizando poucas iterações do processo de relaxamento, conforme pode ser vista na Figura 5.12. Isto poupa tempo e aumenta o desempenho do sistema quando comparado com a técnica de Dapper et al. (DAPPER, 2007).



(a) (b) (c)

Figura 5.12: (a) Um campo potencial após o processo de relaxamento. (b) A posição do agente é mapeada como obstáculo. (c) O relaxamento é feito para suavizar o campo vetorial.

5.3 Gerando o Movimento

O movimento do grupo é o resultado de um algoritmo de duas camadas. A primeira camada controla o movimento individual de um agente, enquanto a segunda camada é responsável pelo movimento do grupo como um todo. Na primeira camada, o usuário desenha a formação (Figura 5.13-(a)(c)) a qual é processada e associada ao mapa do grupo (Figura 5.13-(b)(d)). Todos os agentes de um grupo estão dentro desse mapa e movem-se de acordo com as forças resultantes percebidas por eles (Figure 5.13-e). A segunda camada planeja um caminho livre de colisão o qual o mapa do grupo irá seguir. Qualquer

algoritmo de planejamento de caminho pode ser usado nessa camada. Na implementação atual, foi utilizado o planejador em alto nível, apresentado na Seção 5, integrado com o planejador BVP, apresentado na Seçao 3.1.

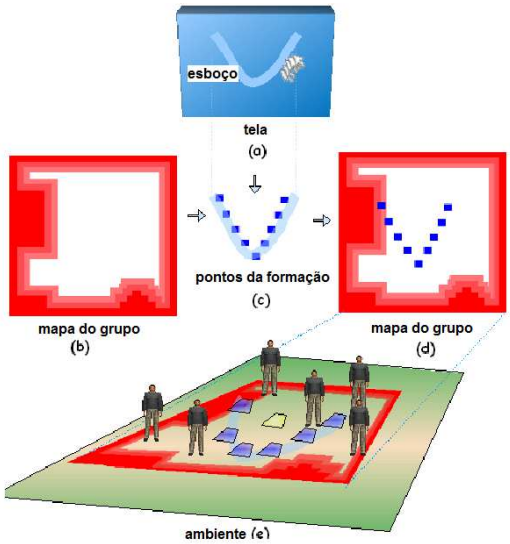


Figura 5.13: Esquema da geração do movimento: (a) O usuário desenha a formação na tela; alguns pontos são amostrados (c); o mapa do grupo é criado (b); e os agentes movem-se de acordo com as forças resultantes que o afetam (d, e).

5.3.1 Primeira Camada: Movendo os Agentes Individualmente

Como pode ser visto na Figura 5.10, cada agente está associado ao mapa do grupo ao qual pertence e é influenciado pela força que o mantém em formação e pela força exercida pelo campo potencial. A cada passo, o agente combina essas forças, gerando uma força resultante que o guia através de um caminho livre de colisão e que o mantém em formação.

O gradiente calculado a partir do mapa do grupo de um agente k é

$$\mathbf{g} = \left(\frac{\mathcal{P}_{x_k+1,y_k} - \mathcal{P}_{x_k-1,y_k}}{2}, \frac{\mathcal{P}_{x_k,y_k+1} - \mathcal{P}_{x_k,y_k-1}}{2} \right)$$

onde $\mathcal{P}_{(x_k,y_k)}$ é o potencial armazenado na posição (x_k, y_k) ocupada pelo agente k no mapa do grupo. Projetando os pontos da formação no mapa do grupo, pode-se calcular a força **z** cuja direção é determinada pela reta-suporte que liga o ponto da formação (x_k^l, y_k^l) à posição corrente (x_k, y_k) do agente a ele associado, e cujo sentido o do segmento orientada do agente para o ponto da formação. Assim, $\mathbf{z}_k = (x_k^l, y_k^l) - (x_k, y_k)$. O termo **z**, representado por sua orientação ω^t é então adicionado à Equação 3.6, produzindo a equação seguinte, que relaciona a posição r do agente k no tempo t,

$$\Delta\mathbf{r} = \upsilon_{max} \Bigg[\underbrace{\mu_{\beta i} \Big(\Psi(|\zeta^t - \varphi^{t-1}|)(cos(\varphi^t), sin(\varphi^t)) \Big)}_{\text{ambiente}} + \underbrace{(1-\mu_{\beta i}) \Big(\Psi(|\omega^t - \omega^{t-1}|) min(\sqrt{(x_k^l - x_k)^2 + (y_k^l - y_k)^2}, 1)(cos(\omega^t), sin(\omega^t)) \Big)}_{\text{formação}} \Bigg] \quad (5.1)$$

onde

φ^t é a orientação do agente no instante t, definida na Equação 3.6;

ζ^t é a orientação do gradiente descendente na posição atual no instante t;

ω^t é a orientação de **z**;

$\Psi : \mathbf{R} \rightarrow \mathbf{R}$, definida na Equação 3.6;

(x_k^l, y_k^l) é a posição do ponto da formação associado ao agente;

(x_k, y_k) é a posição atual do agente.

v_{max} não deve ser muito maior que a velocidade do mapa do grupo para garantir que o agente sempre permaneça dentro do mapa do grupo. O valor escalar μ_{β_i} é associado a cada célula da região β_i. Se o termo relacionado à força da formação for nulo, então o grupo se move sem uma formação definida.

5.3.2 Segunda camada: Movendo os Grupos

O movimento dos grupos é produzido pelo controle do movimento dos mapas de formação. Para isso, considera-se o ponto médio de cada mapa do grupo como um único indivíduo. Portanto, qualquer algoritmo de planejamento de caminho pode ser utilizado para obter-se um caminho livre de obstáculos. Em nossas implementações, utilizou-se o planejador BVP.

Em cada passo, o mapa do grupo move-se no ambiente. Informações sobre a região do ambiente delimitada pelo mapa do grupo são projetadas no mapa do grupo. Obstáculos são projetados no mapa do grupo e os objetivos são mapeados como objetivos intermediários, na borda do mapa do grupo, e é realizado o processo de relaxamento. Após o relaxamento, as células ocupadas pelos agentes são marcadas como obstáculos. Nessa etapa, os pontos de formação são rotacionados para alinhar sua orientação com a direção fornecida pelo planejador global, nesse caso, o planejador BVP. Os pontos da formação são, então, reprojetados no mapa do grupo, exercendo uma força atrativa nos personagens.

Na seção seguinte, é apresentado um pseudo-código da técnica proposta nos capítulos anteriores.

5.3.3 Algoritmo

Nesta seção, é apresentado o algoritmo que implementa os conceitos mostrados para produzir o movimento de grupos de agentes em um ambiente virtual.

O primeiro laço inicializa os mapas do grupo. Os dois próximos laços são executados para obter do usuário o caminho a ser seguido e a formação a ser mantida pelo grupo, caso o movimento do grupo ocorra respeitando alguma formação específica.

O laço 15 executa as atualizações do mapa do grupo e o prepara para que os agentes possam se mover. O laço 25 move os agentes de cada grupo de forma individual e assíncrona e, em seguida, o grupo se move. A primeira e segunda camadas do algoritmo são executadas nesses dois laços.

Os passos de 5 a 8 são executados apenas se o planejador global utilizado for o planejador descrito na Seção 5. É importante ressaltar que os passos 18, 22 e 33 dependem do planejador global. Qualquer planejador que forneça uma direção livre de obstáculos pode ser utilizado nessa etapa.

Algorithm 3 Planejador BVP - Extensão para Grupos

1: Selecionar as unidades que constituirão cada grupo.
2: **for** cada grupo g_k **do**
3: Criar o mapa do grupo.{*Conforme descrito na Seção 5.2.2.*}
4: **end for**
5: **for** cada grupo g_k **do**
6: Desenhar a trajetória.
7: Processar os pontos da trajetória.{*Conforme descrito na Seção 5.*}
8: **end for**
9: **for** cada grupo g_k **do**
10: **if** possuir formação **then**
11: Desenhar a formação.
12: Processar os pontos da formação.{*Conforme descrito na Seção 5.2.1.*}
13: **end if**
14: **end for**
15: **for** cada grupo g_k **do**
16: Detectar obstáculos estáticos e dinâmicos.
17: Atualizar o mapa do grupo com informação sobre os obstáculos e os agentes.
18: Computar o gradiente global $\mathbf{o}_k$.{*Obtido a partir do planejador global.*}
19: **if** possuir formação **then**
20: Rotacionar os pontos da formação
21: **end if**
22: Obter o $objetivo(\mathbf{o}_k)$ e o projetar na borda do mapa do grupo, ou nas células livres mais próximas.
23: Executar o relaxamento, criando as regiões de influência β_i.
24: **end for**
25: **for** cada grupo g_k **do**
26: **for** cada agente a_i^k **do**
27: Marcar a célula onde se encontra o agente a_i^k como célula livre.
28: Computar a direção livre de obstáculos que o agente a_i^k seguirá de acordo com a equação 5.2.
29: Marcar a célula onde se encontra o agente a_i^k como obstáculo.
30: Executar algumas iterações do relaxamento no mapa do grupo g_k.
31: **end for**
32: **end for**
33: Mover o mapa do grupo, como se fosse um único agente.{*Na implementação, o grupo foi considerado um agente do planejador BVP.*}
34: **while** não alcançou o objetivo final **do**
35: Repita os passos 15 to 33.
36: **end while**
37: Pare de mover.

6 EXPERIMENTOS, RESULTADOS E DISCUSSÃO

Para validar a técnica proposta neste trabalho foi desenvolvida uma aplicação, que consiste de um ambiente virtual em 3D, povoado por humanóides virtuais. A Figura 6.1 mostra uma tela da aplicação desenvolvida.

O planejador BVP, descrito na Seção 3.1, foi re-implementado, levando em conta todas as otimizações mencionadas na Seção 4.1. Até o momento, o planejador BVP não tinha sido validado com humanos virtuais. Assim, este trabalho serviu também para validar a utilização do planejador BVP com humanos virtuais.

Inicialmente, foi utilizada a biblioteca **ReplicantBody**[1] para cuidar das animações dos humanos virtuais. Porém, atualmente o projeto encontra-se parado, e devido à falta de suporte e a alguns *bugs* encontrados, optou-se por eliminá-la da implementação, e utilizar apenas **OpenSceneGraph**.

Para a implementação do planejador, utilizou-se então a linguagem de programação C++ com as bibliotecas **OpenSceneGraph**, **Cal3D**, **osgCal**, **OpenGL** e **GLUT**. Todos os modelos utilizados são livres, grátis, fornecidos nos exemplos da **OpenSceneGraph** e de um projeto chamado Delta3D, também código aberto, construído a partir da **OpenSceneGraph**.

- **OpenSceneGraph (OSG)** é uma biblioteca com código aberto, que fornece um grafo de cena, junto com funções eficientes de *rendering* utilizadas para obter alto desempenho em aplicações 3D. É amplamente utilizada em aplicativos de visualização científica, modelagem computacional, realidade virtual, jogos computacionais e simulações. A biblioteca é escrita em C++ e utiliza **OpenGL**. É multiplataforma, podendo ser utilizada nos sistemas operacionais Microsoft Windows, Mac OS X, Linux, IRIX, Solaris e FreeBSD. Essa biblioteca pode ser obtida em *http://www.openscenegraph.org*.

- **Cal3D** é uma biblioteca para animação de personagens virtuais baseada em esqueletos. É inteiramente escrita em C++ e, assim como OSG, é independente de plataforma. Esta biblioteca pode ser obtida em *http://home.gna.org/cal3d/*.

- **osgCal** é um *plugin* para poder utilizar Cal3D com a biblioteca OSG. Esse *plugin* pode ser obtido em *http://osgcal.sourceforge.net*.

Durante este trabalho, foram desenvolvidos alguns experimentos para avaliar as técnicas e testar sua utilidade na geração de caminhos para grupos de humanos virtuais. Foram propostas algumas situações onde se utiliza as abordagens desenvolvidas. Na seção seguinte, são mostradas algumas soluções anteriores para o mapa do grupo, utilizada em

[1] http://www.vrlab.umu.se/research/replicantbody/

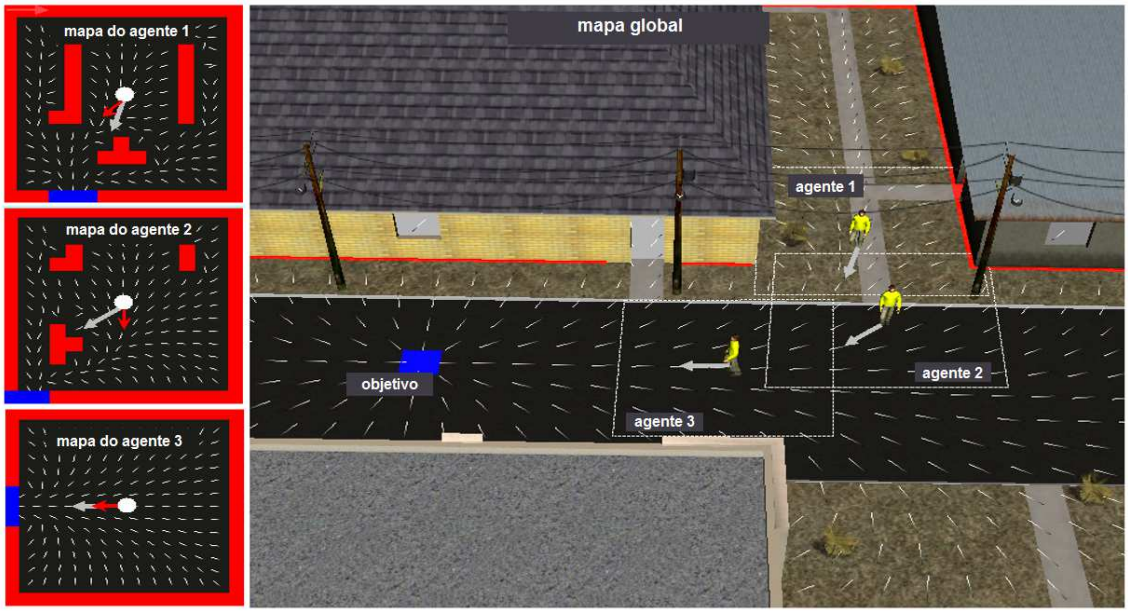


Figura 6.1: Ambiente virtual de teste. Neste cenário existem 3 agentes que devem alcançar a posição $target_{o1}$.

uma versão prévia da implementação. É mostrada também a substituição do planejador global por um planejador em alto-nível, baseado em esboço, e também, a estratégia adotada para esboçar uma forma específica para definir a formação do grupo.

Foram realizados alguns experimentos para gerar situações muitas vezes desejadas em jogos ou ambientes virtuais, que são mostrados em seguida, como a capacidade de manter a formação enquanto passa por ambientes repleto de obstáculos, a possibilidade de seguir um líder durante o movimento, a naturalidade do movimento ao passar por passagens estreitas e a possibilidade de escolha de quando ocorrerá a quebra do grupo.

Por fim, a nova técnica é analisada em comparação com a técnica anterior e são comentadas as vantagens, como o ganho de desempenho, assim como as desvantagens de utilizar a técnica proposta, ao invés da técnica anteriormente proposta por Dapper et al (DAPPER, 2007) para o movimento de grupos.

6.1 Soluções Anteriores para o Mapa do Grupo

Abaixo seguem algumas idéias que foram usadas nas versões prévias em conjunto com o mapa do grupo, enumerando suas vantagens e desvantagens.

6.1.1 Mapeando os Pontos da Formação como Objetivos

Uma alternativa para a construção do mapa do grupo para tratar o problema da formação seria mapear todos os pontos pertencentes a $L = \{(x_i^l, y_i^l)\}$ como objetivos no mapa, atualizando os potenciais das células que correspondem a esses pontos com o valor 0. Durante cada passo do deslocamento desses pontos, conforme discutido na Seção 5.3.2, uma rotação é realizada nesses pontos para alinhá-los com a direção fornecida pelo planejador global. Conforme pode ser visto na Figura 6.2, após a rotação, alguns pontos da formação podem se distanciar bastante de sua posição anterior, e portanto, é feita uma interpolação linear entre sua posição atual e sua posição anterior. O processo do relaxamento é executado, e após o relaxamento, o campo potencial gerado faz com que cada

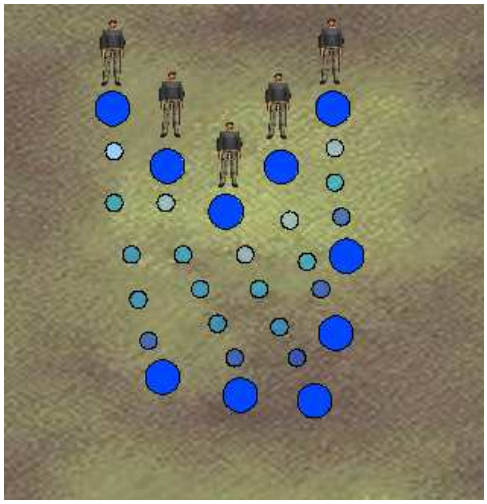
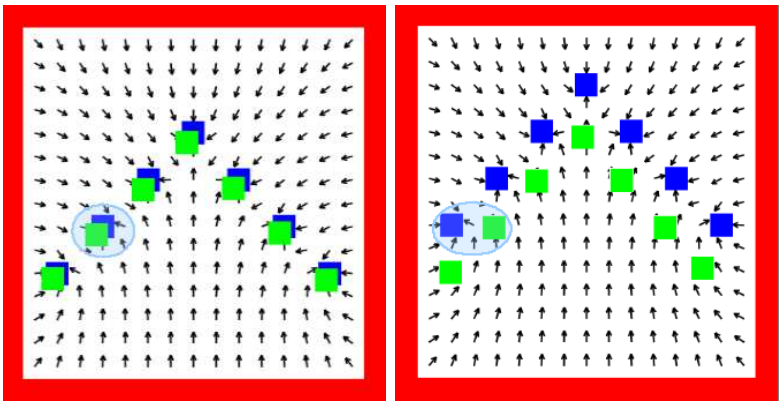


Figura 6.2: Rotação dos pontos da formação. Durante a rotação dos pontos da formação estes podem se distanciar muito em relação a sua posição anterior.

agente sempre se dirija ao objetivo mais próximo. Porém, o objetivo mais próximo nem sempre é o objetivo que está anteriormente associado ao agente, como pode ser visto na Figura 6.3. Após um passo, dependendo da formação, o objetivo mais próximo de um agente pode estar mais próximo de seu vizinho. Isso faz com que o agente compartilhe objetivos, perdendo a coerência da formação e gerando movimentos não-naturais. Essa situação geralmente acontece após uma rotação, os pontos gerados pela interpolação podem cruzar-se, conforme mostra a Figura 6.4. O agente segue esses pontos, mapeados como objetivos, e após o cruzamento, o agente pode seguir o objetivo do agente vizinho.



(a) (b)

Figura 6.3: (a) Agentes estão seguindo uma formação (b) Após um passo, o objetivo de um agente fica mais próximo de seu vizinho, fazendo com que os agentes percam a formação e compartilhem objetivos.

6.1.2 Gerando Comportamento Diferenciado para Cada Entidade do Grupo

Com a utilização do mapa do grupo, cada grupo pode ter um comportamento específico, mas os agentes que o compõe estarão sujeitos ao mesmo campo potencial, ou seja, estarão sujeitos aos mesmos pares $(\mathbf{v}, \epsilon)$. Para se obter um comportamento diferenciado para cada entidade, seria necessário a utilização de um mapa local para cada agente, mas

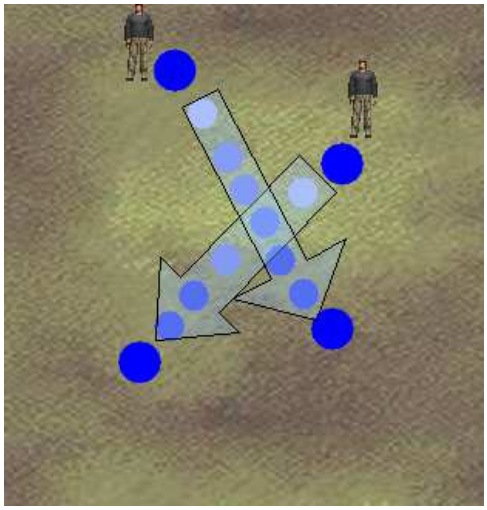


Figura 6.4: Durante a rotação dos pontos da formação, os caminhos podem cruzar-se.

sempre fornecendo o mesmo objetivo para todos os agentes. E o movimento de formações poderia ser obtido naturalmente como discutido na Seção 5.2.2. Entretanto, diferente da abordagem descrita anteriormente, não seria possível garantir a coesão do grupo ao passar entre obstáculos e não haveria ganho algum no desempenho, se comparado com o planejador BVP.

6.2 Esboçando Caminhos em Alto-Nível

Com a técnica proposta para esboçar caminhos em alto-nível, foi possível especificar objetivos dinâmicos, possibilitando a interação com os agentes. A Figura 6.5 mostra a utilização do planejador baseado em esboços. Na Figura 6.5-a, foi esboçado um caminho sobre a rua. Na Figura 6.5-b, é mostrada uma outra situação onde os agentes estão seguindo um caminho esboçado.

(a) (b)

Figura 6.5: Especificando caminhos em alto-nível.

Conforme mencionado no Capítulo 5, esse planejador substitui o planejador global e deixa o usuário livre para desenhar qualquer trajetória a ser seguida pelos agentes. Entretanto, essa liberdade pode acarretar em um problema. Quando um caminho intercepta algum obstáculo muito maior do que uma dimensão de seu mapa do grupo, esse obstáculo pode bloquear a passagem, impedindo que os agentes prossigam.

6.3 Esboçando Qualquer Formação

Na Seção 5.2.1 foi apresentada uma maneira de lidar com o problema de especificar uma formação. O usuário é livre para esboçar qualquer formação desejada e os agentes ocuparão algum lugar nessa formação.

Com a técnica abordada, a formação pode ser esboçada dinamicamente enquanto os agentes se movimentam. Uma mudança na forma da formação também pode ser feita dinamicamente, bastando apenas esboçar como será a nova forma.

A Figura 6.6 mostra exemplos de formações. Essas formações podem ser alteradas durante o movimento, esboçando uma nova formação. Os agentes, então, migram para a nova posição na formação, enquanto mantém seu caminho rumo ao objetivo do grupo.

(a) (b) (c)

Figura 6.6: (a) Uma formação desenhada arbitrariamente. (b) Uma formação em quadrado. (c) Uma formação em “V”.

6.4 Passando através de Passagens Estreitas

Geralmente, as técnicas baseadas em campos potenciais apresentam um comportamento não-natural diante de passagens estreitas. Algumas técnicas tornam-se mais lentas quando há essas passagens, devido à necessidade de um passo adicional no algoritmo para suavizar o caminho.

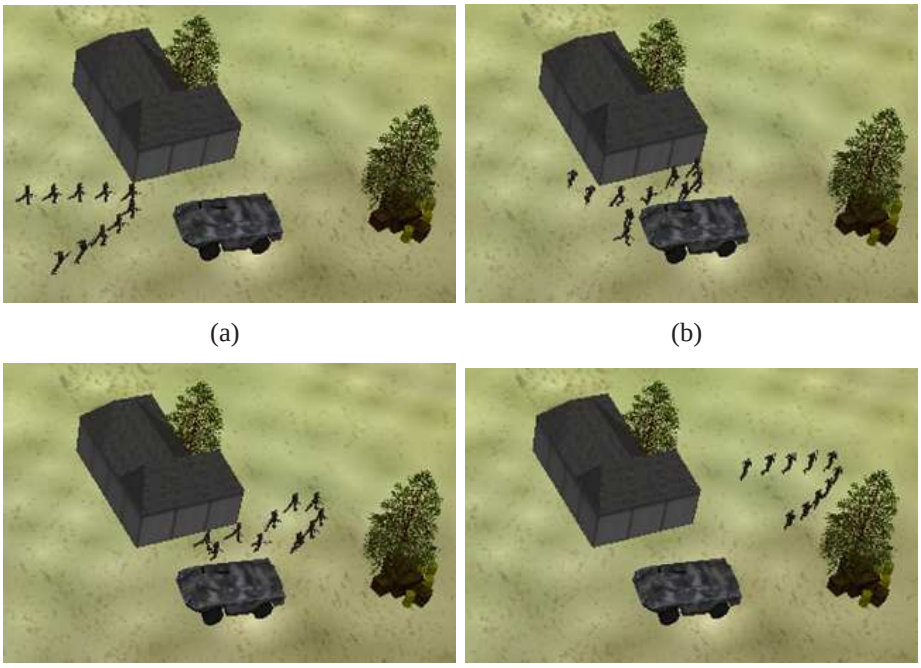
A técnica proposta produz um caminho suave na presença de corredores e passagens estreitas. Casualmente, é possível que ocorra o problema do achatamento. Esse problema ocorre porque em determinadas regiões, o potencial fica muito próximo do valor 1, de forma que o gradiente não é mais calculado corretamente.

Dapper et al. (DAPPER, 2007) resolveram o problema do achatamento através da colocação de subobjetivos nas regiões achatadas, possibilitando sempre a geração de caminhos suaves em passagens estreitas. A Figura 6.7 mostra agentes passando em uma passagem estreita. A formação é preservada sempre que possível.

6.5 Seguindo um Líder

Um comportamento interessante, como o de um grupo seguir uma unidade, pode ser facilmente obtido com a técnica proposta. Para que isso seja possível, uma entidade deve ser definida como o líder. Essa entidade pode ser algum agente, ou algum objeto que esteja presente no ambiente. Esse objeto é guiado pelo seu mapa local, através do planejador BVP, ou por qualquer outro planejador.

Para fazer com que o grupo siga alguma entidade, basta utilizar a posição em que a entidade se encontra e mapeá-la como objetivo intermediário no mapa do grupo. A

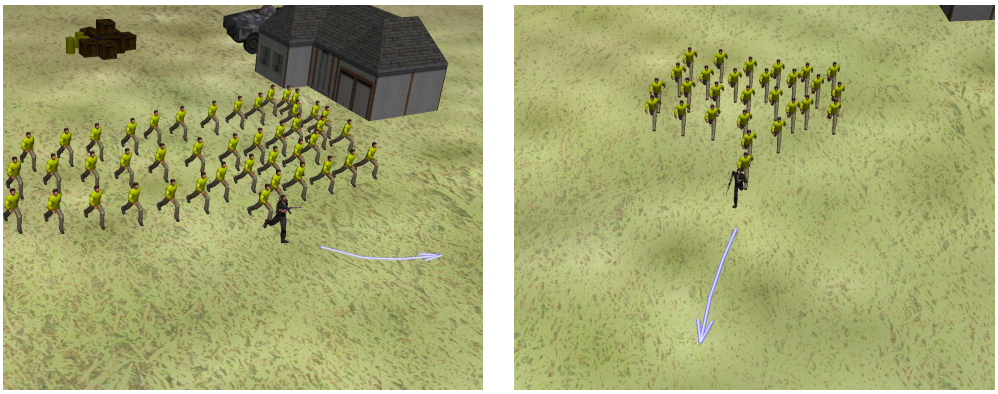


(a) (b)

(c) (d)

Figura 6.7: Um grupo em formação (a) precisa atravessar uma passagem estreita (b); a formação é deformada (c) para permitir a passagem do grupo; e quando o ambiente é aberto novamente (d), o grupo volta à formação original.

Figura 6.8 ilustra esse comportamento. Um planejamento é feito para o agente de preto, enquanto o grupo composto por agentes vestidos de amarelo seguem-no.

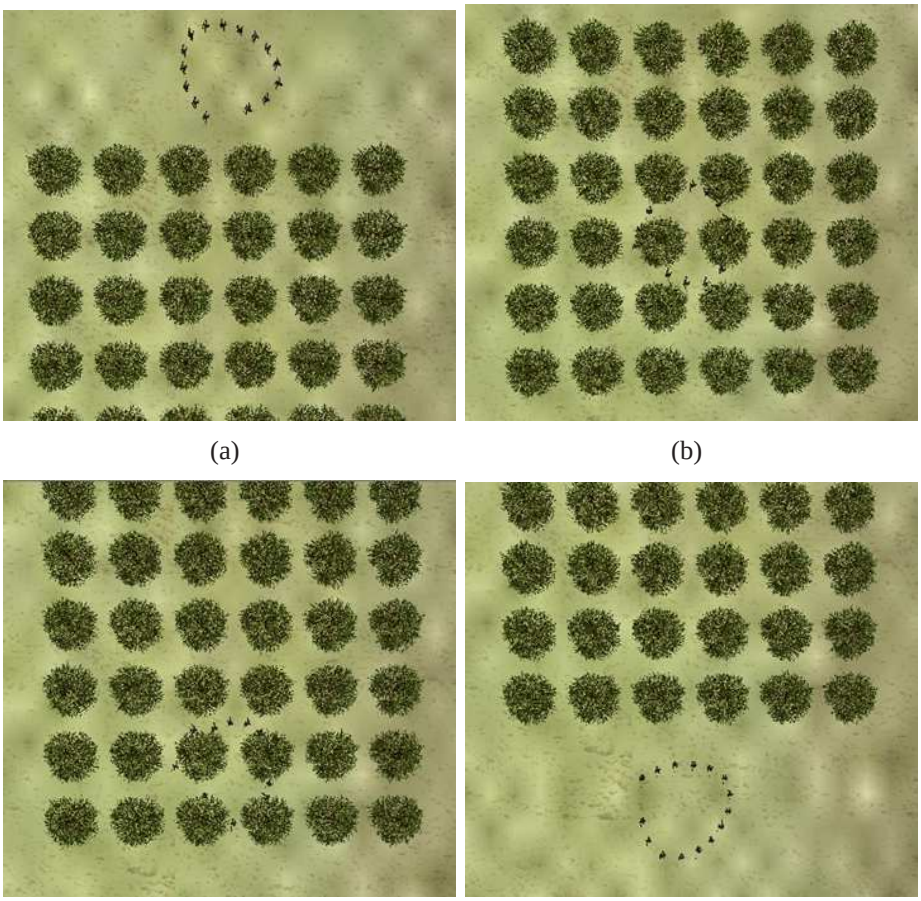


(a) (b)

Figura 6.8: Líder guiando um grupo em formação.

6.6 Ambientes com Muitos Obstáculos

Em ambientes onde se encontram muitos obstáculos, o grupo se dispersa ao passar por eles. Porém, enquanto o grupo caminha mantendo uma formação, é desejável que esta formação seja mantida. A Figura 6.9 mostra um grupo passando por um ambiente com muitos obstáculos. O grupo mantém, sempre que possível, a formação, enquanto desvia-se dos obstáculos e evita a colisão com os outros agentes.



(a) (b)

(c) (d)

Figura 6.9: (a) Grupo passando por um ambiente povoado por obstáculos, mantendo a formação sempre que possível (b,c,d).

6.7 Controle da Coesão do Grupo

Em ambientes que possuem regiões com muitos obstáculos, o grupo pode quebrar-se. Obstáculos que estão inteiramente representados dentro do mapa do grupo são automaticamente evitados, quebrando o grupo em subgrupos que seguem rotas alternativas para alcançar o mesmo objetivo, como pode ser visto na Figura 6.10.

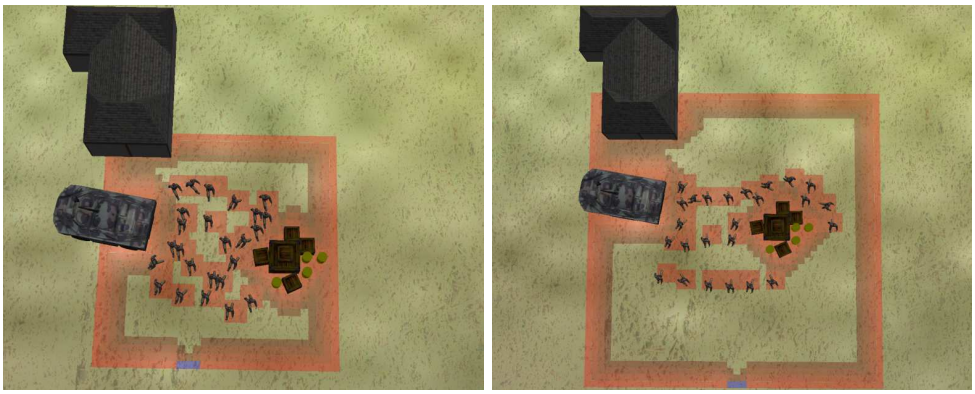
Esse comportamento pode ser controlado com a técnica proposta, através de mudanças do tamanho do mapa do grupo. Como exemplo, quando um grupo está caminhando em uma floresta, é aceitável que alguns arbustos sejam evitados quebrando o grupo, mas ao passar por grandes pedras, o grupo deve permanecer coeso.

O comportamento de coesão do grupo depende da relação direta entre o tamanho do mapa do grupo e o tamanho do obstáculo. Em um jogo, ou algum ambiente virtual, o artista que modelou o ambiente poderia especificar quais obstáculos são grandes o bastante para que o grupo permaneça unido. Essa informação pode ser utilizada para reduzir o tamanho do mapa do grupo dinamicamente.

A Figura 6.11 mostra ambas as situações. Para um mapa do grupo com 20×20 células, o grupo passa pelo obstáculo mantendo a coesão. Para um mapa do grupo com 30×30 células, o mesmo grupo se subdivide.

(a) (b)

Figura 6.10: (a) Um grupo de agentes deve mover-se na direção indicada pela seta. (b) O grupo inesperadamente se subdivide (NIEUWENHUISEN; KAMPHUIS; OVERMARS, 2007).



(a) (b)

Figura 6.11: Utilizando um mapa de formação pequeno, o grupo passa pelos obstáculos mantendo a coesão (a); mas com um mapa do grupo grande, o mesmo grupo se subdivide (b).

6.8 Considerações sobre Desempenho

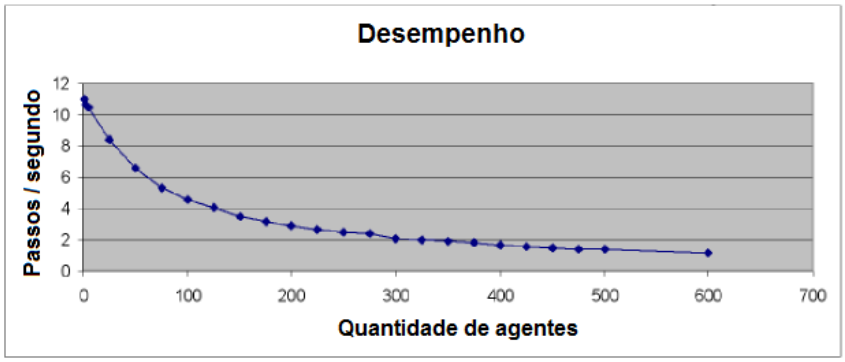
Em um ambiente em que seja necessário utilizar o planejador apresentado neste trabalho, para cada novo passo do agente, o planejador de movimento deve fornecer a nova posição e a nova orientação do agente. Assim, para avaliar se o algoritmo pode ser utilizado em tempo-real, Dapper et al. (DAPPER, 2007) consideraram a freqüência de passos que uma pessoa pode atingir. Segundo Mazarakis e Avaritsiotis (MAZARAKIS; AVARITSIOTIS, 2005), essa freqüência varia de 0.9 a 1 Hz para alguém andando muito devagar; até 3.5 Hz para alguém correndo muito rápido; sendo que a média para uma pessoa caminhando é de 2 Hz. Portanto, o processamento deve ser suficiente para a renderização do ambiente virtual, incluindo a animação dos agentes, e para calcular até 3.5 passos por segundo do algoritmo de planejamento de movimento.

Com o objetivo de aumentar o número de agentes na simulação, em muitas aplicações, onde a maioria das pessoas caminha no ambiente, pode-se considerar a média de 2 passos por segundo. Para animações, quando existe foco em determinadas regiões ou personagens, pode-se optar por utilizar o algoritmo apenas entre os personagens em destaque e para os demais, utilizar outras técnicas como as usadas em simulação de multidões. Para estimar o número de agentes que o algoritmo pode controlar, foi realizado um expe-

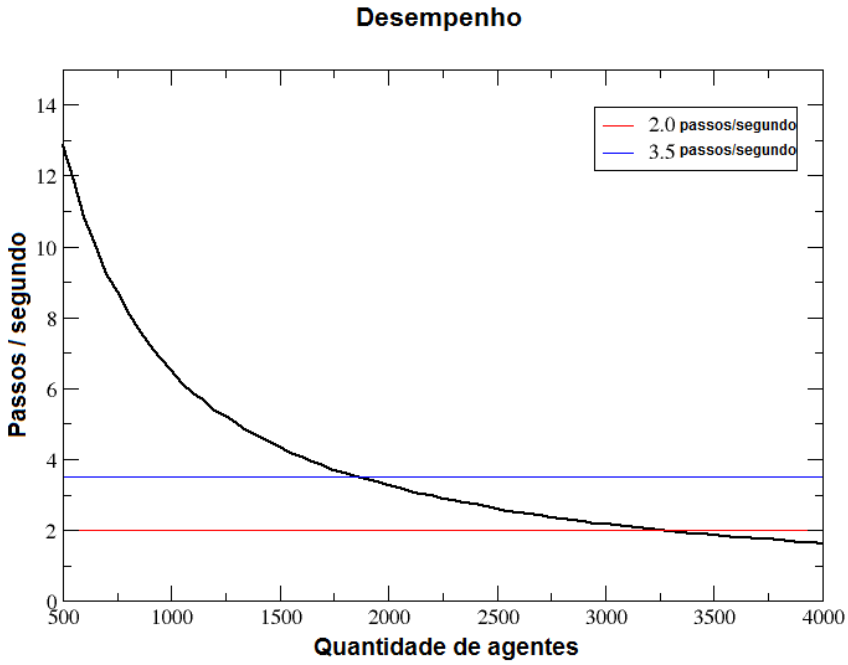
rimento medindo a freqüência de passos em função do número de agentes. O ambiente utilizado é o mesmo dos demais experimentos deste trabalho. A utilização desse ambiente aproxima-se do custo real do algoritmo, sendo que se for utilizado um ambiente 3D mais complexo, o custo deste último deverá ser avaliado.

A Figura 6.12-a mostra o resultado da simulação realizada por Dapper et al. (DAPPER, 2007) em uma máquina com processador ATHLON 64 3500+ 2.21GHz, 2.0 GB de RAM e placa gráfica nVidia GeForce 7800 GTX. Foram utilizadas 60 iterações na etapa do relaxamento da matriz local a cada passo. Neste caso, se for considerada a média de 2 passos por segundo, é possível utilizar até 300 agentes. Para 3.5 passos por segundo, até 200. A Figura 6.12-b apresenta os resultados obtidos com a reimplementação dessa técnica fazendo uso das otimizações propostas neste trabalho, utilizando também 60 iterações na etapa do relaxamento. Nesse caso, se for considerada a média de 2 passos por segundo, é possível utilizar até 3200 agentes. Para 3.5 passos por segundo, até 1800 agentes.

Na Figura 6.13, foi comparada a quantidade de células utilizadas (e conseqüentemente, memória) pela técnica proposta neste tabalho com a técnica previamente proposta por Dapper et al. A curva vermelha ilustra a técnica proposta do Dapper et al., utilizando um mapa local muito pequeno para cada agente, possuindo 10×10 células. A curva azul representa a técnica proposta neste trabalho. Os experimentos foram realizados não permitindo que a quantidade de agentes exceda 30% do tamanho do mapa do grupo (o espaço livre é necessário para os agentes poderem se mover livremente dentro do mapa). Foi escolhido o valor 30% por ser um valor razoável para fornecer uma boa porcentagem de área livre. Entretanto, mapas com mais de 30% de sua área ocupada por agentes produzem melhores resultados. A técnica proposta neste trabalho utiliza uma menor quantidade de células em todas as situações.



(a)



(b)

Figura 6.12: Comparação da técnica proposta por Dapper et al. (DAPPER; PRESTES; NEDEL, 2007) (a) e a reimplementação dessa técnica fazendo uso das otimizações propostas neste trabalho (b).

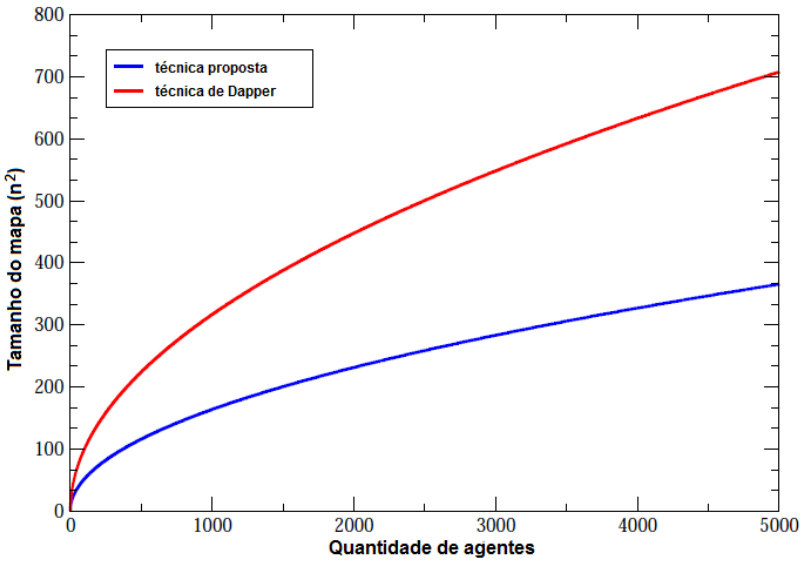


Figura 6.13: Comparação no tamanho do mapa entre a técnica proposta neste trabalho e a técnica proposta por Dapper et al. (DAPPER, 2007)

7 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou uma extensão do planejador BVP proposto por Dapper et al. (DAPPER, 2007) para possibilitar, de forma eficiente, a interação com grupos de agentes, permitindo a definição de objetivos dinamicamente e tornando o planejador BVP aplicável a ambientes virtuais interativos.

Alguns experimentos foram realizados com a técnica proposta por Dapper et al., com o intuito de otimizar e formalizar algumas propriedades percebidas.

Primeiramente, percebeu-se que a etapa que exigia mais processamento no planejador BVP era o processo de relaxamento. Neste trabalho, esse processo foi analisado com mais cuidado, concluindo-se que a maneira utilizada por Dapper et al. não era a forma ótima. Foi proposta uma maneira de acelerar a convergência do método, mantendo a coerência espacial dos mapas locais entre cada passo dos agentes.

Através de experimentos realizados, percebeu-se que os agentes poderiam exibir comportamentos interessantes, na maneira como interagem com o ambiente. Quanto mais informação sobre o ambiente estiver disponível para o agente, mais ele tenderá a mudar seu comportamento para evitar regiões com uma grande concentração de obstáculos. Isto poderia ser utilizado para gerar comportamentos encontrados na vida real. No mundo real, se alguém está com pressa, ou não está em um bom dia e avista um aglomerado de pessoas, a tendência dessa pessoa é desviar sua trajetória, evitando uma interação com o grupo. Caso contrário, essa pessoa pode passar entre os indivíduos da multidão, interagindo melhor com o grupo. O comportamento de uma pessoa muda conforme seu estado emocional ou suas necessidades. Mostrou-se que é possível gerar esses comportamentos variando alguns parâmetros como o tamanho da janela local.

Para ser possível utilizar o planejador BVP em ambientes interativos, foi proposta a substituição do planejador global por um planejador em alto-nível, baseado em esboço, possibilitando a interação do usuário com os agentes, característica que não era possível com a técnica de Dapper et al.

Para o controle do grupo de forma mais eficiente, foi proposta uma técnica onde se utiliza um mapa para cada grupo. Isso, além de utilizar menos memória, exige um menor tempo de processamento. Utilizando o planejador BVP, cada grupo possui um valor particular para $\vec{v}$ e ϵ, possibilitando um movimento individual diferenciado. A coerência fica a critério do usuário, podendo ser mantida ou não. Uma estratégia para manter a formação foi proposta, possibilitando ao usuário definir qualquer forma para formação, através de esboços.

A técnica proposta mostrou ser mais eficiente ao lidar com grupos que a técnica previamente proposta. Os resultados validaram a utilização dessa técnica em ambientes interativos em tempo-real na presença de humanos virtuais.

Algumas sugestões para trabalhos futuros são:

- Utilizar os parâmetros ϵ e $\mathbf{v}$ para gerar a distorção do potencial. Um protótipo foi desenvolvido onde se pode perceber que é possível obter os resultados da Seção 5.1 através da variação dos parâmetros ϵ e $\mathbf{v}$ dinamicamente;

- Analisar detalhadamente a deformação da formação, para que um grupo possa passar por obstáculos pequenos mantendo a formação;

- Desenvolver um algoritmo eficiente para GPUs. A definição de um algoritmo mais adequado para ser usado em placas gráficas pode diminuir o problema de desempenho, possibilitando a utilização de matrizes maiores para os mapas dos grupos ou o aumento do número de agentes na simulação. Essa idéia considera a tendência de evolução do hardware gráfico maior do que as dos processadores convencionais;

- Pesquisar estruturas de dados dedicadas para que a implementação na GPU se dê de forma mais eficiente;

- Estender a técnica proposta para que seja possível planejar um movimento efetivamente em 3D, de maneira eficiente. Isso seria útil para planejar o movimento de pássaros, ou de peixes, que possuem vários graus de liberdade e estão livres para se mover pelo ambiente em todas as dimensões.

REFERÊNCIAS

BALCH, T.; HYBINETTE, M. Social Potentials for Scalable Multirobot Formations. In: ICRA, APRIL 24-28, SAN FRANCISCO, CA, USA, 2000. **Anais...** IEEE, 2000.

BAYAZIT, O. B.; LIEN, J.-M.; AMATO, N. M. Better group behaviors in complex environments using global roadmaps. In: ICAL 2003: PROCEEDINGS OF THE EIGHTH INTERNATIONAL CONFERENCE ON ARTIFICIAL LIFE, 2003, Cambridge, MA, USA. **Anais...** MIT Press, 2003. p.362–370.

BAYAZIT, O.; LIEN, J.; AMATO, N. Better group behaviors using rule-based roadmaps. In: IN PROC. INT. WKSHP. ON ALG. FOUND. OF ROB. (WAFR) (NICE, FRANCE), 2002. **Anais...** [S.l.: s.n.], 2002.

BERG, J. P. van den; PATIL, S.; SEWALL, J.; MANOCHA, D.; LIN, M. Interactive navigation of multiple agents in crowded environments. In: SI3D, 2008. **Anais...** ACM, 2008. p.139–147.

CONNOLLY, C.; GRUPEN, R. On the Applications of Harmonic Functions to Robotics. **International Journal of Robotic Systems**, [S.l.], v.10, p.931–946, 1993.

DAPPER, F. **Planejamento de Movimento para Pedrestes Utilizando Campos Potenciais**. 2007. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul, Brasil.

DAPPER, F.; PRESTES, E.; IDIART, M. A. P.; NEDEL, L. P. Simulating Pedestrian Behavior with Potential Fields. **Proc. of CGI**, [S.l.], p.324–335, 2006.

DAPPER, F.; PRESTES, E.; NEDEL, L. P. Generating Steering Behaviors for Virtual Humanoids using BVP Control. **Proc. of CGI**, [S.l.], 2007.

DELOURA, M. **Game Programming Gems 1**. [S.l.]: Charles River Media, 2000. p.443–440.

DIETRICH, C. A.; NEDEL, L. P.; COMBA, J. L. D. A Sketch-Based Interface to Real-Time Strategy Games Based on a Cellular Automaton. In: **Game Programming Gems 7**. [S.l.]: Charles River Media, 2008. p.59–68.

FREDSLUND, J.; MATARIC, M. A general algorithm for robot formations using local sensing and minimal communications. **IEEE Transactions on Robotics and Automation**, [S.l.], v.18, n.5, p.837–846, 2002.

GERAERTS, R.; OVERMARS, M. A comparative study of probabilistic roadmaps planning. In: IN: ALGORITHMIC FOUNDATIONS O ROBOTICS, 2004. **Anais...** Springer-Verlag, 2004. p.43–57.

GOETZ, P. Too Many Clicks! Unit-Based Interfaces Considered Harmful. **http://www.gamasutra.com, last access 17/10/2007**, [S.l.], 2006.

HELBING, D. A Mathematical Model for the Behavior of Individuals in a Social Field. In: JOURNAL OF MATHEMATICAL SOCIOLOGY, 1994. **Anais...** [S.l.: s.n.], 1994. v.19, n.3, p.189–219.

HELBING, D.; MOLNAR, P. A social force model for pedestrian dynamics. **Phys. Rev. E**, [S.l.], v.51, p.4284–4286, 1995.

HOLLEMAN, C.; KAVRAKI, L. A framework for using the workspace medial axis in PRM planners. In: ICRA, 2000. **Anais...** [S.l.: s.n.], 2000. v.2, p.1408–1413.

KAMPHUIS, A.; OVERMARS, M. Motion planning for coherent groups of entities. **ICRA, San Diego, CA.**, [S.l.], 2004.

KAMPHUIS, A.; OVERMARS, M. H. Finding paths for coherent groups using clearance. In: SCA - SYMPOSIUM ON COMPUTER ANIMATION, 2004, Aire-la-Ville, Switzerland, Switzerland. **Anais...** Eurographics Association, 2004. p.19–28.

KAVRAKI, L. E.; LATOMBE, J.-C. Randomized Preprocessing of Configuration Space for Fast Path Planning. In: ICRA, 1994. **Anais...** [S.l.: s.n.], 1994. p.2138–2145.

KAVRAKI, L.; KOLOUNTZAKIS, M.; LATOMBE, J. Analysis of probabilistic roadmaps for path planning. In: IEEE INTERNAT. CONF. ROBOT. AUTOM, 1996. **Anais...** [S.l.: s.n.], 1996. v.4, p.3020–3025.

KAVRAKI, L.; SVESTKA, P.; LATOMBE, J.; OVERMARS, M. **Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces**. Stanford, CA, USA: [s.n.], 1994.

KHATIB, O. **Commande dynamique dans l'espace opérational des robots manipulaters en présence d'obstacles**. 1980. Tese (Doutorado em Ciência da Computação) — École Nationale Supérieure de l'Aéronatique et de l'Espace, France.

LAMIRAUX, F.; BONNAFOUS, D.; LEFEBVRE, O. Reactive path deformation for nonholonomic mobile robots. In: IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATIONS, 2004. **Anais...** [S.l.: s.n.], 2004. p.967–977.

LAVALLE, S. M. **Planning Algorithms**. [S.l.]: Cambridge University Press (também disponível em http://msl.cs.uiuc.edu/planning/), 2006.

LEROY, S.; LAUMOND, J.-P.; SIMéON, T. Multiple Path Coordination for Mobile Robots: a geometric algorithm. In: IJCAI '99: PROCEEDINGS OF THE SIXTEENTH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1999, San Francisco, CA, USA. **Anais...** Morgan Kaufmann Publishers Inc., 1999. p.1118–1123.

LI, T.-Y.; CHOU, H.-C. Motion planning for a crowd of robots. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 2003. **Anais...** [S.l.: s.n.], 2003. p.4215–4221.

LI, Y.; GUPTA, K. Large-Scale Agent Formations in Virtual Environments Using Linear Elastic Shapes. In: PROCEEDINGS OF THE 18TH INTERNATIONAL CONFERENCE ON COMPUTER ANIMATION AND SOCIAL AGENTS, 2005. **Anais...** [S.l.: s.n.], 2005.

MAZARAKIS, G. P.; AVARITSIOTIS, J. N. A prototype sensor node for footstep detection. In: WIRELESS SENSOR NETWORKS, 2005. PROCEEEDINGS OF THE SECOND EUROPEAN WORKSHOP ON, 2005. **Anais...** [S.l.: s.n.], 2005. p.415–418.

MUSSE, S. R.; THALMANN, D. Hierarchical Model for Real Time Simulation of Virtual Human Crowds. **IEEE TVCG**, Piscataway, NJ, USA, v.7, n.2, p.152–164, 2001.

MUSSE, S. R.; ULICNY, B.; AUBEL, A.; THALMANN, D. Groups and crowd simulation. In: ACM SIGGRAPH 2005 COURSES, 2005, New York, NY, USA. **Anais...** ACM Press, 2005. p.2.

NIEUWENHUISEN, D.; KAMPHUIS, A.; OVERMARS, M. H. High quality navigation in computer games. **Sci. Comput. Program.**, Amsterdam, The Netherlands, The Netherlands, v.67, n.1, p.91–104, 2007.

POTTINGER, D. Implementing Coordinated Movement. **Game Developer**, [S.l.], p.48–58, 1999.

POTTINGER, D. Coordinated Unit Movement. **Game Developer**, [S.l.], p.42–51, 1999.

PRESTES, E. **Navegação Exploratória baseada em Problemas de Valores de Contorno**. 2003. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul, Brasil.

REIF, J. H.; WANG, H. Social potential fields: a distributed behavioral control for autonomous robots. In: WAFR: PROCEEDINGS OF THE WORKSHOP ON ALGORITHMIC FOUNDATIONS OF ROBOTICS, 1995, Natick, MA, USA. **Anais...** A. K. Peters, 1995. p.331–345.

REYNOLDS, C. Flocks, herds and schools: a distributed behavioral model. In: SIGGRAPH, 1987, New York, NY, USA. **Anais...** ACM Press, 1987. p.25–34.

REYNOLDS, C. Steering Behaviors for Autonomous Characters. In: GAME DEVELOPERS CONFERENCE, 1999, San Francisco, California, USA. **Anais...** Miller Freeman Game Group, 1999. p.763–782.

REYNOLDS, C. Big fast crowds on PS3. In: ACM SIGGRAPH SYMPOSIUM ON VIDEOGAMES, 2006, New York, NY, USA. **Anais...** ACM Press, 2006. p.113–121.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence**: a modern approach. [S.l.]: Pearson Education, 2003.

SANCHEZ, G.; LATOMBE, J. Using a prm planner to compare centralized and decoupled planning for multi-robot systems. In: IN: IEEE INT. CONF. ON ROBOTICS AND AUTOMATION, 2002. **Anais...** [S.l.: s.n.], 2002. p.2112–2119.

SCHNEIDER, F.; WILDERMUTH, D. A potential field based approach to multi robot formation navigation. **Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003 IEEE International Conference on**, [S.l.], v.1, p.680–685 vol.1, 2003.

SONG, P.; KUMAR, V. A Potential Field Based Approach to Multi-Robot Manipulation. In: ICRA 2002, MAY 11-15, WASHINGTON, DC, USA, 2002. **Anais...** IEEE, 2002. p.1217–1222.

SUD, A.; ANDERSEN, E.; CURTIS, S.; LIN, M.; ; MANOCHA, D. Real-time Path Planning for Virtual Agents in Dynamic Environments. **IEEE Virtual Reality 2007**, [S.l.], 2006.

SVESTKA, P.; OVERMARS, M. Coordinated path planning for multiple robots. In: ROBOTICS AND AUTONOMOUS SYSTEMS, 1998. **Anais...** [S.l.: s.n.], 1998. v.23, p.125–152.

TREUILLE, A.; COOPER, S.; POPOVIC, Z. Continuum crowds. In: SIGGRAPH '06: ACM SIGGRAPH 2006 PAPERS, 2006, New York, NY, USA. **Anais...** ACM Press, 2006. p.1160–1168.

TREUILLE, A.; COOPER, S.; POPOVIć, Z. Continuum crowds. **ACM Trans. Graph.**, New York, NY, USA, v.25, n.3, p.1160–1168, 2006.

TREVISAN, M.; IDIART, M. A.; PRESTES, E.; ENGEL, P. M. Exploratory Navigation based on Dynamic Boundary Value Problems. **Journal of Intelligent and Robotic Systems**, [S.l.], v.45, p.101–114, 2006.

ULICNY, B.; THALMANN, D. Crowd simulation for interactive virtual environments and VR training systems. In: EUROGRAPHIC WORKSHOP ON COMPUTER ANIMATION AND SIMULATION, 2001, New York, NY, USA. **Proceedings...** Springer-Verlag New York: Inc., 2001. p.163–170.