UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

BIBLIOTECA PARA EXPLORAR ALGORITMOS DE PLANEJAMENTO DE MOVIMENTO UTILIZANDO CAMPOS POTENCIAIS NO SDK DO IOS 4

MARCELO ROBERTO FERRARI

BLUMENAU
2011

2011/1-28

MARCELO ROBERTO FERRARI

BIBLIOTECA PARA EXPLORAR ALGORITMOS DE PLANEJAMENTO DE MOVIMENTO UTILIZANDO CAMPOS POTENCIAIS NO SDK DO IOS 4

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Dalton Solano dos Reis, M.Sc. - Orientador

BLUMENAU
2011

2011/1-28

BIBLIOTECA PARA EXPLORAR ALGORITMOS DE PLANEJAMENTO DE MOVIMENTO UTILIZANDO CAMPOS POTENCIAIS NO SDK DO IOS 4

Por

MARCELO ROBERTO FERRARI

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Dalton Solano dos Reis, M.Sc. – Orientador, FURB
Membro:	Prof. Aurélio Faustino Hoppe, M.Sc. – FURB
Membro:	Prof. Paulo Cesar Rodacki Gomes, Dr. – FURB

Blumenau, 27 de Junho de 2011

Dedico este trabalho à minha esposa, ao meu filho, à minha mãe e meu irmão, e a todos que me ajudaram diretamente em sua realização.

AGRADECIMENTOS

A Deus, por iluminar-me e dar-me paciência durante esta jornada.

À minha esposa, Franciely, por sua compreensão, paciência e amor.

Ao meu filho, Vinícius, pelas horas de alegria.

À minha mãe, Geny, e meu irmão, Diego, pelo apoio e amor.

Aos meus amigos, pelo bom humor e apoio.

Ao meu orientador, Dalton Solano dos Reis, pela orientação, comprometimento e dedicação.

Não cabe a nós decidir o que é certo ou errado... A nós cabe apenas decidir o que fazer com o tempo que nos é dado.

John Ronald Reuel Tolkien

RESUMO

Este trabalho apresenta a implementação de uma biblioteca de planejamento de movimentos de um agente virtual para o iPhone OS 4, baseada em campos potenciais atrativos e repulsivos. Esta biblioteca também apresenta uma forma de resolver o problema de mínimos locais, através da utilização da solução numérica de problema de valor de contorno. Para demonstrar o funcionamento da biblioteca de planejamento de movimento, é disponibilizada uma aplicação exemplo, com alguns cenários pré-definidos.

Palavras-chave: Planejamento de movimento. Campos potenciais. Problema de valor de contorno. iOS 4.

ABSTRACT

This paper presents the implementation of a virtual agent's path planning library for iPhone OS 4, based on attractive and repulsive potential fields. This library also provides a way to solve the problem of local minima, using the numerical solution of boundary value problem. To demonstrate the operation of path planning library, a sample application is available with some predefined scenarios.

Key-words: Path planning. Potential fields. Boundary value problem. iOS 4.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

CUDA – *Compute Unified Device Architeture*

GPU – *Graphics Processor Unit*

HTML – *HyperText Markup Language*

ID - *Identity*

iOS 4 – *iPhone Operating System* 4

OpenGL ES – *Open Graphics Library for Embedded Systems*

PDF – *Portable Document Format*

PVC – Problema de Valor de Contorno

RF – Requisito Funcional

RNF – Requisito Não Funcional

RTF – *Rich Text Format*

SDK – *Software Development Kit*

UIKit – *User Interface Kit*

UML – *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

Atualmente muitos tipos de jogos são compostos por vários atores sintéticos que devem atuar como agentes autônomos. Agentes autônomos são personagens com a capacidade de desempenhar um papel para o meio ambiente com um comportamento e capacidade de improvisação. Para mover um agente em um mundo sintético é necessária uma representação semântica do ambiente, bem como a definição de uma posição inicial e um objetivo para este agente (FISCHER; SILVEIRA; NEDEL, 2009, p. 112-113).

Planejar o movimento de um agente autônomo, em seu modo mais simples, é basicamente planejar um caminho contínuo e livre de colisões com obstáculos, a partir de uma configuração inicial, para chegar até uma configuração final ou objetivo (PAN; LAUTERBACH; MANOCHA, 2010). No mundo real, considerando-se várias pessoas posicionadas em um mesmo local e com um mesmo objetivo, cada uma seguirá um caminho diferente, pois suas estratégias dependem de fatores como sua constituição física, personalidade, humor, raciocínio e urgência (FISCHER; SILVEIRA; NEDEL, 2009, p. 112-113).

Em um ambiente virtual é possível criar caminhos alternativos para agentes autônomos através da utilização da técnica de campos potenciais. Este campo potencial é gerado para incorporar obstáculos e objetivos e deve guiar os agentes autônomos até seus objetivos evitando estes obstáculos (KHATIB, 1980). Mas, a execução desta técnica pode levar a um problema de mínimos locais, que normalmente pode ser interpretado por um local côncavo e simétrico, onde a força de atração de seu objetivo é igual à força de repulsão dos obstáculos. (SILVEIRA, 2010, p. 24).

Existem várias maneiras de resolver o problema de mínimos locais. Uma delas é a utilização de campos potenciais baseados em Problema de Valor de Contorno (PVC), que utilizam funções harmônicas como base dos cálculos (DAPPER, 2007, p. 28; SILVEIRA, 2010, p. 31).

Diante do exposto, este trabalho apresenta uma biblioteca que realiza os planejamentos de movimentos para agentes autônomos utilizando a técnica de campos potenciais, para dispositivos móveis baseados na plataforma *iPhone Operating System* 4 (iOS 4). Esta biblioteca também consegue resolver o problema de mínimos locais utilizando a técnica de PVC.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho foi desenvolver uma biblioteca para explorar algoritmos de planejamento de movimento para o *Software Development Kit* (SDK) da plataforma iOS 4.

Os objetivos específicos do trabalho foram:

a) desenvolver um algoritmo para planejamento de movimentos de agentes autônomos utilizando a técnica de campos potenciais;
b) eliminar o problema de mínimos locais através de PVCs.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado em quatro capítulos. O primeiro capítulo trata da introdução, a justificativa do trabalho, o objetivo geral e específico.

No segundo capítulo é exposta a fundamentação teórica, onde é apresentada a definição de planejamento de movimentação, campos potenciais e PVCs. Também é apresentada a plataforma iOS 4, base para o funcionamento da biblioteca. O capítulo traz ainda uma descrição de alguns trabalhos correlatos.

No terceiro capítulo é abordado o desenvolvimento do trabalho. Nele são apresentados os requisitos a serem atendidos, a especificação da biblioteca, sua implementação, técnicas e ferramentas utilizadas, sua operacionalidade e resultados.

Ao final, no quarto capítulo são apresentadas a conclusão final e as sugestões de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentada a fundamentação teórica necessária para o desenvolvimento de um planejador de movimentos e seus principais métodos, sendo um deles campos potenciais, que é exposto na sequência. Logo após é apresentado o PVC e o método de Gauss-Seidel, utilizado para resolvê-lo bem como métodos para a movimentação do agente. Também é apresentada uma visão geral da plataforma iOS 4, que é a base para o funcionamento da biblioteca. Ao final são apresentados os trabalhos correlatos.

2.1 PLANEJAMENTO DE MOVIMENTO

O problema básico do planejamento de movimento consiste em encontrar um caminho livre de colisões com obstáculos estáticos, para movimentar um robô rígido ou articulado entre uma posição inicial e uma final. Este problema é puramente geométrico e aparentemente simples, mas para robôs com vários graus de liberdade é um problema computacionalmente complexo (LATOMBE, 1999, p. 1).

O planejamento de movimento geralmente é realizado em um agente autônomo de cada vez e é dividido em pelo menos duas tarefas. A primeira preocupa-se em calcular um caminho global e indica o melhor caminho para o agente autônomo até o objetivo. Esta tarefa normalmente ignora obstáculos e outros agentes autônomos em movimento. A segunda tarefa, também chamada de local, é encarregada de manter o agente autônomo no caminho global e em uma velocidade razoável, desviando dos obstáculos até alcançar o objetivo (ARIKAN; CHENNEY; FORSYTH, 2001).

O planejamento de movimento de agentes autônomos é frequentemente utilizado com uma combinação de métodos como *scripts*, grades, *roadmaps* e métodos locais reativos, também conhecidos como campos potenciais (SILVEIRA, 2010, p. 22). Uma breve descrição destes métodos seria:

a) planejamento baseado em *scripts*: onde os caminhos são previamente definidos de forma manual, baseados na percepção do artista. Por ser definido manualmente, este planejamento toma um grande tempo de desenvolvimento e pelo mesmo motivo as chances de um mesmo caminho ser tomado diversas vezes é grande

(NIEUWENHUISEN; KAMPHUIS; OVERMARS, 2006, p. 1-2);

b) planejamento baseado em grades de células: onde o ambiente é dividido em células que são marcadas como livres ou ocupadas. A partir disso, pode-se encontrar um caminho que leve o agente autônomo da posição atual até seu objetivo utilizando algum algoritmo de pesquisa em células, como por exemplo, o algoritmo *A** (RUSSELL; NORVIG, 1995, p. 796-798);
c) planejamento baseado em *roadmaps*: que consiste em uma espécie de mapa de todos os possíveis caminhos até o objetivo, que são gerados em uma fase anterior ao planejamento e durante a execução da animação o agente autônomo deve encontrar o caminho mais próximo de onde se encontra até seu objetivo (NIEUWENHUISEN; KAMPHUIS; OVERMARS, 2006, p. 4-11);
d) planejamento baseado em campos potenciais: onde é criado um campo potencial no ambiente, fazendo com que o agente chegue até o objetivo, evitando os obstáculos (KHATIB, 1980).

2.2 CAMPOS POTENCIAIS

Khatib (1980) propôs utilizar campos potenciais para planejar caminhos. Nesta técnica é gerado um campo potencial ou campo de força, que se expande por todo o ambiente e é criado a partir de forças repulsivas, geradas pelos obstáculos, e por forças atrativas, geradas pelos objetivos. O campo potencial gerado tem a finalidade de guiar os agentes autônomos até os objetivos através de caminhos alternativos e livres dos obstáculos.

A técnica de campos potenciais aplicada localmente é denominada técnica reativa, onde o objetivo é adaptar movimentos previamente computados aos obstáculos encontrados próximo aos agentes, os quais não foram previamente tratados durante o planejamento (SILVEIRA, 2010, p. 24).

Os campos potenciais são divididos em sete tipos diferentes (GOODRICH, 2007, p. 2-9):

a) atrativos: que são gerados pelos objetivos e têm a função de atrair os agentes autônomos;
b) repulsivos: que são gerados pelos obstáculos e têm a função de repelir os agentes autônomos;

c) uniformes: que permitem que agentes possam locomover-se próximos as paredes, que normalmente são interpretadas como obstáculos;
d) perpendiculares: que são potenciais perpendiculares as paredes ou obstáculos e faz com que os agentes autônomos nunca se aproximem;
e) tangenciais: que é uma espécie de espiral potencial ao redor do obstáculo a partir do centro do mesmo, podendo ser em sentido horário ou anti-horário;
f) randômicas: que são potenciais aleatórios para ajudar os agentes autônomos a não ficarem presos em locais mínimos;
g) *avoid past*: que servem para repelir o agente autônomo de um local mínimo através da geração de potencial repulsivo no caminho onde ele já percorreu. Em algum momento o potencial atrativo do objetivo será menor que o repulsivo, fazendo com que o agente autônomo seja repelido para fora deste local mínimo.

Na Figura 1 é apresentado um exemplo da junção dos campos potenciais atrativos, criados pelo objetivo, que esta sendo representado pelo quadrado verde. E por campos potenciais repulsivos criados pelo obstáculo, que esta sendo representado pelo conjunto de quadrados vermelhos. Também é exibido o caminho percorrido por um agente autônomo, representado pelo círculo azul, sofrendo a ação dos campos potenciais em seu movimento.

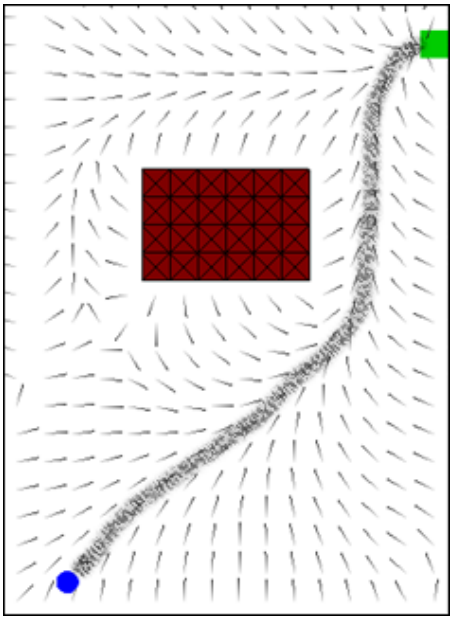


Figura 1 – Trajetória de um agente autônomo passando por um obstáculo e chegando a um destino

Geralmente a técnica utilizada localmente pode levar a mínimos locais. Um exemplo comum deste problema é um obstáculo simétrico e côncavo, que atrai o agente autônomo pelo menor potencial, este ficando preso em uma região onde a força de atração é igual à força de repulsão dos obstáculos. A melhor forma de evitar este problema é através da utilização de soluções numéricas, de uma equação diferencial parcial apropriada com condições de

contorno convenientes, ou seja, um PVC (DAPPER, 2007, p. 18-19).

2.2.1 Vantagens em relação às outras técnicas de planejadores

Em relação a planejadores baseados em *scripts*, *roadmaps* e grades de células pode-se citar as seguintes vantagens:

a) na técnica baseada em *scripts*, os caminhos são previamente definidos de forma manual, enquanto a técnica de campos potenciais define caminhos em tempo de execução, baseados na disposição dos obstáculos dentro do ambiente;
b) na técnica baseada em *roadmaps* existe um grande esforço inicial para que sejam encontrados todos os possíveis caminhos até o objetivo. Isto não acontece em campos potenciais, já que o único esforço realizado é o relaxamento da grade, matematicamente mais simples;
c) na técnica baseada em grades de células é necessário um grande processamento para a execução do algoritmos de pesquisa em células. Já em campos potenciais o caminho é encontrando baseando-se apenas do gradiente da célula atual.

Além destas vantagens, existe a possibilidade de trabalhar com mapas global e local e obstáculos dinâmicos, sendo que no mapa global são mapeados apenas obstáculos estáticos, enquanto no local são mapeados os obstáculos estáticos e dinâmicos. Para maiores informações sobre o assunto, consulte Silveira (2010, p. 31-49).

2.3 PROBLEMA DE VALOR DE CONTORNO

PVCs são normalmente equações diferenciais ordinárias ou equações diferenciais parciais, que possuem valores atribuídos no limite físico do domínio em que o problema é especificado (WEISSTEIN, 2010).

A primeira proposta para planejamento de movimento utilizando PVCs foi feita por Connolly, Burns e Weiss (1990, p. 2102-2103) através da utilização de funções harmônicas. Uma função $p(r)$ é dita harmônica em um domínio $\Omega \subset \Re^n$ quando ela satisfaz a equação de Laplace (Quadro 1).

$$\nabla^2 p(r) = \sum_{i=1}^{n} \frac{\partial^2 p(r)}{\partial x_i^2} = 0$$

onde ∇^2 representa o gradiente e x_i representa a soma dos potencias das células vizinhas

Fonte: Connolly, Burns e Weiss (1990, p. 2102).

Quadro 1 – Equação de Laplace

A equação de Laplace é utilizada em diversas áreas da física, como dinâmica de fluídos, gravitação e eletromagnetismo, por ser capaz de descrever campos de força. Ao ser aplicada, obtêm-se linhas equipotenciais, e se em seguida forem calculadas as linhas de fluxo, normais às linhas equipotenciais, então é possível definir caminhos para que uma partícula possa se mover de um ponto com potencial maior, para um ponto com potencial menor.

Trevisan et al. (2006, p. 103-114) apresentaram um *framework* para navegação exploratória, baseado em funções harmônicas de potencial que não possuem locais mínimos. Estas funções são geradas através da solução numérica de um PVC usando as condições de contorno de Dirichlet, satisfazendo a função apresentada no Quadro 2.

$$\nabla^2 p(r) + \epsilon v . \nabla p(r) = 0$$

onde v é um vetor unitário e representa os eixos X e Y no ambiente bidimensional e ϵ é um valor escalar

Fonte: Dapper, Prestes e Nedel (2007, p. 2).

Quadro 2 – Equação utilizando condições de contorno de Dirichlet

A equação do Quadro 2 pode ser reduzida para $\nabla^2 p(r) = 0$ quando $\epsilon = 0$. O que corresponde à equação de Laplace, obtendo-se assim um planejador baseado em funções harmônicas.

Em tarefas exploratórias, o uso dos termos v e ϵ ajudam o agente autônomo a realizar uma exploração mais eficiente, reduzindo o tempo necessário para adquirir informações em ambientes esparsos. Isso é feito alterando o campo potencial através dos parâmetros v e ϵ que indicam ao agente uma direção preferencial para a exploração (TREVISAN et al., 2006, p. 103-114).

A abordagem utilizada para resolver numericamente um PVC 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 $p_{i,j}^t$ 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 atrai os agentes. O potencial das células livres é calculado usando o método de relaxamento de Gauss-Seidel (SILVEIRA, 2010, p. 32).

2.3.1 Método de Gauss-Seidel

O método de Gauss-Seidel é um método iterativo para resolução de sistemas de equações lineares. Ele consiste em substituir o valor do potencial de cada célula livre pela média simples de seus vizinhos simultaneamente (SILVEIRA, 2010 p. 32-33). No Quadro 3 é apresentada a equação do método de Gauss-Seidel.

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

Onde p representa o valor do potencial de determinada célula, $p_c = p_{i,j}^{t+1}$, $p_t = p_{i,j-1}^{t+1}$, $p_b = p_{i,j+1}^{t}$, $p_l = p_{i-1,j}^{t+1}$, $p_r = p_{i+1,j}^{t}$ $v = (v_x, v_y)$, sendo $v = (v_x, v_y)$, $v_{x,y} \in (-1, 1)$ e $\epsilon \in (-2, 2)$.

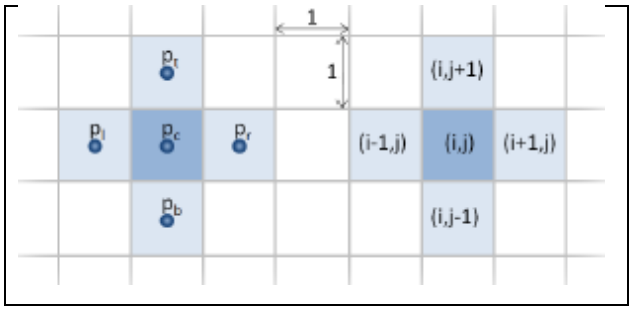
$\frac{p_t + p_b + p_l + p_r}{4}$ equivale à $\nabla^2 p(r)$, da equação do Quadro 2 e

$\frac{\epsilon}{8}\left((p_r - p_l)v_x + (p_b - p_t)v_y\right)$ equivale à $\epsilon v.\nabla p(r)$, da equação do Quadro 2.

Fonte: Dapper, Prestes e Nedel (2007, p. 3), Silveira (2010, p. 33).

Quadro 3 – Equação de Gauss-Seidel

As células da equação podem ser observadas na Figura 2.



Fonte: Silveira (2010, p. 33).

Figura 2 – Localização das células de uma grade regular

2.4 MOVIMENTAÇÃO DO AGENTE

Após a computação do campo potencial, o agente se move seguindo a direção definida pelo gradiente descendente de seu potencial na posição atual (i, j) definido pela equação do Quadro 4 (SILVEIRA, 2010, p. 34).

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

sendo o tamanho da discretização 1 unidade e p representa o valor do potencial de determinada célula

Fonte: Silveira (2010, p. 34).

Quadro 4 – Equação do gradiente descendente

Como a equação é formalmente completa, se existir um caminho em direção ao objetivo, ele será encontrado. Entretanto, este método não gera comportamentos realísticos, como os observados no mundo real. Uma das razões disso ocorrer é que o agente muda sua direção baseado apenas no gradiente descendente de sua região (SILVEIRA, 2010, p. 34).

Dapper, Prestes e Nedel (2007, p. 4-5) propuseram uma solução para esse problema ajustando a posição do agente de acordo com a equação apresentada no Quadro 5.

$$\Delta d = v\big(cos(\varphi^t), sin(\varphi^t)\big)$$

onde v é a velocidade máxima do agente e $\varphi^t = \eta\, \varphi^{t-1} + (1 - \eta)\, \zeta^t$, sendo $\eta \in [0, 1)$ e ζ^t representando a orientação do gradiente descendente representado pela equação do Quadro 4

Fonte: Dapper, Prestes e Nedel (2007, p. 4), Silveira (2010, p. 35).

Quadro 5 – Equação da movimentação do agente com controle de velocidade

Segundo Silveira (2010, p. 35) quando η = 0, o agente ajusta sua orientação usando apenas a informação do gradiente. Se η = 0.5, a direção anterior do agente (φ^{t-1}) e a direção do gradiente influenciam igualmente no cálculo. E quando $\eta \to 1$, o agente reage lentamente, aumentando a chance de colisão com obstáculos.

Para resolver este problema foi adicionado um controle de velocidade, apresentado na equação do Quadro 6 (DAPPER, PRESTES, NEDEL, (2007), p. 5; SILVEIRA, 2010, p. 35).

$$\Delta d = v\big(cos(\varphi^t), sin(\varphi^t)\big)\, \Psi\, (|\varphi^{t-1} - \zeta^t|)$$

onde a função $\Psi : \Re \rightarrow \Re$ é

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

Fonte: Dapper, Prestes e Nedel (2007, p. 4), Silveira (2010, p. 35).

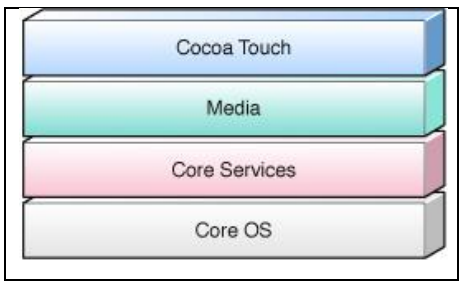
Quadro 6 - Equação da movimentação do agente com controle de velocidade baseado na direção anterior do agente e no gradiente atual

Se $|\varphi^{t-1} - \zeta^t|$ for maior que $\pi/2$, então existe chances de colisão e a função retorna 0, fazendo com que o agente pare. Caso contrário, a velocidade do agente varia de acordo com o risco de colisão. Em regiões com muitos obstáculos, os agentes tendem a mover-se mais lentamente (SILVEIRA, 2010, p. 35-36).

2.5 PLATAFORMA IOS

A plataforma iOS é um conjunto de sistema operacional e tecnologias para executar aplicativos nativamente em dispositivos móveis, como o iPad, o iPhone e o iPod Touch. Esta plataforma herda várias tecnologias e funcionalidades da plataforma Mac OS X e conta também com tecnologias que estão apenas no iOS, como uma interface de multitoques e acelerômetro. Para o desenvolvimento de aplicações na plataforma iOS é utilizado o seu SDK, que possui ferramentas para testes, execução e depuração. Dentro do SDK existe uma ferramenta chamada Xcode, que é utilizada para a edição de código. Nela está disponível o recurso de simulação do iPhone (APPLE INC, 2010a).

O núcleo da plataforma iOS é composta por 4 camadas conforme visto na Figura 3, sendo que no topo delas encontra-se a camada *Cocoa Touch*, que é a camada onde a maioria das aplicações baseadas em Objective-C são desenvolvidas (APPLE INC, 2010a).



Fonte: Apple Inc (2010a).

Figura 3 – Visão geral das camadas da plataforma iOS

Nas camadas *Core OS* e *Core Services* estão as interfaces fundamentais do iOS, incluindo as utilizadas para acessar arquivos, tipos de dados de baixo nível, *sockets* de rede entre outros. Estas interfaces são na sua grande maioria baseadas em C e incluem tecnologias como a *Core Foundation* e *SQLite*. A camada *Media* contém as tecnologias fundamentais utilizadas em desenhos 2D e 3D, além de áudio e vídeo. Esta camada também inclui tecnologias baseadas em C, como o *Open Graphics Library for Embedded Systems* (OpenGL ES) e também o *Core Animation* que é um avançado mecanismo de animação, baseado em Objective-C (APPLE INC, 2010a).

Os *frameworks* existentes nestas camadas fornecem a infraestrutura fundamental para as aplicações. Os *frameworks Foundation* e *User Interface Kit* (*UIKit*), por exemplo, fornecem os principais serviços utilizados por todas as aplicações desenvolvidas para iOS (APPLE INC, 2010a). No *framework UIKit* estão as classes necessárias para construir e manipular a interface para o usuário de aplicativos que rodam no iOS. Ele fornece manipulação de eventos, janelas, exibições e controles projetados especificamente para uma interface de tela sensível ao toque (APPLE INC, 2010b).

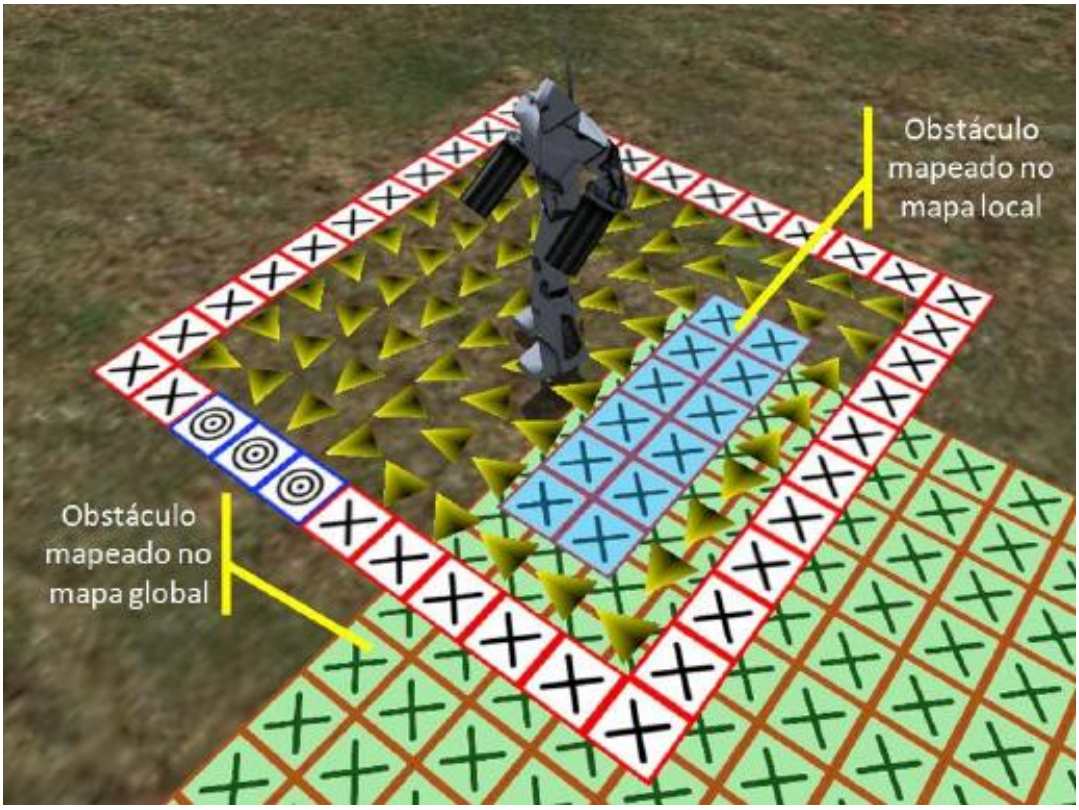
O OpenGL ES, encontrado na camada *Media*, é uma biblioteca gráfica para manipulação de objetos 2D e 3D baseada no OpenGL tradicional, para computadores *desktops*. É composto por um conjunto de centenas de funções, que fornecem acesso a vários recursos de hardware de vídeo. Utilizando-se esta biblioteca, existe a possibilidade de trabalhar com desenhos de forma vetorial, ou seja, definida por vértices, ou com mapas de bits, que são definidos pixel a pixel (KHRONOS GROUP, 2010; OPENGL, 2010).

2.6 TRABALHOS CORRELATOS

O tema planejamento de movimento é bastante discutido na área de computação gráfica e de desenvolvimento de jogos. Existem vários trabalhos e artigos científicos sobre este assunto. Dentre eles, foram escolhidos os trabalhos descritos por Fischer (2008), Ferrari (2009) e por Mafra (2004).

O trabalho de Fischer (2008) é um planejador de caminhos baseado em campos potenciais. O algoritmo descrito neste trabalho consegue planejar caminhos para vários agentes autônomos ao mesmo tempo e tem a possibilidade de atribuir não só um objetivo, como uma lista de objetivos para cada um dos agentes autônomos do ambiente. Também

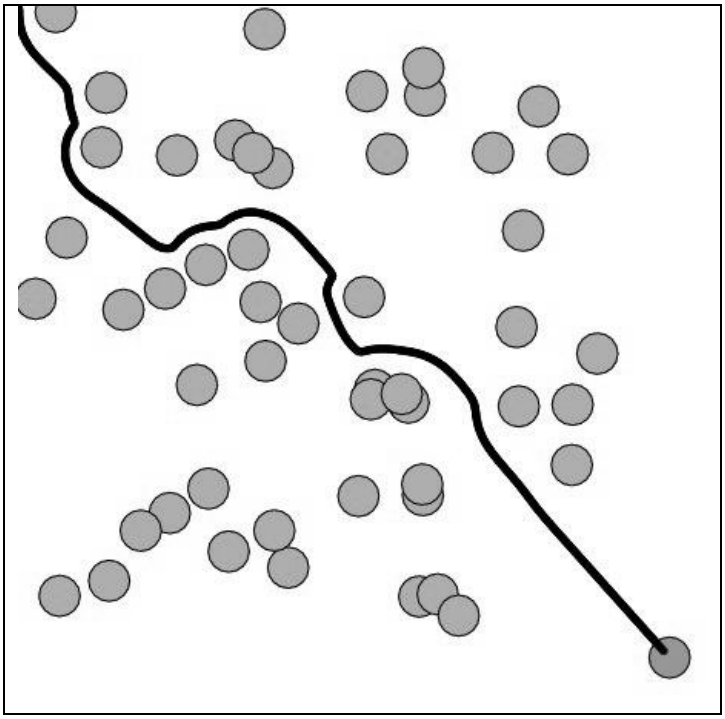
existe a possibilidade de dividir os agentes em subgrupos, onde cada subgrupo possui um objetivo em comum. Também possui definições de mapa potencial global e local, sendo que no mapa global são mapeados apenas obstáculos estáticos, enquanto no local são mapeados os obstáculos estáticos e dinâmicos. Ele foi desenvolvido em C++ e na linguagem *Compute Unified Device Architeture* (CUDA), que é uma linguagem para utilização de paralelismo na *Graphics Processor Unit* (GPU) das placas de vídeo, que segundo Fischer (2008), aumenta o desempenho do algoritmo. Por fim, consegue evitar caminhos mínimos através da utilização da técnica de PVCs para os campos potenciais, utilizando o método de Gauss-Seidel. Na Figura 4 é possível observar o mapa global e o mapa local para um agente.



Fonte: Fischer (2008, p. 36).

Figura 4 – Mapa global e mapa local de um agente

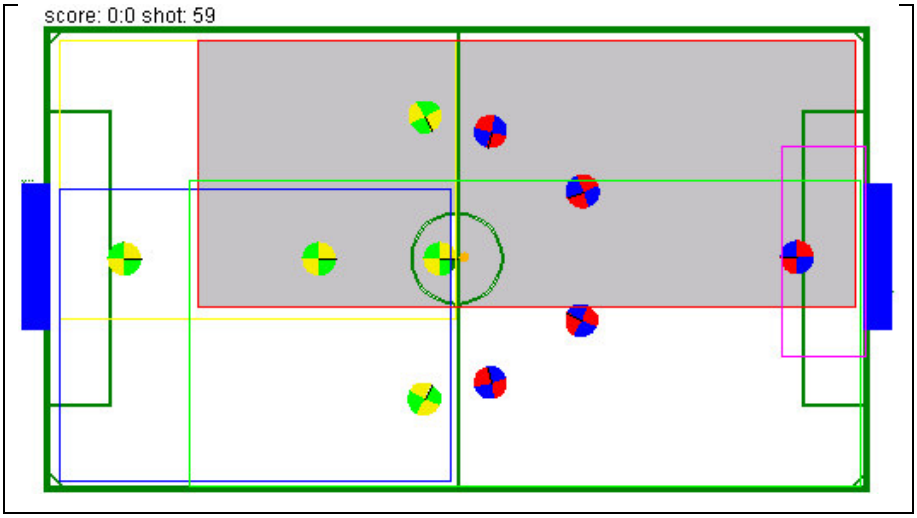
Já o trabalho de Ferrari (2009) é um planejador de movimentos que contém uma família de campos potenciais para melhorar o controle do movimento do agente autônomo. Os potenciais tanto dos objetivos quanto dos obstáculos podem ser parametrizados e possuem comprimento de escala e grau variáveis. Por meio destes parâmetros é possível controlar a distância mínima que o agente deve manter-se de um obstáculo. Com essa parametrização, aumentando a força do potencial, ele também consegue evitar os locais mínimos. Na Figura 5 pode-se observar a movimentação de um agente em um ambiente com distribuição uniforme de obstáculos.



Fonte: Ferrari (2009).

Figura 5 – Movimentação do agente em um ambiente com obstáculos distribuídos uniformemente

Por fim o trabalho de Mafra (2004) é um protótipo de times de futebol de robôs utilizando robótica baseada em comportamento através de campos potenciais. O protótipo foi baseado no simulador TBSim do ambiente TeamBots, que é utilizado por pesquisadores em robótica móvel na área de sistemas multi-agentes. O campo potencial é gerado com base na área de atuação do robô dentro do campo de futebol, em outros robôs interpretados como obstáculos e sobre um ponto de atração ou objetivo, que não deixa o robô se afastar de sua área de atuação. Na Figura 6 é possível visualizar área de atuação da lateral direita.



Fonte: Mafra (2004, p. 50).

Figura 6 – Área de atuação da lateral direita

2.6.1 Comparativo entre os trabalhos correlatos

No Quadro 7 é apresentado um comparativo entre as principais características dos trabalhos correlatos.

	FISCHER 2008	**FERRARI 2009**	**MAFRA 2004**
Suporte a mais de 1 agente simultaneamente	Sim	Não	Sim
Suporte a mais de 1 objetivo sequencialmente	Sim	Não	Não
Movimentação próxima ao mundo real	Sim	Não	Não
Evita mínimos locais	Sim	Sim	Não se aplica
Suporte a plataforma móvel iOS 4	Não	Não	Não

Quadro 7 – Comparativo entre os trabalhos correlatos

3 DESENVOLVIMENTO

Este capítulo detalha as etapas do desenvolvimento da biblioteca. São apresentados os requisitos, a especificação e a implementação da mesma, mencionando as técnicas e ferramentas utilizadas. Também é apresentada a operacionalidade da biblioteca e os resultados obtidos.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Abaixo são apresentados os Requisitos Funcionais (RF) e Requisitos Não-Funcionais (RNF) a serem atendidos pela biblioteca desenvolvida:

a) permitir que seja informado um ponto de saída (RF);
b) permitir que sejam informados um ou mais pontos de chegada (RF);
c) disponibilizar um aplicativo exemplo para permitir a execução de testes na biblioteca (RF);
d) permitir que sejam visualizados os mapas virtuais e a movimentação dos agentes autônomos (RF);
e) ser implementado utilizando o ambiente de desenvolvimento Xcode (RNF);
f) ser implementado utilizando a linguagem Objective-C (RNF);
g) ser compatível com a plataforma móvel iOS 4 (RNF);
h) possuir documentação (RNF);
i) ser implementado utilizando a técnica de campos potenciais (RNF);
j) deverá evitar o problema de mínimos locais através da utilização da técnica de PVC (RNF).

3.2 ESPECIFICAÇÃO

A especificação do presente trabalho foi desenvolvida utilizando alguns diagramas da *Unified Modeling Language* (UML), utilizando a ferramenta Enterprise Architect 7.0.817 para

a elaboração dos diagramas de casos de uso, de classes, de sequência e de atividades.

3.2.1 Diagrama de casos de uso

Como este trabalho é um planejador de movimentos para ser utilizado numa aplicação, o desenvolvedor da aplicação deverá ser considerado como o `Usuário`. A Figura 7 apresenta o diagrama de casos de uso com as interações do `Usuário` com a biblioteca.

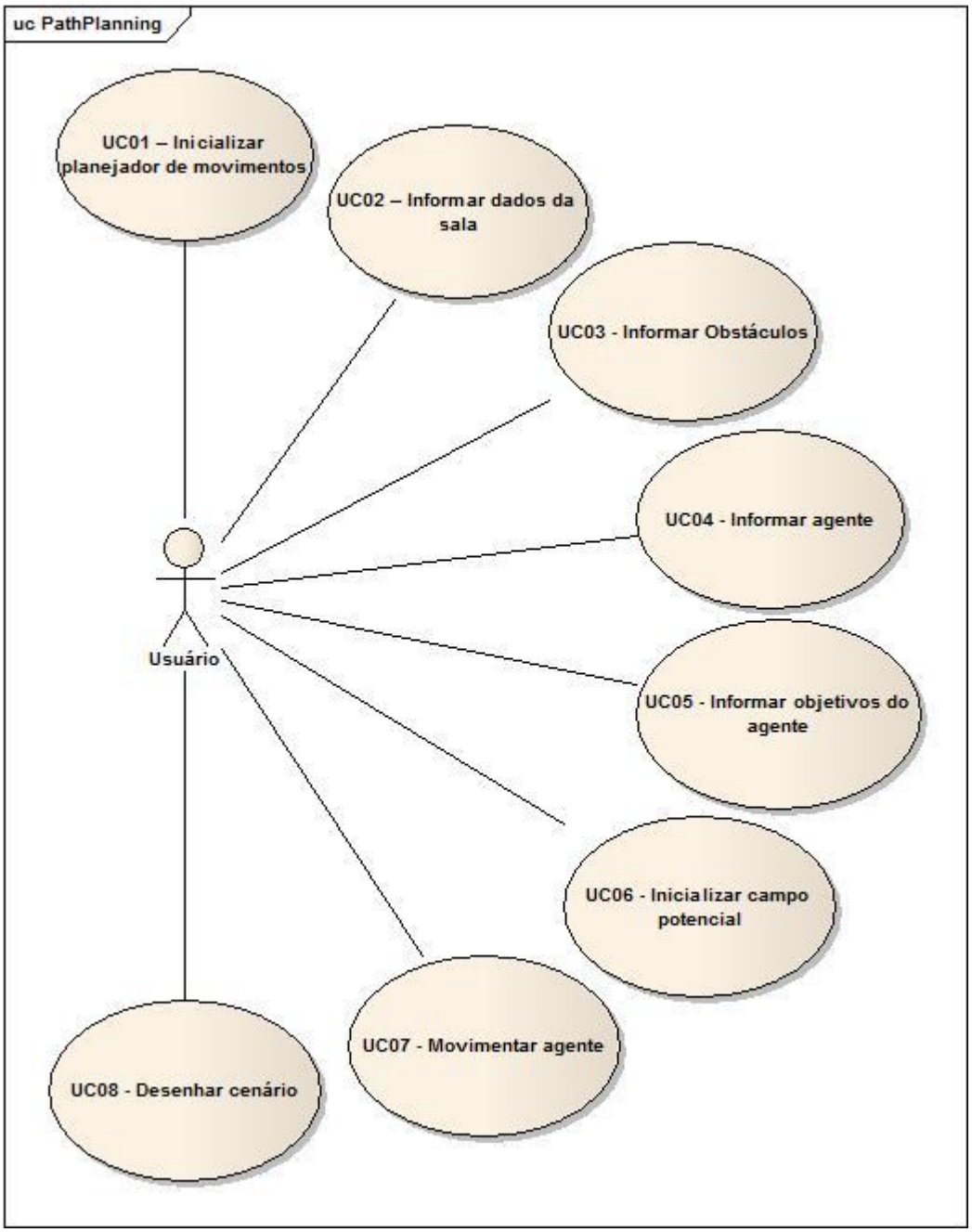


Figura 7 – Diagrama de casos de uso do planejador de movimentos

O caso de uso `UC01` (Quadro 8) descreve como o usuário irá inicializar o planejador de

movimentos. Este caso possui apenas um cenário principal.

UC01 – Inicializar planejador de movimentos	
Cenário principal	1) O usuário cria uma instância da classe `PathPlanning` chamando o método `init`. 2) O usuário pode informar o tamanho das células da grade através dos atributos `verticalResolution` e `horizontalResolution`.
Pós-condições	A instância criada está pronta para planejar movimentos de agentes virtuais.

Quadro 8 – Detalhamento do caso de uso UC01

O segundo caso de uso `UC02` (Quadro 9) demonstra como o usuário informa o ambiente ou sala especificando sua posição inicial. Este caso possui apenas um cenário principal.

UC02 – Informar dados da sala	
Pré-condições	O usuário deve ter inicializado o planejador de movimento.
Cenário principal	1) O usuário chama o método `createRoomWithSmallerX:andGreaterX:andSmallerY:andGreaterY:andSmallerZ:andGreaterZ:` da classe `PathPlanning`.
Pós-condições	A sala esta adicionada no planejador de caminhos.

Quadro 9 – Detalhamento do caso de uso UC02

O terceiro caso de uso `UC03` (Quadro 10) demonstra de que forma o usuário pode informar a posição dos obstáculos dentro da sala. Este caso possui apenas um cenário principal.

UC03 – Informar obstáculos	
Pré-condições	O usuário deve ter informado os dados da sala.
Cenário principal	1) O usuário chama o método `addObstacleWithSmallerX:andGreaterX:andSmallerY:andGreaterY:andSmallerZ:andGreaterZ:` da classe `PathPlanning`.
Pós-condições	Os obstáculos estão adicionados à sala.

Quadro 10 – Detalhamento do caso de uso UC03

O quarto caso de uso `UC04` (Quadro 11) demonstra como o usuário informa o agente e especifica sua posição inicial e a velocidade em que irá se movimentar. Este caso possui apenas um cenário principal.

UC04 – Informar agente	
Pré-condições	O usuário deve ter informado os dados da sala.
Cenário principal	1) O usuário chama o método `addRobotWithX:andY:andZ:andStepMove:andIDRobot:` da classe `PathPlanning`.
Pós-condições	O agente está adicionado à sala.

Quadro 11 – Detalhamento do caso de uso UC04

No quinto caso de uso `UC05` (Quadro 12) é possível verificar como o usuário informa os objetivos do agente. Este caso possui apenas um cenário principal.

UC05 – Informar objetivos do agente	
Pré-condições	O usuário deve ter informado o agente.
Cenário principal	1) O usuário chama o método `addWayPointWithX:andY:andIDRobot:` da classe `PathPlanning`.
Pós-condições	Os objetivos estão associados ao agente.

Quadro 12 – Detalhamento do caso de uso UC05

O sexto caso de uso `UC06` (Quadro 13) demonstra como o usuário inicia os cálculos do campo potencial do planejador de caminhos. Este caso possui apenas um cenário principal.

UC06 – Inicializar campo potencial	
Pré-condições	O usuário deve ter informado os objetivos do agente.
Cenário principal	1) O usuário chama o método `start` da classe `PathPlanning`.
Pós-condições	O campo potencial está calculado e pronto para a movimentação do agente.

Quadro 13 – Detalhamento do caso de uso UC06

No sétimo caso de uso `UC07` (Quadro 14) é apresentada a forma de movimentar o agente dentro da sala. Este caso possui apenas um cenário principal.

UC07 – Movimentar agente	
Pré-condições	O usuário deve ter inicializado os cálculos.
Cenário principal	1) O usuário chama o método `movementRobots` da classe `PathPlanning`.
Pós-condições	A posição e a velocidade do agente são atualizadas levando em consideração o campo potencial da sala.

Quadro 14 – Detalhamento do caso de uso UC07

No oitavo e último caso de uso `UC08` (Quadro 15) é demonstrada a forma de desenhar o cenários e seus objetos. Estes objetos são o agente e seus objetivos e os obstáculos além do campo potencial. Este caso possui apenas um cenário principal.

UC08 – Desenhar cenário	
Pré-condições	O usuário deve ter inicializado os cálculos.
Cenário principal	1) O usuário chama o método `movementRobots` da classe `PathPlanning`.
Pós-condições	A posição e a velocidade do agente são atualizadas levando em consideração o campo potencial da sala.

Quadro 15 – Detalhamento do caso de uso UC08

3.2.2 Diagrama de classes

A Figura 8 apresenta as classes do planejador de caminhos e da aplicação exemplo. Na sequencia é feita uma análise sobre a funcionalidade de cada uma delas. Algumas classes auxiliares foram omitidas para uma melhor visualização do diagrama, não comprometendo o entendimento do mesmo.

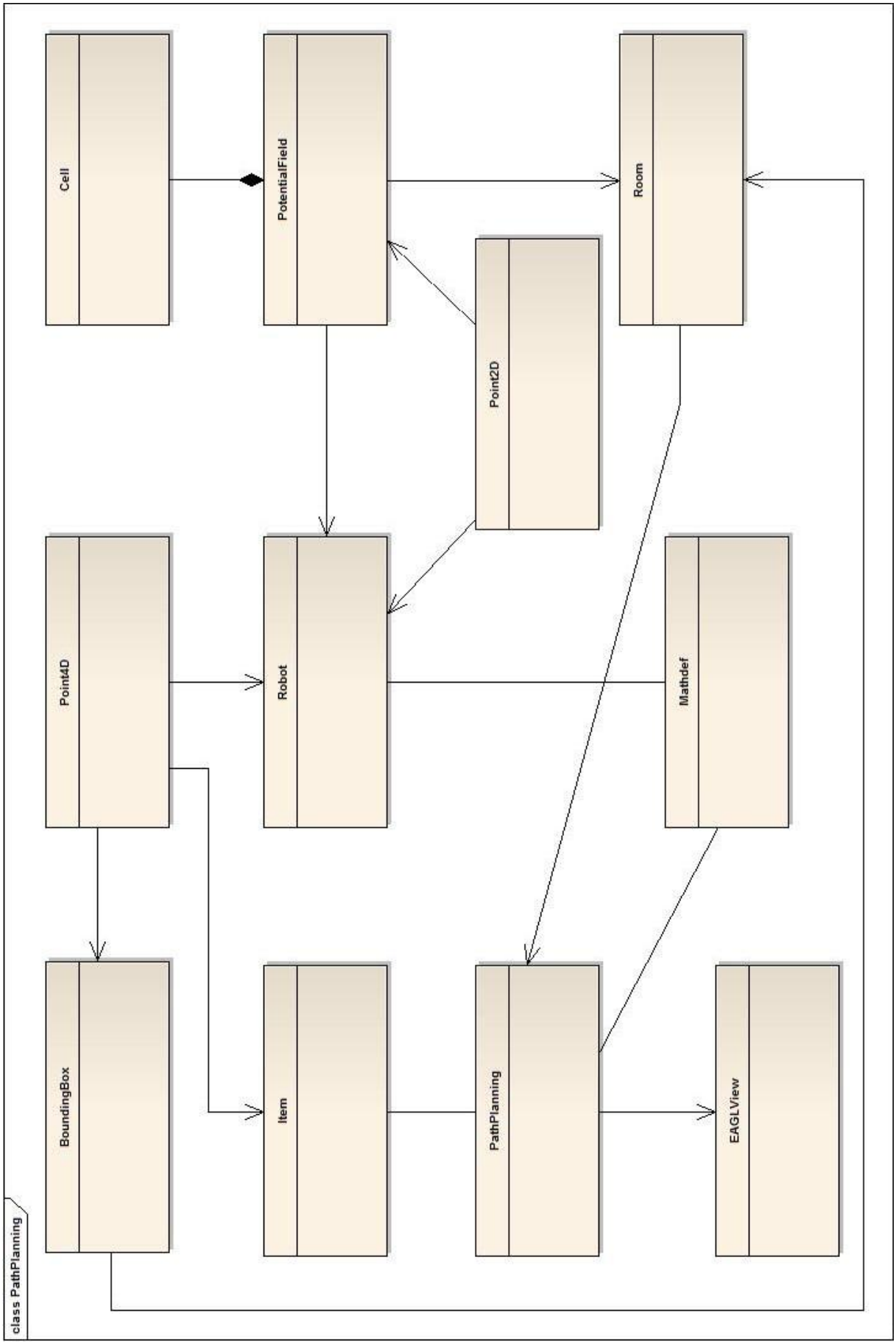


Figura 8 – Diagrama de classes do *PathPlanning*

3.2.2.1 Classes `Point2D`, `Point4D` e `BoundingBox`

As classes `Point2D` e `Point4D` representam respectivamente orientações no espaço bidimensional e tridimensional no espaço virtual. A classe `Point4D` também é utilizada para representar visualmente a movimentação do agente. Já a classe `BoundingBox` representa o espaço retangular tridimensional ocupado por um objeto.

Nas Figura 9, Figura 10 e Figura 11 são exibidos respectivamente os diagramas das classes `Point2D`, `Point4D` e `BoundingBox`.

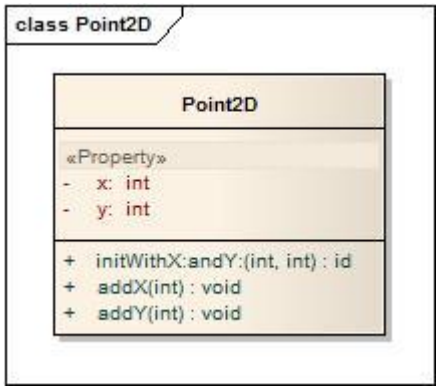


Figura 9 – Diagrama da classe `Point2D`

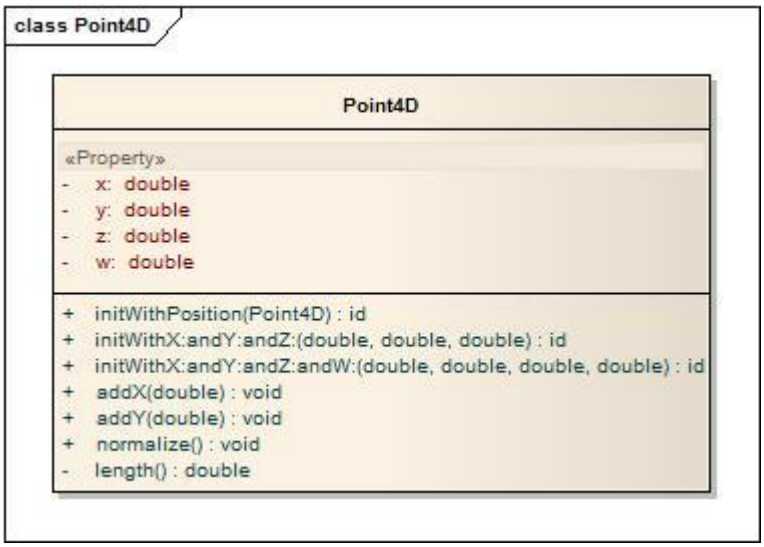


Figura 10 – Diagrama da classe `Point4D`

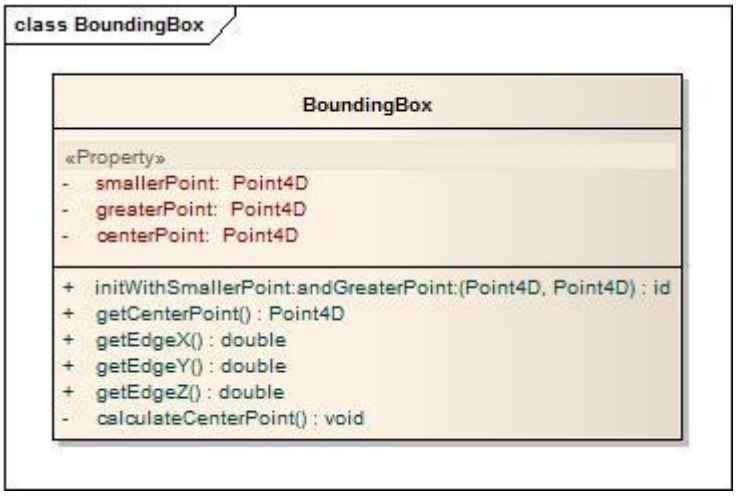


Figura 11 - Diagrama da classe `BoundingBox`

3.2.2.2 Classes `Room` e `Cell`

A classe `Cell` representa uma área da grade do campo potencial. Cada célula possui um valor potencial e um tipo, que pode ser livre, obstáculo ou objetivo. A classe `Room` representa o ambiente em que o campo potencial será criado. Esta classe disponibiliza o método `startPotentialFieldWithHorizontalSize:andVerticalSize:andHorizontalResolution:andVerticalResolution:`, que inicializa o campo potencial da mesma, baseado nos obstáculos e no primeiro objetivo do agente.

Nas Figura 12 e Figura 13 são exibidos respectivamente os diagramas das classes `Cell` e `Room`.

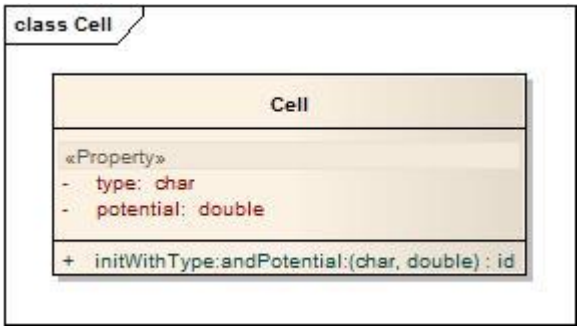


Figura 12 - Diagrama da classe `Cell`

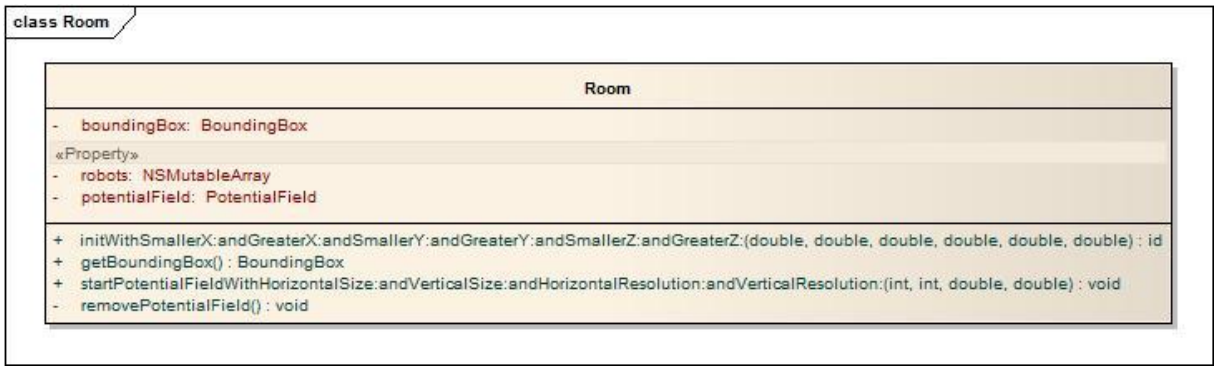


Figura 13 - Diagrama da classe `Room`

3.2.2.3 Classe `Robot`

A classe `Robot` representa os agentes virtuais que se movimentam em um ambiente virtual. A classe possui uma coleção de objetivos do robô, além de controles de posicionamento bidimensional e tridimensional. Esta classe também disponibiliza o método `move`, que executa todos os cálculos necessários para a movimentação do agente.

Na Figura 14 é exibido o diagrama da classe `Robot`.

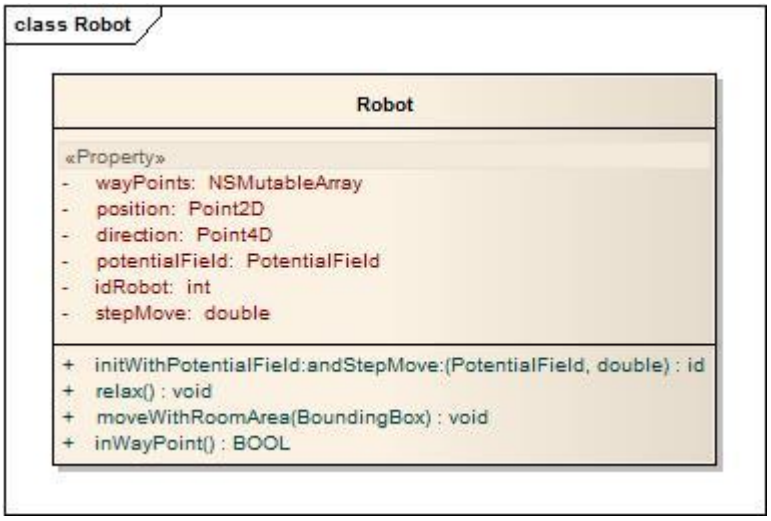


Figura 14 - Diagrama da classe `Robot`

3.2.2.4 Classe `PotentialField`

A classe `PotentialField` é a principal classe da biblioteca e representa o campo potencial utilizado para a movimentação do agente. O campo potencial é formando por uma grade com células retangulares, onde cada célula pode ser livre, possui um obstáculo ou ser

um objetivo. Para guardar estas informações é utilizada uma coleção da classe `Cell`. Este campo potencial é formado pelos tipos atrativos e repulsivos, sendo os atrativos os objetivos e os repulsivos os obstáculos.

Seu método mais importante, `relax`, efetua os cálculos necessários para a geração do campo potencial, utilizando a equação de Gauss-Seidel apresentada no Quadro 3.

Na Figura 15 é exibido o diagrama da classe `PotentialField`.

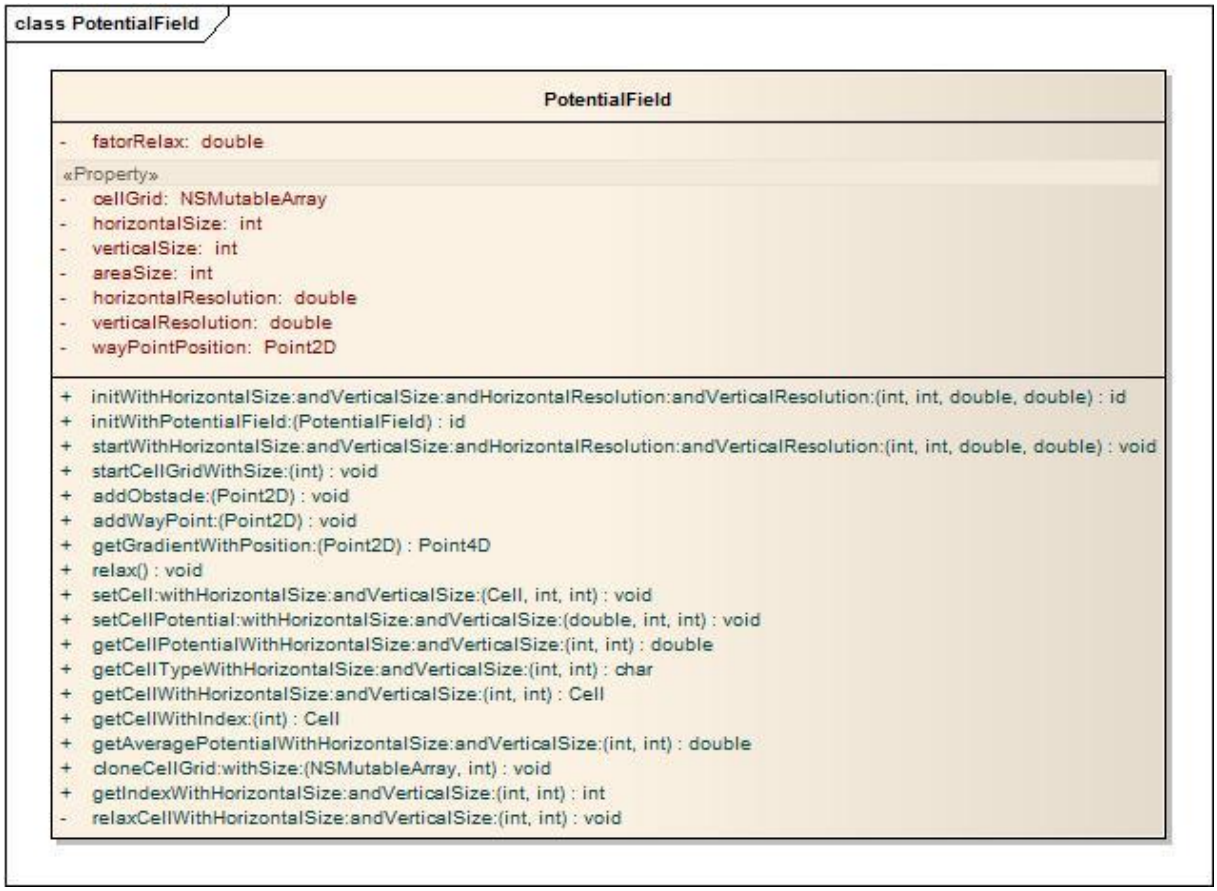


Figura 15 - Diagrama da classe `PotentialField`

3.2.2.5 Classe `PathPlanning`

Como a classe `PathPlanning` é a classe que é instanciada para a utilização da biblioteca, ela contém todos os métodos que são utilizados para alimentar as classes `Room` , `Robot` e `PotentialField`. Disponibiliza o método `movementRobots`, que além de ser utilizado para iniciar a movimentação do agente, verifica se o mesmo já alcançou seus objetivos. Esta classe também é responsável por desenhar os objetos na tela do dispositivo, através do método `drawGridRobot`.

Na Figura 16 é exibido o diagrama da classe `PathPlanning`.

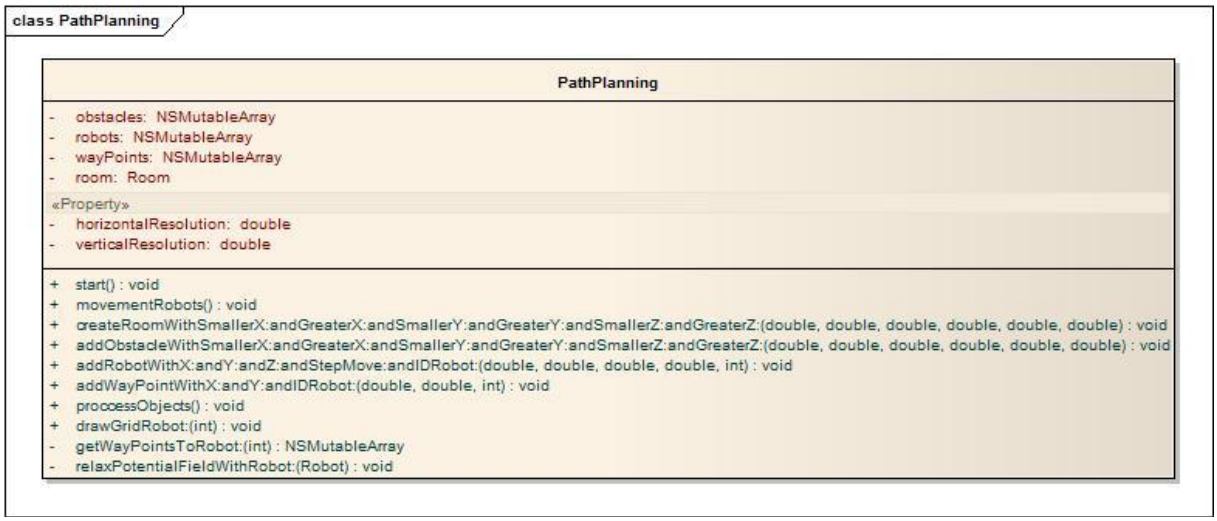


Figura 16 - Diagrama da classe `PathPlanning`

3.2.3 Diagrama de sequência

O diagrama de sequencia da Figura 17 mostra a interação do `Usuário` com a aplicação de planejador de caminhos para os casos de uso `UC01` à `UC08`.

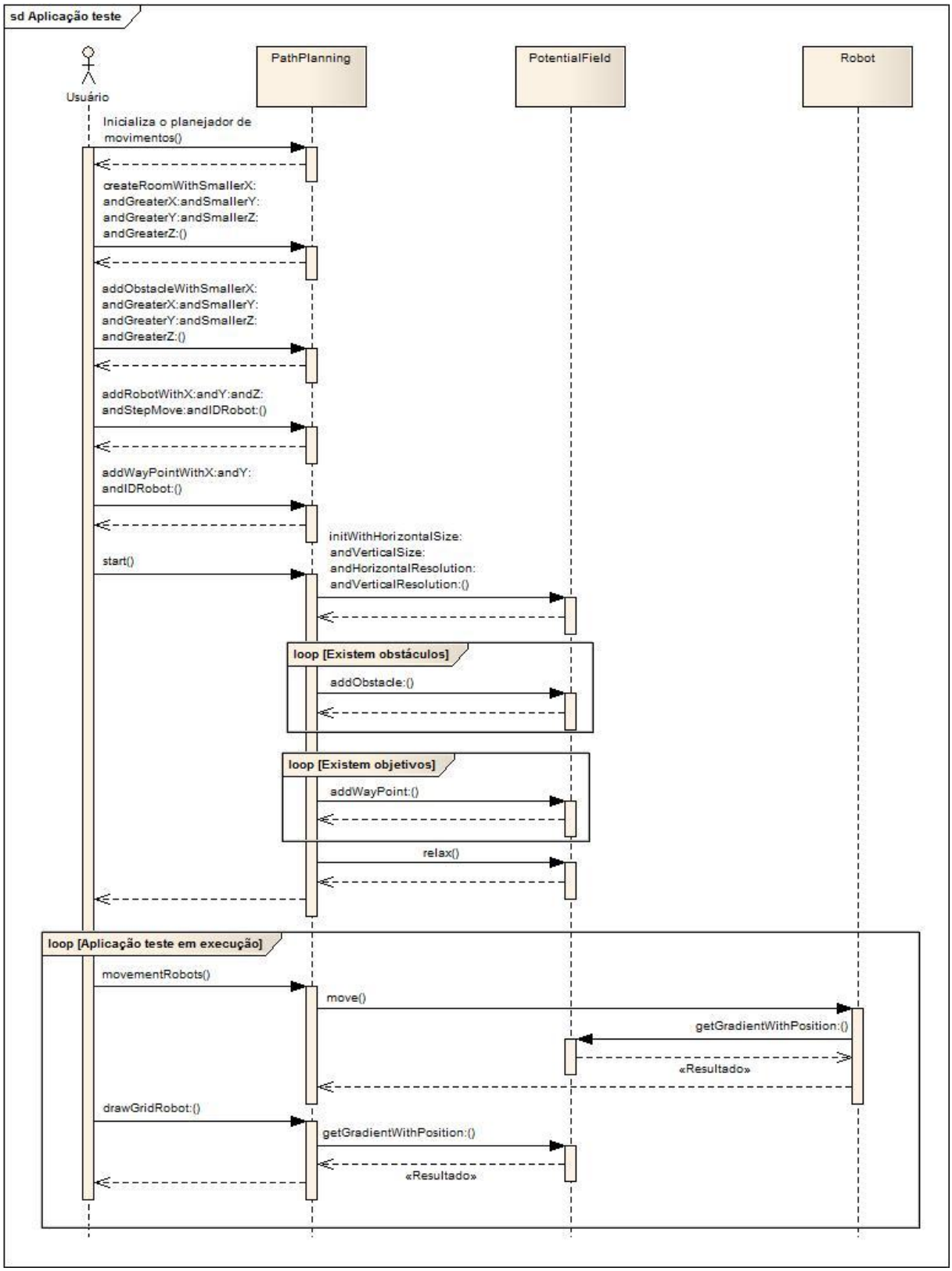


Figura 17 – Diagrama de sequência da aplicação de planejador de caminhos

Inicialmente o `Usuário`, deve inicializar a biblioteca, criando uma instância da mesma. Após este passo, é criado o ambiente em que o campo potencial será aplicado. Em seguida, são informados os obstáculos, o agente e seus objetivos em sequência. Após a chamada do

método `start`, é feito o processamento dos dados de entrada, como obstáculos e objetivos e é executada a rotina de relaxamento do campo potencial para o primeiro objetivo do agente. Por fim é executado o método que realiza a movimentação do agente virtual e a exibição gráfica de todo o ambiente, com obstáculos, agente e objetivo e o próprio campo potencial.

3.2.4 Diagrama de atividades

A Figura 18 detalha as etapas do planejador de movimentos.

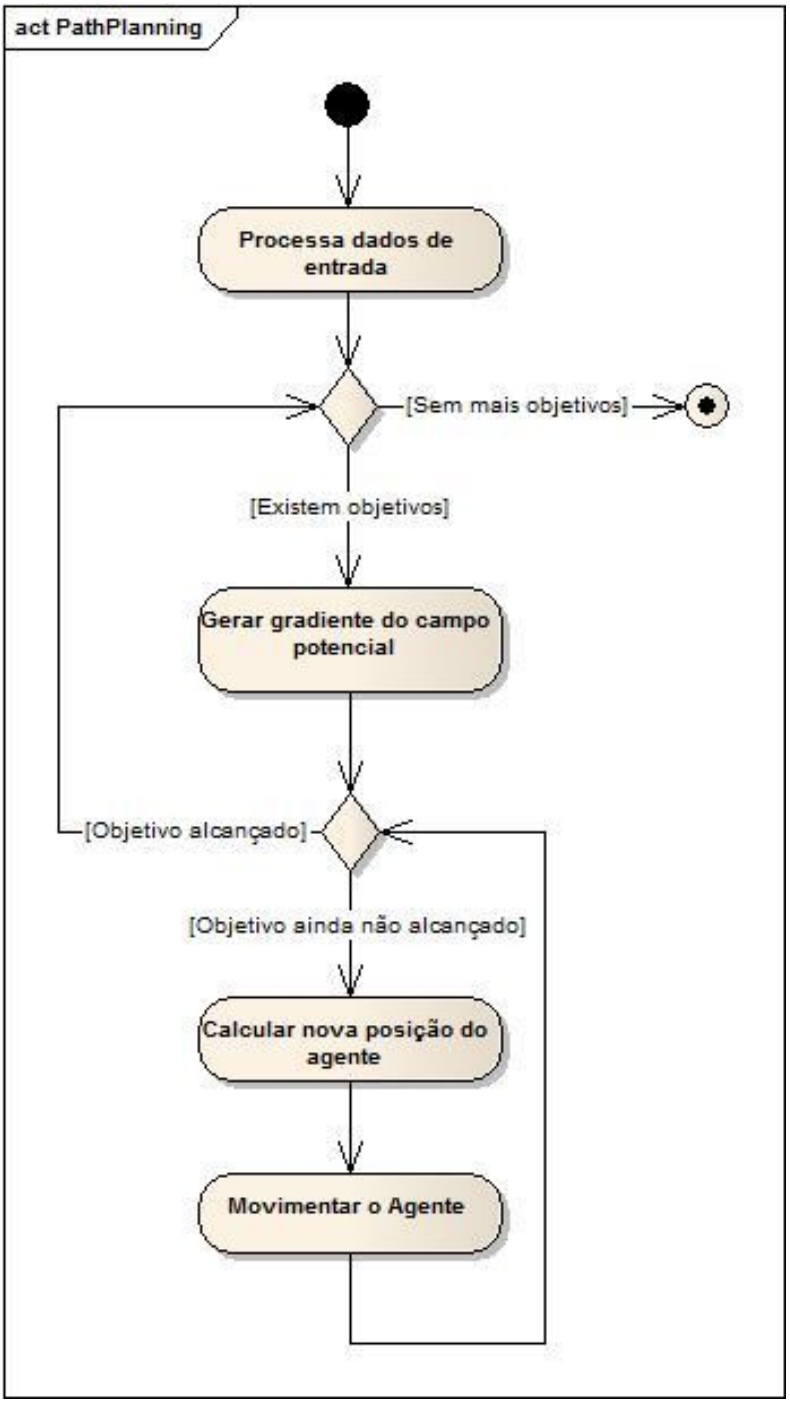


Figura 18 – Diagrama de atividades do planejador de movimentos

A primeira atividade que a biblioteca executa é o processamento dos dados de entrada. Logo em seguida entra em um laço principal, que é executado enquanto existirem objetivos para o agente. Dentro deste laço é gerado o gradiente do campo potencial, ou seja, o relaxamento do campo potencial. Logo em seguida entra em um novo laço, que é executado enquanto o objetivo atual do agente não for alcançado. Dentro deste segundo laço são realizadas as atividades de cálculo da nova posição do agente e a movimentação deste agente. Por fim, se o objetivo foi alcançado e não existir mais nenhum objetivo para o agente é

encerrada a execução da biblioteca.

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas bem como a operacionalidade da biblioteca.

3.3.1 Técnicas e ferramentas utilizadas

A implementação da biblioteca foi realizada utilizando a linguagem de programação Objective-C e a biblioteca gráfica OpenGL ES, dentro do ambiente de desenvolvimento Xcode 4.

Para a criação da documentação da biblioteca foi utilizada a ferramenta Doxygen, utilizada para geração da documentação de código-fonte. Esta ferramenta suporta as linguagens de programação C++, C, Java, Objective-C, Python, IDL, Fortran, VHDL, PHP e C#. A ferramenta tem suporte para gerar a documentação nos formatos *Rich Text Format* (RTF), PostScript, *Portable Document Format* (PDF) com *hyperlinks*, *HyperText Markup Language* (HTML), Latex e *Unix man pages* (HEESCH, 2011).

Segundo Heesch (2011) a documentação pode ser extraída diretamente de código não documentado ou documentado. É possível visualizar os relacionamentos entre vários elementos, incluindo gráficos de dependência, diagramas de classes, diagramas de colaboração, todos gerados automaticamente. A documentação do presente trabalho encontra-se em Ferrari (2011) e a documentação da classe `PotentialField` se encontra no Anexo A.

Já para a execução de testes unitários foi utilizado o *iPhone Simulator*, que é o simulador oficial oferecido pela empresa Apple.

3.3.2 Campo potencial

Para criar o campo potencial o ambiente é dividido em células de mesmo tamanho, formando uma grade, onde inicialmente todas as células são livre e com potencial igual a 0.

Esta divisão é feita através do método estático `getCellPositionWithX:andY:andHorizontalResolution:andVerticalResolution:andUp:` da classe `Mathdef`, que retorna a quantidade de células horizontais e verticais devem ser criadas. O código fonte deste método é apresentado no Quadro 16.

```
+ (Point2D *) getCellPositionWithX: (double) x_ andY: (double) y_
andHorizontalResolution: (double) horizontalResolution_
andVerticalResolution: (double) verticalResolution_ andUp: (BOOL) up_
{
1   int x = (int) (x_ / horizontalResolution_);
2   int y = (int) (y_ / verticalResolution_);
3   if (up_ == YES)
4   {
5     x = (x * horizontalResolution_) < x_ ? (x + 1) : x;
6     y = (y * verticalResolution_) < y_ ? (y + 1) : y;
7   }
8   return [[Point2D alloc] initWithX: x andY: y];
}
```

Quadro 16 – Código fonte do método `getCellPositionWithX:andY:andHorizontalResolution:andVerticalResolution:andUp:`

Após a criação do campo potencial com células livres, devem ser adicionados os obstáculos e os objetivos ao mesmo. Para tanto existem os métodos `addObstacle:` e `addWayPoint:` da classe `PotentialField`, que adicionam respectivamente obstáculos e objetivos ao campo potencial.

Inicialmente apenas o primeiro objetivo do agente é levado em consideração para a geração do campo potencial, caso contrário o campo potencial gerado apontaria para vários locais diferentes. Todas as células que foram marcadas como obstáculos recebem o valor potencial `1.0`. Já as células marcadas como objetivo, recebem o valor potencial `0.0`. O código fonte do método `addObstacle:` pode ser visto no Quadro 17 e o do método `addWayPoint:` no Quadro 18.

```
- (void) addObstacle: (Point2D *) obstacle
{
1   Cell * cell = [[Cell alloc] initWithType: CellObstacle
andPotential: 1];
2   [self setCell: cell withHorizontalSize: obstacle.x andVerticalSize:
obstacle.y];
}
```

Quadro 17 - Código fonte do método `addObstacle:`

```
- (void) addWayPoint: (Point2D *) wayPoint
{
1   Cell * cell = [[Cell alloc] initWithType: CellWayPoint
andPotential: 0];
2   [self setCell: cell withHorizontalSize: wayPoint.x andVerticalSize:
wayPoint.y];
3   wayPointPosition = wayPoint;
}
```

Quadro 18 - Código fonte do método `addWayPoint:`

Por fim, é efetuado o relaxamento do campo potencial através do método `relax` da classe `PotentialField`, cujo código fonte é apresentado no Quadro 19. O relaxamento do campo potencial consiste em encontrar o valor potencial de cada célula da grade, levando em consideração o valor potencial das células vizinhas à cima, à baixo, à direita e à esquerda. O cálculo das células é propagado iniciando-se nos quatro cantos do ambiente e indo para as direções opostas, tanto no sentido vertical, quanto no sentido horizontal. Após todas as propagações estarem finalizadas, todas as células da grade conterão um valor potencial entre `0.0` e `1.0`. Apenas as células marcadas como obstáculos ou como objetivo não sofrerão alterações em seus valores potenciais durante este processo.

```
   - (void) relax
   {
1    for (int v = 0; v < verticalSize; v++)
2    {
3       for (int h = 0; h < horizontalSize; h++)
4       {
5          [self relaxCellWithHorizontalSize: h andVerticalSize: v];
6       }
7    }
                                        ...
25   for (int h = horizontalSize - 1; h > 0; h--)
26   {
27      for (int v = 0; v < verticalSize; v++)
28      {
29         [self relaxCellWithHorizontalSize: h andVerticalSize: v];
30      }
31   }
   }
```

Quadro 19 – Código fonte do método `relax`

Neste processo as células marcadas como obstáculos serão interpretadas como campos potenciais do tipo repulsivos e as marcadas como objetivos serão interpretadas como do tipo atrativos.

3.3.2.1 Problema de valor de contorno

Para resolver o PVC foi implementada a equação de Gauss-Seidel descrita no Quadro 3, através do método `getAveragePotentialWithHorizontalSize:andVerticalSize:` da classe `PotentialField`. Onde ϵ da equação, corresponde à variável `fatorRelax` e o valor utilizado é `0.2`. Para todas as posições em horizontal ou vertical que estejam fora do ambiente, o valor potencial considerado é `1.0`, ou seja, é considerado um obstáculo. O código fonte desse método pode ser visualizado no Quadro 20.

```
- (double) getAveragePotentialWithHorizontalSize: (int)
horizontalSize_ andVerticalSize: (int) verticalSize_
{
                                        ...
   // Busca o valor potencial das células vizinhas
   if (horizontalSize_ > 0)
      left = [self getCellPotentialWithHorizontalSize: (horizontalSize_
- 1) andVerticalSize: verticalSize_];
   else
      left = 1.0;
   if (horizontalSize_ < (horizontalSize - 1))
      right = [self getCellPotentialWithHorizontalSize:
(horizontalSize_ + 1) andVerticalSize: verticalSize_];
   else
      right = 1.0;
                                        ...
   // Calcula o valor potencial de acordo com a equação de Gauss-
Seidel
   double diference = (fatorRelax / 8) * ((top - bottom) + (right -
left));
   return (((top + bottom + right + left) / 4) + diference);
}
```

Quadro 20 – Código fonte do método `getAveragePotentialWithHorizontalSize:andVerticalSize:`

3.3.3 Movimentação do agente

O método `movementRobots` da classe `PathPlanning`, inicializa a movimentação do agente. Enquanto existir um objetivo o agente é movimentado. Se o agente alcançar um objetivo, o método verifica se existem outros objetivos para este agente. Em caso positivo, o novo objetivo é adicionado no campo potencial e removido o antigo. Após é refeito o relaxamento do campo para o novo objetivo. Caso não existe nenhum outro objetivo, o agente para de se mover. O código fonte desse método pode ser visto no Quadro 21.

```
- (void) movementRobots
{
1    for (Robot * robot in room.robots)
2    {
3       if ([robot inWayPoint] == NO)
4       {
5          [robot moveWithRoomArea: [room getBoundingBox]];
                                        ...
18      }
19      else
20      {
21         if (robot.potentialField.wayPointPosition != nil)
22         {
23            Point2D * wayPointPosition = [[Point2D alloc] initWithX:
robot.potentialField.wayPointPosition.x andY:
robot.potentialField.wayPointPosition.y];
24            // Remove o objetivo alcançado da lista de objetivos do
agente
25            [robot.wayPoints removeObject:
robot.potentialField.wayPointPosition];
26            if (robot.wayPoints.count != 0)
27            {
28               // Marca o objetivo como célula vazia
29               [robot.potentialField setCell: [[Cell alloc] initWithType:
CellFree andPotential: 0.0] withHorizontalSize: wayPointPosition.x
andVerticalSize: wayPointPosition.y];
30               // Busca o próximo objetivo
31               [robot.potentialField addWayPoint: [robot.wayPoints
objectAtIndex: 0]];
32               // Relaxa o campo potencial para o  próximo objetivo
33               [self relaxPotentialFieldWithRobot: robot];
34            }
35            else
36               robot.potentialField.wayPointPosition = nil;
37            [wayPointPosition release];
38         }
39      }
40   }
}
```

Quadro 21 – Código fonte do método `movementRobots`

Para movimentar o agente o método `movementRobots` da classe `PathPlanning`, utiliza um método auxiliar chamado `moveWithRoomArea:`, e que se encontra na classe `Robot`. Este método movimenta o agente na horizontal e na vertical, através do gradiente descendente da célula em que ele se encontra. Além de utilizar o gradiente descendente é utilizado mais um parâmetro de controle de velocidade definido pelo `Usuário`. O código fonte do método `move` pode ser visto no Quadro 22.

```
- (void) moveWithRoomArea: (BoundingBox *) roomArea
{
1    if (potentialField.wayPointPosition != nil)
2    {
3       Point4D * gradient = [potentialField getGradientWithPosition:
position];
4       [gradient normalize];
5       [direction addX: gradient.x * stepMove];
6       [direction addY: gradient.y * stepMove];
7       position = [Mathdef getCellPositionWithX: direction.x -
roomArea.smallerPoint.x andY: direction.y - roomArea.smallerPoint.y
andHorizontalResolution: potentialField.horizontalResolution
andVerticalResolution: potentialField.verticalResolution andUp: NO];
8       [gradient release];
9    }
}
```

Quadro 22 – Código fonte do método `moveWithRoomArea:`

No método `getGradientWithPosition:` da classe `PotentialField` é implementada a equação do Quadro 4 e retorna o gradiente descendente de uma célula da grade. O código fonte desse método pode ser observado no Quadro 23.

```
- (Point4D *) getGradientWithPosition: (Point2D *) position_
{
1    Point4D * point = [[Point4D alloc] init];
2    if ([self getCellPotentialWithHorizontalSize: position_.x
andVerticalSize: position_.y] == 0)
3    {
4       point.x = 0;
5       point.y = 0;
6       return point;
7    }
8    double x = [self getCellPotentialWithHorizontalSize: (position_.x -
1) andVerticalSize: position_.y] - [self
getCellPotentialWithHorizontalSize: (position_.x + 1) andVerticalSize:
position_.y];
9    double y = [self getCellPotentialWithHorizontalSize: position_.x
andVerticalSize: (position_.y - 1)] - [self
getCellPotentialWithHorizontalSize: position_.x andVerticalSize:
(position_.y + 1)];
10   double mod = sqrt(x * x + y * y);
11   double fatorMod = 0.00000005;
12   if (fabs(mod) > fabs(fatorMod))
13   {
14      point.x = (x / mod);
15      point.y = (y / mod);
16   }
17   else
18   {
19      point.x = 0;
20      point.y = 0;
21   }
22   return point;
}
```

Quadro 23 – Código fonte do método `getGradientWithPosition:`

3.3.4 Operacionalidade da implementação

Esta seção tem por objetivo mostrar a operacionalidade da implementação em nível de usuário, para isto será exemplificada a utilização da biblioteca de planejamento de caminho. A seguir será apresentada a utilização de cada método disponível para o usuário.

3.3.4.1 Instância e dados de entrada

Nesta seção é mostrado como instanciar a biblioteca de planejamento de caminho e informar os dados de entrada. O Quadro 24 apresenta a implementação do método `initWithCoder:` da classe `EAGLView`, já implementando o planejador de movimentos.

```
   - (id) initWithCoder: (NSCoder *) coder
   {
                                          ...
11    pathPlanning = [[PathPlanning alloc] init];
12    pathPlanning.verticalResolution = 40.0;
13    pathPlanning.horizontalResolution = 40.0;
14
15    [pathPlanning createRoomWithSmallerX: 0.0 andGreaterX: 640.0
      andSmallerY: 0.0 andGreaterY: 960.0 andSmallerZ: 0.0 andGreaterZ:
      0.0];
16
17    [pathPlanning addObstacleWithSmallerX: 200.0 andGreaterX: 400.0
      andSmallerY: 500.0 andGreaterY: 600.0 andSmallerZ: 0.0 andGreaterZ:
      0.0];
18
19    [pathPlanning addRobotWithX: 80.0 andY: 10.0 andZ: 0.0 andStepMove:
      1.5 andIDRobot: 1];
20
21    [pathPlanning addWayPointWithX: 600.0 andY: 800.0 andIDRobot: 1];
22
23    [pathPlanning start];
                                          ...
   }
```

Quadro 24 – Código fonte do método `initWithCoder:`

Na linha 11 do Quadro 24 a classe é `PathPlanning` instanciada. Nas linhas 12 e 13 são definidos os valores de `verticalResolution` e `horizontalResolution` respectivamente. Estes valores definem a quantidade de vezes em que o ambiente será dividido verticalmente e horizontalmente, para formar a grade do campo potencial. Neste caso esta sendo definido `40.0` para ambos. Na linha 15 é criado o ambiente onde será gerado o campo potencial, informando sua posição inicial e final nos eixos x, y e z do ambiente tridimensional. Neste ambiente esta sendo definida sua posição inicial em x, y e z como `0.0` e

sua posição final em `640.0` para x, `960.0` para y e `0.0` para z.

Já na linha 17 esta sendo adicionado um obstáculo ao campo potencial, também é informanda sua posição inicial e final nos eixos x, y e z. Este obstáculo começa em `200.0` para x, `400.0` para y e `0.0` para z e termina em `400.0` para x, `500.0` para y e `0.0` para z. Na linha 19 é inserido o agente que será movimentado, informando sua posição no ambiente tridimensional, o fator de movimentação e um número único ou *identity* (ID) para controle. Ele se encontra em `80.0` no eixo x, `10.0` no eixo y e `0.0` no eixo z. Seu fator de movimentação é `1.5` e seu ID é `1`. Já na linha 21 é adicionado um objetivo para o agente, informando sua posição no ambiente bidimensional e o ID do agente ao qual ele deve ser relacionado. A posição deste objetivo é `600.0` em x e `800.0` em y e o ID do agente é `1`.

Por fim, o conteúdo da linha 23 é um código importante, pois nele é inicializado o planejador de movimentos. Após a execução deste código o campo potencial estará calculado para os dados informados à biblioteca.

3.3.4.2 Visualização do campo potencial e movimentação do agente

Esta seção apresenta a implementação do método `presentFramebuffer` da classe `EAGLView`. Neste método é feita a representação gráfica do campo potencial e da movimentação do agente no ambiente definido. O Quadro 25 apresenta a implementação deste método.

```
- (BOOL) presentFramebuffer
{
                                        ...
3     if (context)
4     {
                                        ...
8        [pathPlanning movementRobots];
9        [pathPlanning drawGridRobot: 1];
                                        ...
12    }
                                        ...
}
```

Quadro 25 – Código fonte do método `presentFramebuffer`

O conteúdo da linha 8 do Quadro 25 é um código importante, pois inicia a movimentação lógica do agente dentro da biblioteca.

Por fim, na linha 9 é chamado o método que faz a representação gráfica do campo potencial do agente. Nesta representação gráfica, são exibidos os obstáculos, como um quadrado cortado por um x, em vermelho, o agente, que é um círculo da cor azul, o objetivo

atual, sendo um quadrado de cor verde e o campo potencial em si. Este último é representado por um conjunto de setas que indicam a direção em que o agente deve se mover a partir daquele ponto. A direção de cada seta é calculada através do gradiente descente daquela célula. Por exemplo, se o ponto for logo à frente de um obstáculo, a seta estará apontando praticamente para o lado oposto ao obstáculo, fazendo com que o agente não colida com o mesmo. Nesta representação gráfica, também está inclusa a representação da movimentação do agente. Na Figura 19 é apresentada a simulação do cenário de exemplo criado.

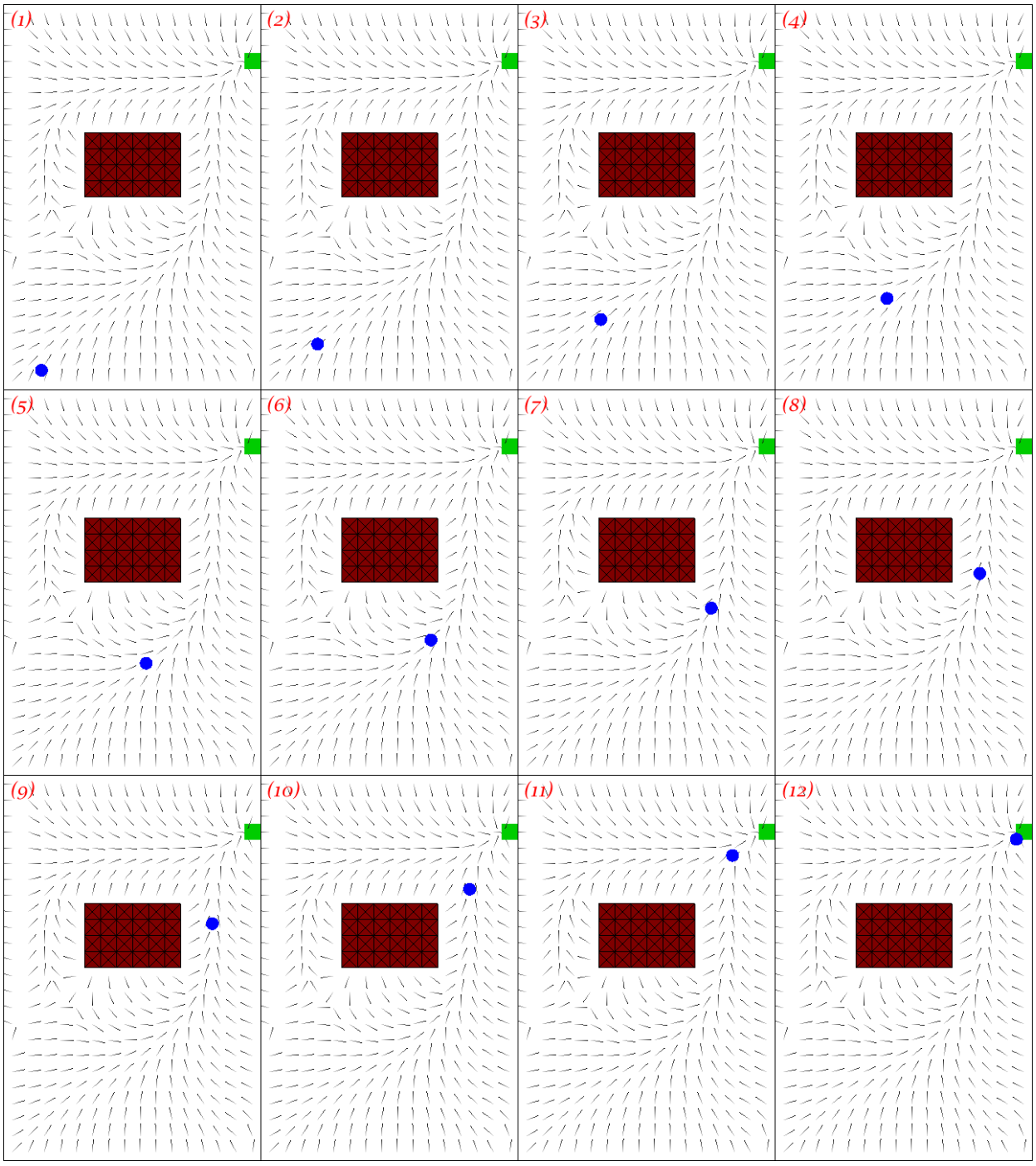


Figura 19 - Animação da simulação exemplo

Na Figura 19 são apresentados 12 frames da simulação do cenário de exemplo, em pontos distintos da execução, desde o início até a chegada ao objetivo. O campo potencial gerado pela configuração do ambiente esta sendo representado pelo conjunto de setas pretas que apontam para o objetivo. Nesta representação o agente percorre o caminho que esta mais destacado, ou seja, onde é possível observar que as setas estão convergindo na direção do objetivo com maior intensidade.

3.4 RESULTADOS E DISCUSSÃO

Os testes da biblioteca foram realizados utilizando dois cenários diferentes. E em cada um destes cenários ainda foram utilizados dois tamanhos diferentes para a grade do campo potencial. Os dois tamanhos utilizados na grade do campo potencial são de 10 x 10 células e 20 x 20 células. Para a execução dos testes foram efetuados utilizando um iMac 9,1 com processador Intel Core 2 Duo com frequência de 2,93 GHz, 4 GBs de memória DDR3 com frequência de 1067 MHz e placa de vídeo NVIDIA GeForce GT 120 com 256 MBs de memória. Também foi utilizado o iPhone Simulator versão 4.3 (238.2).

Os cenários de testes são compostos por aproximadamente 20% de obstáculos. No primeiro teste foi utilizado um cenário com tamanhos da grade potencial 10 x 10 células e 20 x 20 células. Este cenário pode ser visto na da Figura 20.

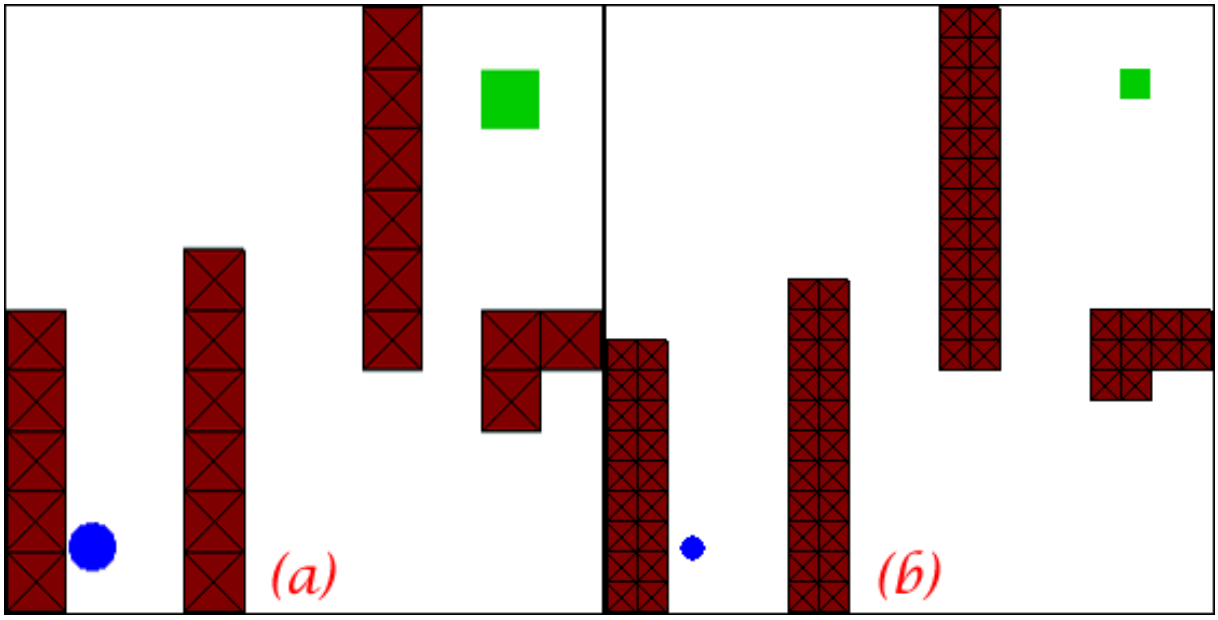


Figura 20 – Cenário de testes 1: (a) tamanho 10 x10 - (b) tamanho 20 x 20

A partir disto foi feita uma verificação de quantas vezes seria necessário executar o

relaxamento das células para obter-se um campo potencial em que o agente conseguisse chegar até seu objetivo. Neste teste foram utilizados os valores 1, 10, 20, 30 e 40 execuções do relaxamento. No Quadro 26 é apresentada uma comparação entre as quantidades de execuções e o resultado obtido com grades de tamanhos 10 x 10 células e 20 x 20 células. O resultado é Sim quando o agente consegue alcançar o objetivo, caso contrário é Não.

Quantidade de execuções do relaxamento	Grade com tamanho 10 x 10 células	Grade com tamanho 20 x 20 células
1	Não	Não
10	Não	Não
20	Sim	Não
30	Sim	Não
40	Sim	Sim

Quadro 26– Quantidade de execuções do relaxamento do cenário 1

No teste com grade de tamanho 10 x 10 células, o campo foi gerado com apenas 20 execuções do relaxamento, enquanto no mesmo cenário com grade de tamanho 20 x 20 células foram necessárias 40 execuções do relaxamento. A partir disto pode-se verificar que para um mesmo cenário, quanto maior for a grade do campo potencial, mais vezes o relaxamento deve ser executado para criar um campo potencial em que o agente alcance o objetivo.

A quantidade de iterações do relaxamento é diretamente relacionada ao tamanho da grade de células, pois os cálculos efetuados para gerar os potenciais, sempre se baseiam em seus vizinhos. Portanto para uma grade maior, nas primeiras iterações do método de relaxamento os vizinhos da célula atual estrão com valores muito aleatórios. Se uma célula vizinha é um obstáculo, o potencial gerado será mais próximo de 1. Enquanto em células que tem todas as células vizinhas forem livres, o potencial gerado será bem próximo do 0. À medida que for aumentando a quantidade de iterações, o valor do potencial vai se aproximando de 1 e se estabilizando, assim formando um campo potencial mais adequado.

Na Figura 21 é possível observar o resultado dos testes do cenário 1, onde o objetivo foi alcançado.

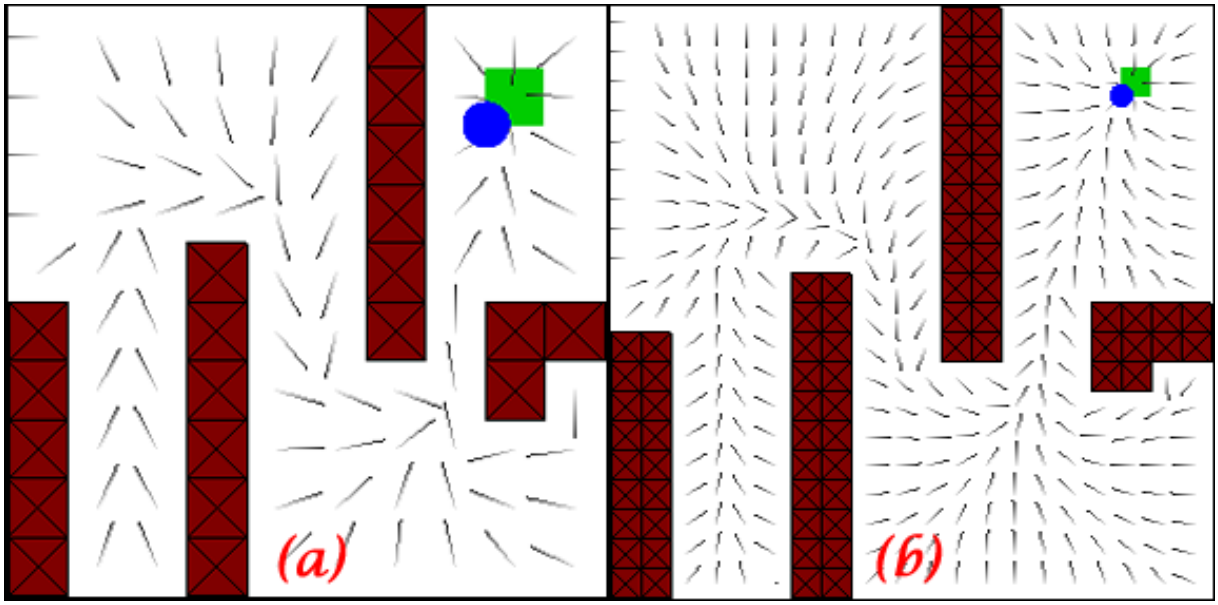


Figura 21 – Resultado dos testes do cenário 1: (a) tamanho 10 x10 - (b) tamanho 20 x 20

No segundo cenário, foi criada uma situação de mínimo local que pode ser observado na Figura 22.

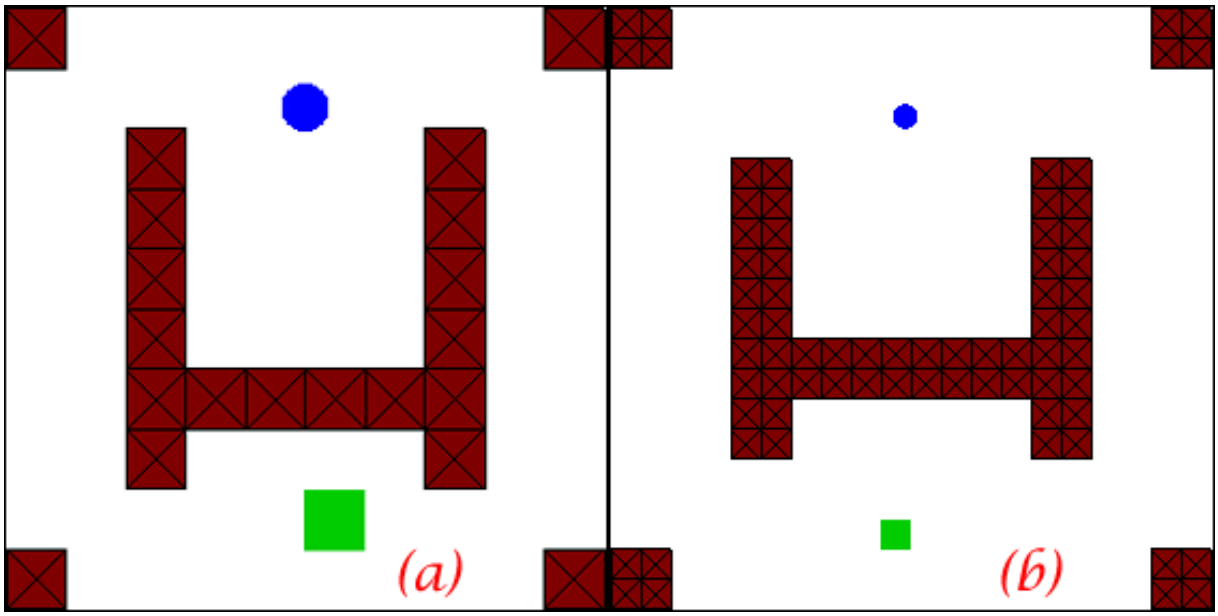


Figura 22 – Cenário de testes 2: (a) tamanho 10 x10 - (b) tamanho 20 x 20

Neste teste foram utilizados os valores 1, 10, 20, 30, 40 e 50 execuções do relaxamento. No Quadro 27 é apresentada uma comparação entre as quantidades de execuções e o resultado obtido com grades de tamanhos 10 x 10 células e 20 x 20 células. O resultado é Sim quando o agente consegue alcançar o objetivo, caso contrário é Não.

Quantidade de execuções do relaxamento	Grade com tamanho 10 x 10 células	Grade com tamanho 20 x 20 células
1	Não	Não
10	Não	Não
20	Sim	Não
30	Sim	Não
40	Sim	Não
50	Sim	Sim

Quadro 27 – Quantidade de execuções do relaxamento do cenário 2

No teste com grade de tamanho 10 x 10 células, o campo foi gerado com 20 execuções do relaxamento, a mesma quantidade de execuções utilizada no cenário 1 com o mesmo tamanho. Já o mesmo cenário com grade de tamanho 20 x 20 células foram necessárias 50 execuções do relaxamento, 10 a mais que o cenário 1. A partir disto pode-se verificar que quanto mais complexo for o cenário, mais vezes o relaxamento deve ser executado para criar um campo potencial em que o agente alcance o objetivo. Na Figura 23 é possível observar o resultado dos testes do cenário 2, onde o objetivo foi alcançado.

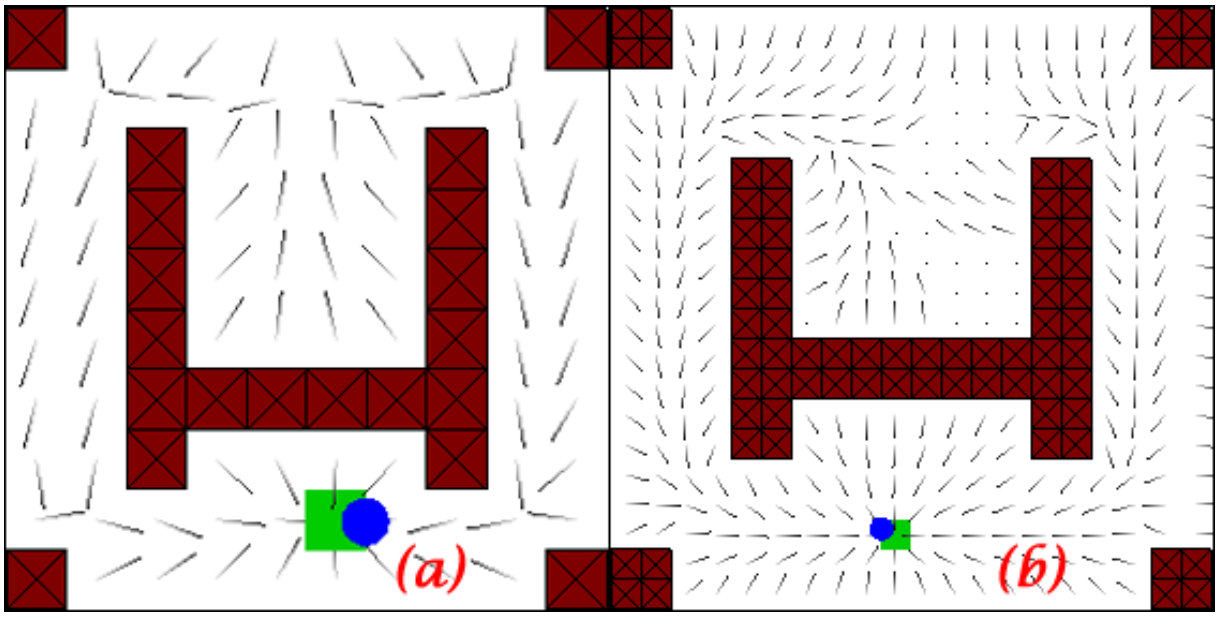


Figura 23 – Resultado dos testes do cenário 2: (a) tamanho 10 x10 - (b) tamanho 20 x 20

Em ambos os cenários de teste, mesmo necessitando de certo processamento, foi possível alcançar o objetivo do agente. Também foi possível verificar que a biblioteca consegue evitar mínimos locais, que era um dos requisitos funcionais.

Além dos dois cenários citados, foram feitos testes em mais 10 cenários diferentes e em todos os cenários, o planejador de movimentos conseguiu fazer com que o agente chegasse ao objetivo com 40 execuções ou menos do relaxamento. Além disto, 5 destes

cenários possuíam vários objetivos para o agente e todos foram alcançados. Outro ponto positivo é que não houve colisões com obstáculos em nenhum dos cenários testados. Os 10 cenários alternativos em que a biblioteca foi testada, podem ser observados no Apêndice A.

Uma dificuldade encontrada foi em relação à movimentação do agente, que está sendo baseada somente do gradiente descendente do campo potencial (Quadro 4) e no fator de movimentação do agente. Segundo Silveira (2010, p. 34) ao utilizar somente o gradiente descendente para movimentar o agente, faz com que o seu comportamento não seja tão parecido com o mundo real. Dapper, Prestes e Nedel (2007, p. 4) e Silveira (2010, p. 35) propuseram uma equação que gera movimentos mais reais (Quadro 6), mas ao tentar adaptar esta equação a linguagem de programação, a mesma não se comportou como o esperado e o agente não se movia obedecendo ao campo potencial, o que também causava várias colisões. Por estes motivos optou-se em utilizar a movimentação baseada apenas no gradiente descendente.

Levando em consideração que a equação do Quadro 6 não foi implementada, a biblioteca também não consegue planejar movimentos para mais de um agente simultaneamente. Para a biblioteca conseguir planejar os movimentos de vários agentes ao mesmo tempo, sem que ocorram colisões entre os mesmos, faz-se necessária a implementação desta equação. Por este motivo não foi possível desenvolver plenamente o requisito funcional relacionado à quantidade de pontos de partida, estando limitado a apenas um por execução.

Por fim, no Quadro 28 é apresentado um comparativo entre os trabalhos correlatos e o presente trabalho.

	FISCHER 2008	**FERRARI 2009**	**MAFRA 2004**	**Biblioteca de Planejamento de Movimento**
Suporte a mais de 1 agente simultaneamente	Sim	Não	Sim	Não
Suporte a mais de 1 objetivo sequencialmente	Sim	Não	Não	Sim
Movimentação próxima ao mundo real	Sim	Não	Não	Não
Evita mínimos locais	Sim	Sim	Não se aplica	Sim
Suporte a plataforma móvel iOS 4	Não	Não	Não	Sim

Quadro 28 – Comparativo dos trabalhos correlatos

Os trabalho de Ferrari (2009) e Mafra (2004) não possuem suporte a mais de 1 objetivo

sequencialmente, enquanto a biblioteca do presente trabalho possui. Em relação ao trabalho de Fischer (2008) o presente trabalho não possui suporte a mais de um agente e a movimentação do agente é menos semelhante ao mundo real. A grande vantagem do presente trabalho em relação aos trabalhos correlatos, está em a biblioteca ser compatível com a plataforma de dispositivos móveis iOS 4.

4 CONCLUSÕES

Este trabalho apresentou uma biblioteca para planejamento de movimentos de agentes virtuais, baseada em campos potenciais. Estes campos potenciais são livres do problema de mínimos locais, pois são gerados a partir de soluções numéricas para PVCs e geram trajetórias suaves se comparados a outros modelos de movimentação. Com estes argumentos, o presente trabalho atingiu os seus objetivos. Já em relação aos requisitos propostos não completou apenas um dos requisitos. Este requisito diz respeito à quantidade de agentes simultâneos que a biblioteca desenvolvida consegue suportar ao mesmo tempo, que se limita a apenas um. Em contra partida, os demais requisitos foram atendidos plenamente.

Em relação aos trabalhos correlatos, as principais vantagens são o suporte a plataforma para dispositivos móveis iOS 4 e a possibilidade de informar vários objetivos em sequência para o agente.

As principais limitações da biblioteca dizem respeito à quantidade de agentes que podem ser simulados simultaneamente e a movimentação do agente. A biblioteca só é capaz de planejar movimentos para um agente de cada vez. E por sua vez a movimentação deste agente não gera comportamentos realísticos, como os observados no mundo real.

4.1 EXTENSÕES

Como sugestões para possíveis extensões à biblioteca desenvolvida citam-se:

a) implementar a equação de movimentação descrita no Quadro 6, que segundo Silveira (2010, p. 35-36), torna a movimentação do agente mais próxima do mundo real;
b) permitir mais de um agente simultaneamente na biblioteca. Para tanto é necessário levar em consideração que qualquer agente passa a ser um obstáculo dinâmico para os outros agentes;
c) permitir a criação de controles de grupos de agentes, permitindo definir formações. Para maiores informações sobre o assunto, consulte Silveira (2010, p. 25-60);
d) divisão do campo potencial em mapa global, onde estão identificados todos os obstáculos estáticos, e mapa local com janela de visualização de tamanho menor,

onde são identificados os obstáculos estáticos e dinâmicos, apenas dentro da janela de visualização;

e) criação de portais entre salas para que o campo potencial calculado seja apenas do ambiente em que o agente se encontra, fazendo com que cada portal acabe sendo um objetivo intermediário.

REFERÊNCIAS BIBLIOGRÁFICAS

APPLE INC. **iOS overview**. [S.l.], 2010a. Disponível em: <http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/URL_iPhone_OS_Overview/index.html>. Acesso em: 14 set. 2010.

_____. **UIKit framework reference**. [S.l.], 2010b. Disponível em: <http://developer.apple.com/library/ios/#documentation/UIKit/Reference/UIKit_Framework/_index.html>. Acesso em: 15 set. 2010.

ARIKAN, Okan; CHENNEY, Stephen; FORSYTH, David. A. Efficient multi-agent path planning. In: EUROGRAPHICS WORKSHOP ON ANIMATION AND SIMULATION, 3rd, 2001, Manchester. **Proceedings...** [Springer-Verlag]: [New York], 2001. Não paginado. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.67.6406&rep=rep1&type=pdf>. Acesso em: 4 set. 2010.

CONNOLLY, Christopher I.; BURNS, Brian J.; WEISS, Richard. Path planning using Laplace's equation. In: IEEE INTERNATION CONFERENCE ON ROBOTICS AND AUTOMATION, 7th, 1990, Cincinnati. **Proceedings…** Cincinnati: IEEE Computer Society Press, 1990. p. 2102-2106. Disponível em: <http://vkuliah.files.wordpress.com/2010/04/1990-connolly.pdf>. Acesso em: 4 set. 2010.

DAPPER, Fábio. **Planejamento de movimento para pedestres utilizando campos potenciais**. 2007. 73 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/12219/000622643.pdf?sequence=1>. Acesso em: 19 ago. 2010.

DAPPER, Fábio; PRESTES, Edson; NEDEL, Luciana P. **Generating steering behaviors for virtual humanoids using BVP control**. Porto Alegre, [2007?]. Disponível em: <http://www.inf.ufrgs.br/cgi2007/cd_cgi/papers/dapper.pdf>. Acesso em: 12 set. 2010.

FERRARI, Fabricio. A new parameterized potential family for path planning algorithms. **International Journal on Artificial Intelligence Tools**, New Jersey, v. 18, n. 6, p. 949–957, Apr. 2009. Disponível em: <http://www.ferrari.pro.br/home/publications/papers/FFerrari_IJAIT2009_00047.pdf>. Acesso em: 16 set. 2010.

FERRARI, Marcelo R. [S.l.], 2011. Disponível em: <http://www.inf.furb.br/gcg/files/TCC2011-1-28-MarceloRobertoFerrari_documentacao.zip>. Acesso em: 14 maio 2011

FISCHER, Leonardo G. **Otimização de desempenho em planejadores de caminho usando campos potenciais**. 2008. 63 f. Trabalho de Graduação (Bacharelado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/16009/000681150.pdf?sequence=1>. Acesso em: 19 ago. 2010.

FISCHER, Leonardo G.; SILVEIRA, Renato; NEDEL, Luciana. GPU accelerated path-planning for multi-agents in virtual environments. In: SIMPÓSIO BRASILEIRO DE JOGOS E ENTRETENIMENTO DIGITAL, 8., 2009, Rio de Janeiro. **Anais…** Rio de Janeiro: SBC, 2009. p. 112-118. Disponível em: <http://www.sbgames.org/papers/sbgames09/computing/full/cp15_09.pdf>. Acesso em: 7 set. 2010.

GOODRICH, Michael A. **Potential fields tutorial**. [S.l.], [2007?]. Disponível em: <http://borg.cc.gatech.edu/ipr/files/goodrich_potential_fields.pdf>. Acesso em: 3 set. 2010.

HEESCH, Dimitri. **Doxygen**. [S.l.], 2011. Disponível em: <http://www.stack.nl/~dimitri/doxygen>. Acesso em: 4 maio 2011.

KHATIB, Oussama. **Commande dynamique dans l'espace operationnel des robots manipulateurs en presence d'obstacles**. 1980. 282 f. Thèse (Docteur Ingenieur) – L'École Nationale Supérieure de L'Aéronautique et de L'Espace, Toulouse, France. Disponível em: <http://cs.stanford.edu/groups/manips/images/pdfs/Khatib_1980_thesis.pdf>. Acesso em: 28 ago. 2010.

KHRONOS GROUP. **OpenGL ES overview**. [S.l.], 2010. Disponível em: <http://www.khronos.org/opengles>. Acesso em: 15 set. 2010.

LATOMBE, Jean-Claude. **Motion planning**: a journey of robots, molecules, digital actors, and other artifacts. Stanford, 1999. Disponível em: <http://gamma.cs.unc.edu/courses/planning-f07/PAPERS/Motionp-Planning-Overview.pdf>. Acesso em: 7 set. 2010.

MAFRA, Julio C. **Protótipo de formação em times de futebol de robôs utilizando robótica baseada em comportamento**. 2004. 60 f. Trabalho de Graduação (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. Disponível em: <http://www.bc.furb.br/docs/MO/2004/306333_1_1.pdf>. Acesso em: 4 mar. 2011.

NIEUWENHUISEN, Dennis; KAMPHUIS, Arno; OVERMARS, Mark H. **High quality navigation in computer games**. Netherlands, 2006. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.106.933&rep=rep1&type=pdf>. Acesso em: 9 set. 2010.

OPENGL. In: WIKIPÉDIA, a enciclopédia livre. [S.l.]: Wikipedia Foundation, 2010. Disponível em: <http://pt.wikipedia.org/wiki/OpenGL>. Acesso em: 15 set. 2010.

PAN, Jia; LAUTERBACH, Christian; MANOCHA, Dinesh. g-Planner: real-time motion planning and global navigation using GPUs. In: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, 24th, 2010, Atlanta. **Technical Report...** [S.l.]: AAAI Press, 2010. Não paginado. Disponível em: <http://gamma.cs.unc.edu/gplanner/AAAI.pdf>. Acesso em: 27 ago. 2010.

RUSSELL, Stuart J.; NORVIG, Peter. **Artificial intelligence**: a modern approach. 2nd ed. New Jersey: Prentice-Hall, 1995. Disponível em: <http://flmsdown.net/ebooks/360263-artificial-intelligence-a-modern-approach.html>. Acesso em: 11 set. 2010.

SILVEIRA, Renato. **Planejamento de movimento para grupos utilizando campos potenciais**. 2010. 78 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre. Disponível em: <http://www.inf.ufrgs.br/~rsilveira/dissertacao>. Acesso em: 19 ago. 2010.

TREVISAN, Marcelo et al. Exploratory navigation based on dynamic boundary value problems. **Journal of Intelligent and Robotic Systems**, [S.l.], v. 45, n. 2, p. 101–114, Feb. 2006. Disponível em: <http://www.inf.ufrgs.br/~prestes/publicacoes/JINT1909.pdf>. Acesso em: 21 ago. 2010.

WEISSTEIN, Eric W. **Boundary value problem**. [S.l.], [2010?]. Disponível em: <http://mathworld.wolfram.com/BoundaryValueProblem.html>. Acesso em: 12 set. 2010.

APÊNDICE A - Cenários utilizados para execução dos testes unitários

Na Figura 24 são apresentados os cenários de testes utilizados para verificar o funcionamento da biblioteca. Os cenários estão enumerados de 1 a 10 e estão na ordem em que foram testados. Para os cenários de 6 a 10 é exibido apenas o último objetivo do agente e seu respectivo campo potencial, já que os mesmos possuem vários objetivos. O agente é exibido em seu estado inicial. Também é exibido o caminho que o agente efetuou até chegar ao objetivo.

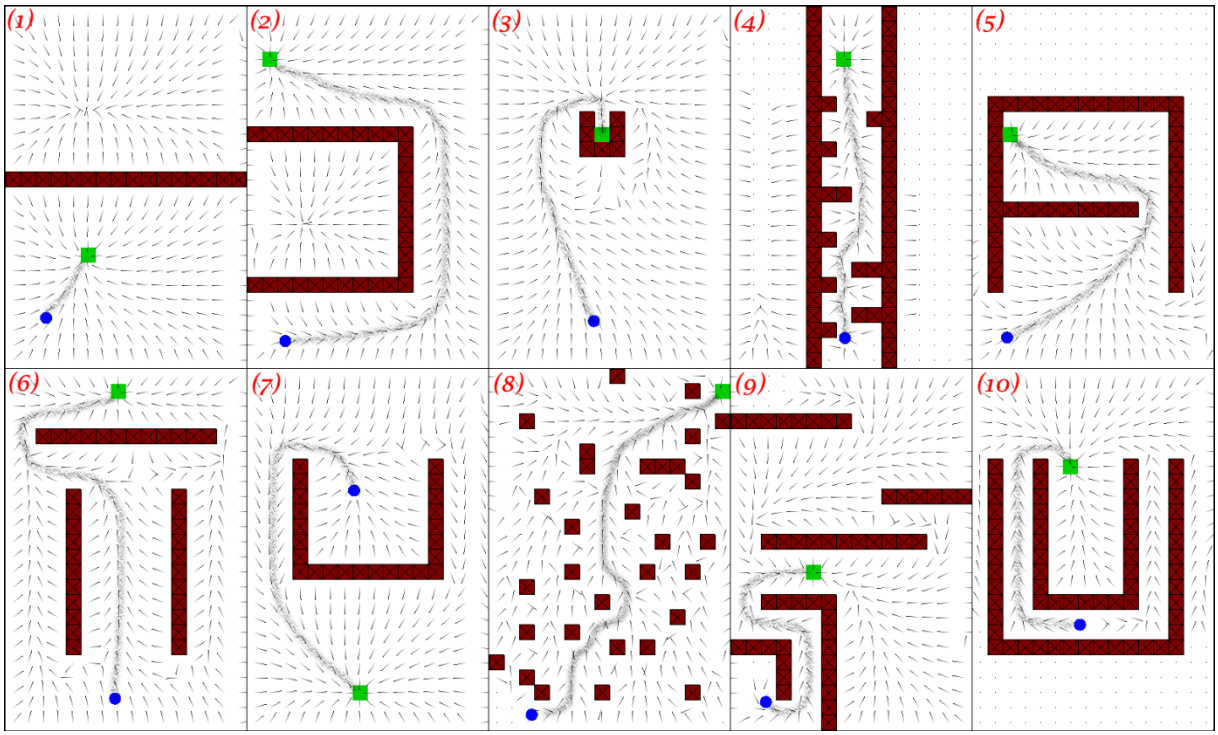


Figura 24 – Cenários de testes utilizados

ANEXO A – Documentação parcial da biblioteca para explorar algoritmos de planejamento de movimento utilizando campos potenciais no SDK do iOS 4

No Quadro 29 é apresentada a documentação parcial desta biblioteca gerada a partir da documentação no código fonte. Neste anexo consta somente a documentação da classe `PotentialField` devido a esta documentação ser extensa. Foi mantida a formatação semelhante à da documentação completa.

PotentialField Class Reference

#import <PotentialField.h>

Public Member Functions

(id) **- initWithHorizontalSize:andVerticalSize:andHorizontalResolution:andVerticalResolution:**
(id) **- initWithPotentialField:**
(void) **- startWithHorizontalSize:andVerticalSize:andHorizontalResolution:andVerticalResolution:**
(void) **- startCellGridWithSize:**
(void) **- addObstacle:**
(void) **- addWayPoint:**
(Point4D *) **- getGradientWithPosition:**
(void) **- relax**
(void) **- setCell:withHorizontalSize:andVerticalSize:**
(void) **- setCellPotential:withHorizontalSize:andVerticalSize:**
(double) **- getCellPotentialWithHorizontalSize:andVerticalSize:**
(char) **- getCellTypeWithHorizontalSize:andVerticalSize:**
(Cell *) **- getCellWithHorizontalSize:andVerticalSize:**
(Cell *) **- getCellWithIndex:**
(double) **- getAveragePotentialWithHorizontalSize:andVerticalSize:**
(void) **- cloneCellGrid:withSize:**
(int) **- getIndexWithHorizontalSize:andVerticalSize:**
(void) **- relaxCellWithHorizontalSize:andVerticalSize:**

Protected Attributes

double **fatorRelax**

Properties

NSMutableArray * **cellGrid**
int **horizontalSize**
int **verticalSize**
int **areaSize**
double **horizontalResolution**
double **verticalResolution**
Point2D * **wayPointPosition**

Detailed Description

Classe de controle do campo potencial

Member Function Documentation

- (void) addObstacle: (Point2D *) obstacle
Adiciona um novo obstáculos ao campo potencial

Parameters:

Obstacle Posição dentro da grade (x,y)

- (void) addWayPoint: (Point2D *) wayPoint
Adiciona um novo objetivo ao campo potencial

Parameters:

wayPoint Posição dentro da grade (x,y)

- (void) cloneCellGrid: (NSMutableArray *) cellGrid_ withSize: (int) size
Copia uma grade potencial existente

Parameters:

cellGrid_ Grade original a ser copiada
size Tamanho da grade

- (double) getAveragePotentialWithHorizontalSize: (int) horizontalSize_ andVerticalSize: (int) verticalSize_
Gera o potencial para determinada célula da grade
Fórmula utilizada: $p_c = \frac{p_t + p_b + p_l + p_r}{4} + \frac{\epsilon}{8}\left((p_r - p_l)v_x + (p_b - p_t)v_y\right)$
onde $p_c = p_{i,j}^{t+1}$, $p_t = p_{i,j-1}^{t+1}$, $p_b = p_{i,j+1}^{t}$, $p_l = p_{i-1,j}^{t+1}$, $p_r = p_{i+1,j}^{t}$ $v = (v_x, v_y)$, sendo v um fator de ajuste e $\epsilon \in (-\mathbf{2}, \mathbf{2})$.

Parameters:

horizontalSize_ Posição horizontal
verticalSize_ Posição vertical

- (double) getCellPotentialWithHorizontalSize: (int) horizontalSize_ andVerticalSize: (int) verticalSize_
Busca o potencial da célula pela sua posição na grade

Parameters:

horizontalSize_ Posição horizontal
verticalSize_ Posição vertical

Returns:

Potencial da Célula

- (char) getCellTypeWithHorizontalSize: (int) horizontalSize_ andVerticalSize: (int) verticalSize_
Busca o tipo de célula pela sua posição na grade

Parameters:

horizontalSize_ Posição horizontal
verticalSize_ Posição vertical

Returns:

Tipo da Célula

- (Cell *) getCellWithHorizontalSize: (int) horizontalSize_ andVerticalSize: (int) verticalSize_
Busca a célula pela sua posição na grade

Parameters:

horizontalSize_ Posição horizontal
verticalSize_ Posição vertical

Returns:
Célula

- (Cell *) getCellWithIndex: (int) index
Busca a célula pelo seu índice

Parameters:
Index índice da célula

Returns:
Célula

- (Point4D *) getGradientWithPosition: (Point2D *) position_
Gera o gradiente de uma célula

Parameters:
position_ Posição da célula dentro da grade (x,y)

Returns:
Gradiente descendente da célula

- (int) getIndexWithHorizontalSize: (int) horizontalSize_ andVerticalSize: (int) verticalSize_
Retorna o índice de uma célula dentro da coleção

Parameters:
horizontalSize_ Posição horizontal
verticalSize_ Posição vertical

- (id) initWithHorizontalSize: (int) horizontalSize_ andVerticalSize: (int) verticalSize_ andHorizontalResolution: (double) horizontalResolution_ andVerticalResolution: (double) verticalResolution_
Inicializa com parâmetros

Parameters:
horizontalSize_ Posição horizontal
verticalSize_ Posição vertical
horizontalResolution_ Resolução horizontal
verticalResolution_ Resolução vertical

- (id) initWithPotentialField: (PotentialField *) potentialField_
Inicializa com um campo potencial

Parameters:
potentialField_ Campo potencial já gerado

See also:
- startWithHorizontalSize:andVerticalSize:andHorizontalResolution:andVerticalResolution:
- cloneCellGrid:withSize:

- (void) relax
Gera o campo potencial para toda a grade

- (void) relaxCellWithHorizontalSize: (int) horizontalSize_ andVerticalSize: (int) verticalSize_
Método auxiliar ao relax

Parameters:
horizontalSize_ Posição horizontal
verticalSize_ Posição vertical

- (void) setCell: (Cell *) cell_ withHorizontalSize: (int) horizontalSize_ andVerticalSize: (int) verticalSize_
Altera uma célula
Parameters:
cell_ Nova Célula
horizontalSize_ Posição horizontal
verticalSize_ Posição vertical

- (void) setCellPotential: (double) potential_ withHorizontalSize: (int) horizontalSize_ andVerticalSize: (int) verticalSize_
Altera o potencial de uma célula
Parameters:
potential_ Valor potencial
horizontalSize_ Posição horizontal
verticalSize_ Posição vertical

- (void) startCellGridWithSize: (int) size
Inicializa um campo potencial vazio
Parameters:
Size Quantidade de células da grade do campo potencial

- (void) startWithHorizontalSize: (int) horizontalSize_ andVerticalSize: (int) verticalSize_ andHorizontalResolution: (double) horizontalResolution_ andVerticalResolution: (double) verticalResolution_
Inicializa o campo potencial
Parameters:
horizontalSize_ Posição horizontal
verticalSize_ Posição vertical
horizontalResolution_ Resolução horizontal
verticalResolution_ Resolução vertical

Member Data Documentation

- (double) fatorRelax `[protected]`
Fator utilizado no relaxamento das células

Property Documentation

- (int) areaSize `[read, write, assign]`
Tamanho total da garde

- (NSMutableArray *) cellGrid `[read, write, retain]`
Coleção das células da grade

- (double) horizontalResolution `[read, write, assign]`
Resolução horizontal

- (int) horizontalSize `[read, write, assign]`
Tamanho horizontal da garde

- (double) verticalResolution `[read, write, assign]`
Resolução vertical

- (int) verticalSize `[read, write, assign]`
Tamanho vertical da garde

- (Point2D *) wayPointPosition `[read, write, retain]`
Posição do objetivo

The documentation for this class was generated from the following files:

- Projeto/TCC2/**PotentialField.h**
- Projeto/TCC2/**PotentialField.m**

Quadro 29 – Documentação parcial da biblioteca