Universidade Federal do Piauí
Campus Senador Helvídio Nunes de Barros
Bacharelado em Sistemas de Informação

André Lima

Inteligência de Enxame aplicada à Robótica

Picos
2013

André Lima

Inteligência de Enxame aplicada à Robótica

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação do Campus Senador Helvídio Nunes de Barros da Universidade Federal do Piauí como parte para obtenção do Grau de Bacharel em Sistemas de Informação.

Área de concentração: Inteligência Artificial.
Orientação: Prof. MSc. Algeir Prazeres Sampaio

Picos
2013

Eu, **André Lima**, abaixo identificado como autor, autorizo a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação abaixo discriminada, de minha autoria, em seu site, em formato PDF, para fins de leitura e ou impressão, a partir da data de hoje.

Picos-PI, 18 de abril de 2013.

FICHA CATALOGRÁFICA
Serviço de Processamento Técnico da Universidade Federal do Piauí
Biblioteca José Albano de Macêdo

L732i Lima, André.
Inteligência de Enxame aplicada à Robótica / André Lima. – 2013.
CD-ROM: il.; 4 ¾ pol. (79 p.)

Monografia (Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí. Picos-PI, 2013.
Orientador: Prof. MSc. Algeir Prazeres Sampaio

1. Formigas. 2. Robótica. 3. *Ant Colony Optimization*. I. Título.

CDD 629.892

André Lima

Inteligência de Enxame aplicada à Robótica

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação do Campus Senador Helvídio Nunes de Barros da Universidade Federal do Piauí como parte para obtenção do Grau de Bacharel em Sistemas de Informação.

Área de concentração: Inteligência Artificial.
Orientação: Prof. MSc. Algeir Prazeres Sampaio

Data de Aprovação:

12 de Abril de 2013

Prof. MSc. Algeir Prazeres Sampaio_____ UFPI

Prof. MSc. Patrícia Medyna Lauritzen de Lucena Drumond_____ UFPI

Prof. MSc. Frank César Lopes Véras_____ UFPI

Picos

2013

“Se não fosse pelo último minuto, muita coisa ficaria sem fazer”

Provérbio Chinês

Aos meus pais, José da Paz de Moura Lima e Cleide Maria Conrado Santos Lima, que me deram força e suporte para atingir essa conquista. A minha irmã Amanda, sempre me ajudando. Aos familiares, em especial aos tios, Maria do Carmo e Pedro que forneceram o conforto do lar e total apoio nos estudos. A prima Shyara pela ajuda e apoio ao longo do curso. Aos colegas e amigos que ajudaram de forma direta e indiretamente. A todos os professores do curso de Sistemas de informação, em especial a Algeir Sampaio que me orientou e me apresentou ao tema desse trabalho. A Patrícia Medyna pela ajuda ao longo desse trabalho.

“Se A é o sucesso, então A é igual a X mais Y mais Z. O trabalho é X; Y é o lazer; e Z é manter a boca fechada.”

Albert Einstein

Resumo

A Ciência da Computação pode se valer de exemplos encontrados na natureza para a resolução de problemas, principalmente problemas relacionados à otimização. As técnicas utilizadas para o projeto de algoritmos computacionais baseados em princípios biológicos são conhecidas por Computação Biológica ou Biocomputação. Dentro desse contesto, baseadas na capacidade de algumas espécies de formigas de encontrar o caminho mais curto entre seu ninho e uma fonte de alimento, deram origem a uma técnica heurística de otimização denominada *Ant Colony Optimization*. Essa técnica tem sido amplamente utilizada na robótica de enxame e vem trazendo resultados muito promissores. Este é um trabalho de foco prático. Nesse sentido, seu objetivo principal é implantar técnicas de controle de enxames em robôs com ênfase no algoritmo das formigas.

Palavras-chave: Formigas, Robótica, Ant Colony Optimization.

Abstract

The Computer science can draw from examples found in nature to solve problems, especially problems related to optimization. The techniques used for the design of computational algorithms based on biological principles are known or Biocomputation Biological Computation. Within this contest, based on the ability of some species of ants to find the shortest path between their nest and a source food, resulted in a heuristic optimization technique named Ant Colony Optimization. This technique has been widely used in swarm robotics has brought very promising results. This is a work of practical focus. In this sense, your main goal is to deploy technical control swarms of robots with emphasis on ant algorithm.

Keywords: Ants, Robotics, Ant Colony Optimization.

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas e siglas

ACO	*Ant Colony Optimization*
BGE	*Blender Game Engine*
IA	Inteligência Artificil
IE	Inteligência de Enxame
LDR	*Light Dependent Resistor*

Sumário

1 Capitulo

1.1 Introdução

Ao falar em inteligência artificial (IA), logo vem à mente máquinas com capacidade intelectual igual ou até superior aos humanos, esse era o primeiro objetivo da IA. Com os anos de pesquisa, percebeu-se a dificuldade e complexidade em representar em sua totalidade a inteligência humana. Atualmente, a IA abrange uma enorme variedade de subcampos, desde áreas de uso geral como aprendizado e percepção (RUSSELL; NORVING, 2004), até tarefas específicas como Planejamento, Visão Computacional, Robótica, Sistemas Especialistas, Processamento de Linguagem Natural, Jogos, Mineração de Dados, Mercado Financeiro e Diversas outras (BALAGE, 2008).

Nos últimos anos a IA vem se desenvolvendo a passos largos, principalmente no campo da robótica, que segundo estudos realizados no Japão e EUA, deve ser uma das 10 linhas de pesquisa com mais trabalhos, à nível mundial, nas próximas décadas (GONCALVES, 2005). No ano passado (2012) os avanços da robótica foram postos a prova:

Em 26 de novembro de 2011, a bordo da nave MSL (*Mars Science Laboratory*), o robótico *rover* batizado de *Curiosity* partiu da terra com destino a Marte (WALL, 2012). Após uma viagem de oito meses e 563 milhões de quilômetros, em 06 de agosto de 2012, a sonda tocou a tênue atmosfera marciana a quase 21 mil quilômetros por hora cerca 17 vezes a velocidade do som. Para realizar o pouso foram usados paraquedas, propulsores e até um "guindaste" para colocar a *Curiosity* em solo com o menor impacto possível (Figura 1). Tudo isso acontece em sete minutos, que são chamados pela NASA de "sete minutos de terror". A diferença da meta de local de pouso foi menor do que 2,4 quilômetros (1,5 milhas) (HAMANN, 2012).

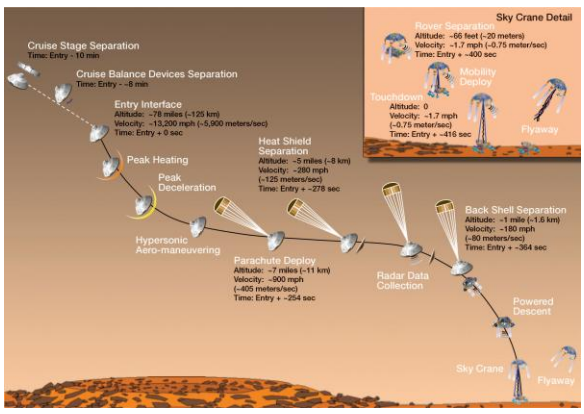


Figura 1 *– Fases do pouso do Curiosity*

Fonte: `http://goo.gl/1RMpt`

Momentos após o pouso, a *Curiosity* enviou as primeiras imagens do solo marciano (Figura 2). Numa delas, uma roda do veículo e outra sombra do jipe apareciam à frente do terreno pedregoso (HAMANN, 2012).

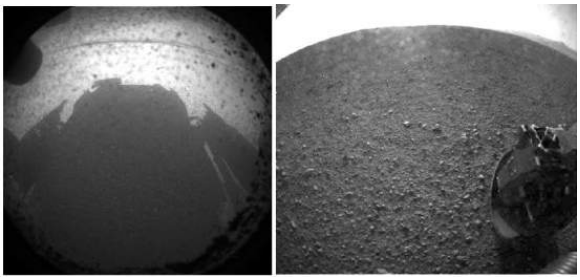


Figura 2 *– Primeiras imagens do Curiosity em Marte*
Fonte: `http://goo.gl/UVbng`

A operação de pouso foi considerada a mais complexa na história dos voos espaciais não-tripulados. Por causa da demora nas comunicações por rádio entre a Terra e Marte, todo o processo precisou ser autoguiado, sem a interferência dos técnicos (HAMANN, 2012).

Os principais objetivos científicos da missão MSL são para ajudar a determinar se Marte poderia ter suportado a vida, bem como determinar o papel da água, e para estudar o clima e geologia de Marte. Para isso a *Curiosity* (Figura 3) esta equipada com vários sensores, câmeras e ferramentas, constituindo um laboratório de pesquisa de última geração o que torna-o um *rover* peso pesado (899kg). A missão também tem o objetivo de preparar a exploração humana que deve acontecer por volta de 2020 (WALL, 2012).

Figura 3 *– Auto-retrato de Curiosity em Marte (31/10/2012)*
Fonte: `http://goo.gl/HLlZx`

O destaque da robótica atual se deve em grande parte a alguns fatores. Primeiramente, a redução de custos associados a hardware que tem permitido a produção de robôs cada vez mais baratos e com grande poder computacional, requisito importante para desenvolvimento de sistemas autônomos (BELO, 2006). Outro fator é o avanço tecnológico que tem permitido a

realização de cálculos computacionais necessários em tempo real e, este fato, tem possibilitado que novas descobertas e aplicações possam ser feitas em sistemas da robótica, tornando essa área uma fonte quase que inesgotável de pesquisa (GONCALVES, 2005).

1.2 Motivação

O grande objetivo da robótica está na criação de robôs autônomos capazes de executar tarefas sem a necessidade de intervenção humana. A autonomia de robôs é de muito interesse em uma grande quantidade de aplicações, principalmente aquelas que envolvem riscos, como operações de resgate e segurança, desarmamento de minas, uso policial, exploração submarina e espacial, aplicações nucleares e militares. Também são importantes as aplicações que envolvem benefícios econômicos, abrangendo a indústria de serviços, agricultura, manufatura e construção e, até mesmo, o uso em cirurgias médicas. (BELO, 2006)

Dentro do cenário apresentado, o estudo de robôs móveis e do planejamento de trajetória tem sido tema central de diversas pesquisas nos meios acadêmico e industrial. Essa área da robótica tem se mostrado um grande desafio, devido ao grande número de possíveis trajetórias a serem percorridas que possibilitam ao robô atingir o seu destino. Além deste item, existem dificuldades no desenvolvimento de algoritmos em aplicações de robôs móveis, de forma a garantir que o robô siga a trajetória previamente planejada (SIERAKOWSKI, 2006).

Uma área emergente neste foco é a cooperação entre robôs. Recentemente, o interesse por este tópico tem crescido, principalmente devido à possibilidade do cumprimento de tarefas complexas com a utilização de diversos robôs simples, ao invés de necessitar de um robô com comportamento complexo (SIERAKOWSKI, 2006). Esta abordagem surgiu no campo da inteligência artificial de enxame, bem como os estudos da biologia de insetos, formigas e outros campos na natureza, onde o comportamento de enxame ocorre.

A robótica de enxame é uma nova abordagem à coordenação dos sistemas de multirobot que consistem de um grande número de robôs simples. Supõe-se que um desejado comportamento coletivo emerge das interações entre os robôs e interações de robôs com o meio ambiente. Esta abordagem surgiu no campo da inteligência artificial de enxame, bem como os estudos de biologia de insetos, formigas e outros campos na natureza, onde o comportamento enxame ocorre.

CASTRO e ZUBEN (2003), citam alguns fatores que trouxeram interesse na robótica de enxame:

- Algumas tarefas são inerentemente muito complexas (ou impossíveis) de serem resolvidas por um único robô;
- Melhorias de desempenho podem ser conseguidas utilizando-se múltiplos robôs;

- A construção e utilização de robôs simples é geralmente muito mais barata, flexível e tolerante à falhas do que projetar um único robô, com alta capacidade de processamento de informação e sensores complicados;
- Estudos em Vida Artificial contribuíram para um maior entendimento e formalização de processos auto-organizados e fenômenos emergentes;
- As características construtivas e sintéticas da robótica coletiva contribuem para uma melhor compreensão de diversos fenômenos biológicos e sociais.
- Uma das características construtivas marcantes da robótica coletiva é a utilização de vários robôs com regras comportamentais simples e individuais.

Segundo CASTRO e ZUBEN (2003), existem quatro classes principais de aplicação para robótica de enxame:

- Exploração: trabalhos em ambientes hostis, inacessíveis ou de difícil comunicação (p. ex. superfície de outros planetas);
- Indústrias: grupos ou times de robôs deverão trabalhar em linhas de montagem;
- Medicina e cirurgia: existem diversas aplicações médicas que utilizam nano robôs, como microcirurgia e controle e local de aplicação de drogas dentro do organismo;
- Materiais inteligentes: são materiais feitos a partir de diversos módulos pequenos cada qual formando uma grande estrutura capaz de alterar sua forma e propriedades físicas dinamicamente.

Uma classe de algoritmos, em especial, ganhou muito destaque nesse ramo nos últimos anos, devido sua praticidade e robustez. Baseado no comportamento da colônia de formigas, o *Ant Colony Optimization* (ACO) vem sendo usado em várias áreas da computação e principalmente na robótica cooperativa (MELO, 2009).

1.3 Problema

O comportamento natural das formigas inspirou a construção de um algoritmo no qual um conjunto de "formigas" artificiais cooperam para a solução de um problema através da troca de informação via feromônio depositado sobre as trilhas. Esse algoritmo tem sido aplicado em problemas de otimização combinatória, como o problema do caixeiro viajante. A analogia adotada advém do fato das formigas conseguirem restabelecer rotas interrompidas por obstáculos,

de maneira que a nova rota é a mais curta possível. Nesse algoritmo, formigas "virtuais" são enviadas para várias cidades e, após cada interação, a quantidade de feromônio em cada trilha é atualizada proporcionalmente à distância percorrida pelas formigas, selecionando o percurso ótimo entre as cidades (BEVILACQUA et al., 1999).

O algoritmos das formigas se mostrou muito eficaz na resolução de problemas de otimização de combinatória, essa característica é visada na robótica cooperativa. Um dos principais problemas em aplicar esse algoritmo em robôs é a estigmergia.

O conceito de estigmergia foi introduzido por *Pierre-Paul Grasse*, na década de 1950, para descrever a comunicação indireta que ocorre entre os indivíduos nas sociedades de insetos sociais (BONABEAU, 1999). Pode ser definido como um exemplo particular de sinergia ambiental ou espacial. A alteração de um ambiente por um indivíduo, desencadeia um estímulo noutros indivíduos que, por sua vez, tendem a intervir e a modificar esse ambiente (MOURA; RAMOS, 2001).

A modelação artificial das colónias de formigas é baseada no conceito de estigmergia artificial, definido como: formas de interações indiretas que ocorrem quando um indivíduo da colônia modifica de alguma forma o ambiente e, algum tempo depois, outro indivíduo responde a essa modificação. Essa modificação é feita por feromônio que são marcadores químico-olfativos. A estigmergia que proporciona a comunicação entre as formigas, permite que seja encontrado, sem utilizar a percepção visual, o menor caminho entre a colônia e uma fonte de alimentos (SILVA et al., 2005).

Devido à robótica não contar ainda com sensores olfativos de grande precisão, a implantação desse algoritmo em enxames de robôs tem se mostrado um grande desafio.

A ideia desse trabalho é criar uma forma mais simples e funcional de simular o feromônio e criar um algoritmo, derivado do ACO, para um enxame de robôs que realizá busca e coleta de matérias de um ponto ao outro percorrendo um caminho ótimo.

1.4 Abordagem

Esse trabalho apresenta a construção de um algoritmo baseado no algoritmo das formigas para ser aplicado em enxames de robôs e a construção de um sistema para representar os feromônios. Para isso é enfatizado um estudo sobre a inteligência de enxame, robótica e algoritmo das formigas. Devido a construção e testes com robôs ser algo trabalhoso, o projeto contou com o desenvolvimento de simuladores para agilizar testes e melhorias do algoritmo. O simulador foi desenvolvido no Blender 3D (Capitulo 3), software de modelagem gráficas com engine para jogos. Os robôs foram construídos com base no Arduino, que é uma plataforma para prototipagem de projetos eletrônicos.

1.5 Objetivos

Esse trabalho é de caráter pratico tendo como área foco a robótica de enxame, tem por objetivo desenvolver um sistema baseado, no comportamento de coleta de alimento das formigas, para ser empregado à enxame de robôs. O sistema devera ser capaz de executar em ambiente real sem a interferências de manipuladores.

Como principal objetivo foi desenvolvido um algoritmo baseado no algoritmo das formigas que pode ser empregado no controle de enxames de robôs e também uma forma de representar o feromônio que é o ponto chave do funcionamento desse algoritmo e também a maior barreira no uso dele em enxame de robôs.

As áreas de estudos que foram necessárias para atingir esses objetivos são:

- Estudar a inteligência de enxame;
- Estudar o comportamento das formigas;
- Estudar o algoritmo das formigas;
- Estudar a comunicação entre robôs;

1.6 Contribuições

Visto o grande potencial da robótica de enxame esse trabalho visou ampliar as suas possibilidades explorando formas diferentes de usar algoritmos de inteligência como o algoritmo das formigas para o controle e navegação de enxames de robôs. As principais contribuições atingidas ao longo desse projeto foram:

- Construção de um simulador de colônia de formigas conhecido como Ants3D. No simulador, o comportamento das formigas de forragear, coletar alimento e forma trilhas fica bem evidente.
- Construção de um algoritmo para navegação de robô por trilhas de pontos coloridos;
- Uso do Blender como ferramenta interface gráfica para sistemas robóticos;
- Uso do mouse como sistema de mensuração do deslocamento e rotação de robôs;

1.7 Trabalhos Relacionados

SERAPIAO (2009) apresenta uma breve revisão de alguns dos mais recentes métodos bioinspirados baseados no comportamento de populações para o desenvolvimento de técnicas de

solução de problemas. Estão presentes nesse estudo as principais metaheurísticas de inteligência de enxame como: otimização por colônia de formigas, otimização por enxame de partículas, algoritmo *shuffled frogleaping*, coleta de alimentos por bactérias e colônia de abelhas. Cada uma mostrando descrições detalhadas do pseudo-codigo e suas devidas aplicações e melhorias no decorre do tempo.

Para uma melhor compreensão do potencial dos algoritmos apresentados foram realizados dois estudos de caso. O primeiro estudo está voltado para o problema de otimização global para minimização de funções. O segundo caso é o uso desses métodos como ferramenta de aplicação para a solução do problema de despacho econômico de carga em sistemas de energia. Os estudo foram realizados com os algoritmos PSO[1], SFL[2], BFO[3] e ABC[4]. Todos os algoritmos foram implementados de acordo com os procedimentos descritos nas suas versões originais. Os métodos foram todos implementados no Matlab 7.4.0 (MathWorks).

COELHO e NETO (2004), apresentam estudo de caso de otimização para os problemas de atribuição quadrática (*quadratic assignment problems*). Para solução do problema foi empregado o algoritmo das formigas com várias configurações. Para se avaliar o desempenho do ACS[5], foram realizados testes com problemas *benchmark*. Para cada teste foram realizados 20 experimentos com um máximo de 5000 atribuições (iterações) para otimização do QAP[6] por meio do ACS. Notou-se pelos resultados dos testes que o desempenho do ACS foi aprimorado ou deteriorado de acordo com o ajuste realizado nos parâmetros de configuração do ACS. Entretanto, o ACS provou ser muito robusto, desde que soluções próximas as melhores soluções conhecidas foram obtidas.

MIAZAKI (2007), realizou estudos de controle de um grupo de robôs para a solução coletiva das tarefas de exploração do ambiente e localização de objetos. Foi desenvolvido um sistema de controle de navegação baseado em colônia de formigas para um time de robôs, Esse sistema tem como base a utilização de marcadores ou feromônios artificiais, que podem ser depositados pelos robôs para marcar determinadas posições do ambiente.

Usando várias tecnológias, como processamento de imagens, robótica, simulação, comunicação via rede e computação bioinspirada, chego-se a contrução de um sistema que une robôs reais e simulados simultaneamente.

COSTA (2007a) em sua dissertação de mestrado investiga o papel da comunicação indireta nas tarefas de exploração e forrageamento, e como isso afeta as decisões de um agente simples e traz um comportamento emergente útil à toda colônia de formigas. O trabalho teve como

[1] *Particle Swarm Optimization*
[2] *algoritmo shuffled frog-leaping*
[3] *Bacterial Foraging Optimization*
[4] *Artificial Bee Colony*
[5] *ant colony systems*
[6] *quadratic assignment problem*

propósito modelar e desenvolver um ambiente para simulação de multi-agente, mais especificamente, de formigas artificiais, com o objetivo de permitir o estudo de técnicas de navegação e exploração de ambientes baseados em comunicação indireta entre os agentes.

Ao final do projeto foi criada uma ferramenta de simulação pronta para a execução de diversos experimentos acerca da comunicação indireta em sistemas de múltiplos agentes autônomos, através da criação de uma interface rica em painéis de configuração claros e objetivos, dando suporte ao estudo e controle de diversas características. Essas características, tais como taxa de evaporação e limiar de ativação do feromônio, por sua vez se encontram amparadas e fundamentadas na revisão bibliográfica apresentada na dissertação.

NEUFERT (2006), em sua monografia realizou estudos sobre algoritmo para cálculo de melhor rota de forma estática utilizando o algoritmo de Otimização por Colônia de Formiga. O estudo tem foco no problema dos Veículos Terrestres Autoguiados (Automated Ground Vehicles - AGV) que, quando submetidos ao universo de locomoção grande apresentam dificuldades em realizar do cálculo do caminho mais curto.

Ao longo do projeto chegou-se ao algoritmo ShortestPath-ACO, desenvolvido com base nos algoritmos AS[7], ACS, Simple-ACO e ACO. O algoritmo e capaz de realiza o cálculo do menor caminho em um grafo, dada a coordenada inicial e a coordenada final. Sua natureza heurística garante um bom desempenho em grafos com uma quantidade significativa de nodos e arestas. O projeto deve também o desenvolvimento de um veiculo terrestre composto de um GPS (*Global Positioning System*) que fornecia as coordenadas para o algoritmo gerar a melhor rota.

1.8 Organização do texto

Esta monografia está dividida em seis capítulos.

No capítulo um é apresenta a area do projeto, o problema abordado, objetivos, contribuições e a organização da monografia.

O capítulo dois apresenta o levantamento bibliográfico necessário para a realização do trabalho.

No terceiro capitulo mostra como foi concebido o projeto, ferramentas ultilizadas e tecnologicas desenvolvidas.

No quarto capítulo expoe a implementação dos simuladores baseados no comportamento das formigas.

No capítulo cinco estão apresentados os resultados obtidos, avaliando o funcionamento

[7] *ant systems*

dos simuladores.

No sexto capítulo é feita a conclusão e são observados, pontos que podem ser explorados futuramente em trabalhos da area da robótica de enxame.

2 Capitulo

2.1 Computação Natural

Com o surgimento dos computadores os homens começaram a interagir de outra forma, tanto entre si quanto com a natureza, esta passou a ser usada como fonte de inspiração para o desenvolvimento de novas técnicas para resolução de problemas complexos (XAVIER, 2013).

A computação natural entrou em cena por meados dos anos 1940, mas recebeu uma grande atenção nas últimas três décadas (CASTRO, 2007). É definida como uma abordagem computacional que busca estudar, compreender e aplicar, padrões complexos encontrados na natureza, utilizando-os como base para resolução de problemas, desenvolvimento de novas tecnologias e aperfeiçoamento de sistemas já existentes (CASTRO, 2006).

Segundo CASTRO (2004), existem várias razões para se estudar a computação natural:

- Possibilidade de desenvolver novas ferramentas computacionais para a solução de problemas complexos de engenharia;
- A possibilidade de projetar dispositivos (computacionais) que simulam, emulam, modelam e descrevem sistemas naturais;
- A possibilidade de sintetizar novas formas de vida;
- A possibilidade de utilizar materiais e mecanismos naturais, como cadeias de DNA[1] e dispositivos quânticos, como novos paradigmas de computação em substituição aos computadores atuais baseados em silício.

A computação natural pode ser dividida em três grandes partes (CASTRO; ZUBEN, 2005):

- Computação inspirada na biologia (bioinspirada): utiliza a natureza como fonte de inspiração para o desenvolvimento de novas técnicas de solução de problemas, como as redes neurais artificiais, a computação evolutiva, a inteligência de enxame e os sistemas imunológicos artificiais.
- Biologia inspirada na computação: trata-se basicamente de um processo de síntese que objetiva criar formas, padrões e comportamentos similares àqueles conhecidos na natureza. Além disso, algumas áreas visam o desenvolvimento de organismos artificiais;

[1] Ácido desoxirribonucleico

- Biocomputação: corresponde ao uso de mecanismos naturais para computar. Trata-se de um novo paradigma de computação, cujo objetivo principal é substituir as arquiteturas computacionais conhecidas atualmente. Por exemplo, cadeias de DNA e bits quânticos, são utilizados como estruturas de dados para o desenvolvimento de novas arquiteturas computacionais.

Portanto, a computação natural é uma linha de pesquisa que depõe contra a especialização de disciplinas. Ela mostra, com suas três principais áreas de atuação, que o conhecimento em diversas linhas de pesquisa é necessário para uma maior compreensão da natureza, para o estudo e simulação de sistemas e processos naturais, e para o desenvolvimento de novos paradigmas de computação. Físicos, químicos, engenheiros, matemáticos, biólogos, etc., todos contribuem e trocam idéias para o desenvolvimento da computação natural.

É importante salientar que o desenvolvimento da computação natural também resulta em benefícios para as ciências naturais (biologia, química e física). Diversas ferramentas, algoritmos e sistemas computacionais desenvolvidos em computação natural são empregadas para solucionar problemas em áreas como biologia, bioinformática, imunologia, etc., e também podem ser empregados como modelos abstratos de fenômenos naturais, podendo resultar assim em um melhor entendimento da natureza (CASTRO; ZUBEN, 2005).

2.2 Computação bioinspirada

A computação Bioinspirada tem por objetivo construir sistemas computacionais semelhantes a seres vivos (utilizando conhecimento das ciências biológicas) e utilizar tais sistemas para melhorar nosso entendimento da Biologia e da Natureza. Os sistemas biológicos inteligentes caracterizam-se por serem projetados pela evolução natural. Algumas de suas sub-áreas são: redes neurais artificiais, computação evolutiva (Algoritmos Genéticos), robótica, vida artificial, colônias de formigas e inteligência de enxames. (CARVALHO, 2003)

CASTRO (2007), destaca dois objetivos da computação bioinspirada:

- Conceber modelos teóricos que podem ser implantados em computadores, fiéis o suficiente para investigar os mecanismos naturais, de modo a reproduzir qualitativamente e quantitativamente alguns dos seus funcionamentos.

- Estudo de fenômenos naturais, processos e modelos teóricos, para o desenvolvimento de sistemas computacionais e algoritmos capazes de resolver problemas complexos. A motivação, neste caso, é o de proporcionar (alternativa) solução técnicas para os problemas que não podem ser (satisfatoriamente) resolvidos por outras técnicas mais tradicionais, tais como linear, não-linear e programação dinâmica.

A Computação Bioinspirada desenvolve algoritmos e ferramentas, baseado em processos naturais. Baseia-se na capacidade computacional do sistema nervoso para resolver problemas de reconhecimento de padrões, quando trabalha com redes neurais artificiais. Analisa as propriedades emergentes de colônias de organismos em solucionar problemas de otimização, aproximando a computação da lógica de enxames. E, através da Computação Evolutiva, busca, na capacidade da Natureza, melhorar a adaptação dos indivíduos ao ambiente no decorrer de várias gerações (CARVALHO, 2003).

2.3 Inteligência de enxame

Inspirada nas características de seres que vivem em grupos, nasceu um novo campo de estudo dentro da inteligência artificial: a Inteligência de enxame ou de banco, ou de colônia, no inglês, *Swarm Intelligence*.

Um enxame tem sido definido como um conjunto de agentes (móveis) que possam comunicar-se diretamente ou indiretamente (actuando em seu ambiente local) com os outros e que, coletivamente, realizam uma resolução distribuída de problemas (BOULET, 2010).

Formigas, abelhas, cupins são exemplos de insetos que vivem socialmente, apesar de não possuir um cérebro muito desenvolvido, não sendo individualmente muito inteligentes, conseguem realizar trabalhos complexos e sofisticados (INOVACAOTECNOLOGICA, 2012).

Em um formigueiro, os indivíduos são “bestas”, não possuem nenhuma visão de conjunto e não sabem como o que eles fazem se compõe com os atos dos outros indivíduos. Mas, ainda que as formigas isoladamente sejam “estúpidas”, sua interação produz um comportamento globalmente inteligente (LEVY, 1998).

Os bandos de pássaros, rebanho de animais como carneiros, crescimento de bactérias ou amebas, cardumes ou mesmo multidões de seres humanos, também estão sendo estudados no campo da Inteligência de Enxame (IE).

"Inteligência coletiva": incluir qualquer tentativa de projetar algoritmo ou resolver problemas através de sistemas distribuidos inspirados no comportamento de insetos sociais e sociedades de outros animais (BONABEAU et al., 1999).

Sistemas de inteligência de enxame são tipicamente constituídos por uma população de simples agentes interagindo localmente uns com os outros e com o ambiente. A inspiração muitas vezes vem da natureza, especialmente sistemas biológicos. Os agentes seguem regras muito simples, e apesar de não haver estrutura de controle centralizado para ditar como deve se comportar individualmente, as interações entre tais agentes leva ao surgimento de comportamento global "inteligente".

As propriedades principais de um sistema de inteligência de enxame são (Millonas, 1994):

- Proximidade – os agentes devem ser capazes de interagir;
- Qualidade – os agentes devem ser capazes de avaliar seus comportamentos;
- Diversidade – permite ao sistema reagir a situações inesperadas;
- Estabilidade – nem todas as variações ambientais devem afetar o comportamento de um agente;
- Adaptabilidade – capacidade de adequação a variações ambientais;

Uma colônia de insetos sociais é, sem dúvida, um sistema de resolução de problemas descentralizada, composta de muitos interatuantes relativamente simples. Os problemas diários resolvidos por uma colônia, incluem achar comida, construir ou ampliar um ninho de forma eficiente com divisão de trabalho entre os indivíduos, de forma eficiente alimentando a ninhada, respondendo a externa desafios, espalhando alarme, etc. Muitos desses problemas têm contrapartes em engenharia e ciência da computação. Uma das mais importantes características dos insetos sociais é que eles podem resolver estes problemas de uma forma muito flexível com robustez: a flexibilidade permite a adaptação a ambientes em mudança, enquanto a robustez dota a colônia com a capacidade de funcionar, mesmo que alguns indivíduos possam não desempenhar as suas tarefas. Finalmente, os insetos sociais têm limitadas capacidades cognitivas: é, portanto, simples para projetar agentes, incluindo agentes robóticos, que imitam seu comportamento em algum nível de descrição (BONABEAU et al., 1999).

2.4 Formigas Reais

No mundo real, as formigas andam sem rumo, pelo menos inicialmentem, (Figura 4) até que, encontrada comida, elas retornam à colônia deixando um rastro de feromônio. Se outras formigas encontram um desses rastros, elas tendem a não seguir mais caminhos aleatórios. Em vez disso, seguem a trilha encontrada, retornando e inclusive enfatizando se acharam alimento (Figura 5). Com o transcorrer do tempo, entretanto, as trilhas de feromônio começam a evaporar, reduzindo, assim, sua força atrativa. Quanto mais formigas passarem por um caminho predeterminado, mais tempo será necessário para o feromônio da trilha evaporar. Analogamente, elas marcharão mais rapidamente por sobre um caminho curto, o que implica aumento da densidade de feromônio depositado antes que ele comece a evaporar. A evaporação do feromônio também possui a vantagem de evitar a convergência para uma solução local ótima: se a evaporação não procedesse, todas as trilhas escolhidas pelas primeiras formigas tornar-se-iam excessivamente atrativas para as outras e, neste caso, a exploração do espaço da solução

delimitar-se-ia consideravelmente. Todavia, quando uma formiga encontra um bom (curto) caminho entre a colônia e a fonte de alimento, outras formigas tenderão a seguir este caminho, gerando assim feedback positivo, o que eventualmente torna um determinado caminho mais interessante (Figura 6).

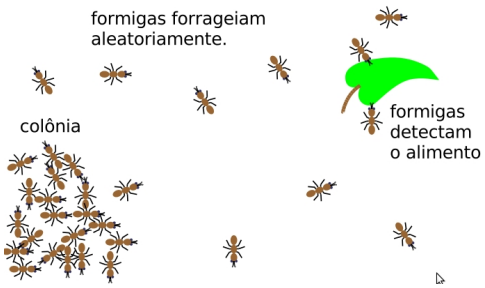


Figura 4 *– Formigas Forrageando*

Fonte: `http://goo.gl/ZfCym`

Figura 5 *– Inicio da formação da trilha*

Fonte: `http://goo.gl/ZfCym`

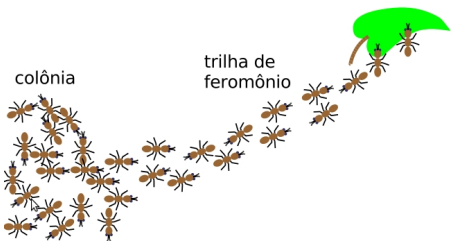


Figura 6 *– Trilha de melhor caminho formada*

Fonte: `http://goo.gl/ZfCym`

2.5 A ponte binaria

Um particular e brilhante experimento foi concebido e executado por DENEUBOURG et al (1990), que utilizou uma ponte de ligação dupla entre um ninho de formigas argentina,

Iridomyrmex humilis e uma fonte de alimento. Eles correram experiências variando a razão entre o comprimento dos dois ramos da ponte dupla (DORIGO; STÜTZLE, 2004).

Uma colônia de insetos sociais é um superorganismo, sem cérebro, e cada trabalhador tem acesso a informações muito local. Como, então, uma colônia é capaz de comportamento, complexo coletivo? (DENEUBOURG et al., 1990).

No primeiro experimento, a ponte tinha dois ramos de igual comprimento. No início, as formigas foram deixadas livres para se mover entre o ninho e a fonte de alimento e a porcentagem de formigas que escolheu um, ou o outro dos dois ramos, foi observado ao longo do tempo. O resultado foi que, embora as escolhas fossem aleatórias na fase inicial, ocorreu, eventualmente, que todas as formigas migraram para um mesmo ramo.

Este resultado pode ser explicado como se segue. As formigas coordenam suas atividades através estigmergia, uma forma de comunicação mediada por modificações do meio ambiente. Na procura de alimento, as formigas depositam uma substância química sobre o solo, essa substância é feromônio que vai agir como meio de comunicação para o enxame. Ao se deslocarem, as formigas formam uma trilha dessa substância, que se torna mais atraente na medida em que mais formigas são atraídas para um determinado caminho e depositam mais feromônio nele, e assim por diante, até que finalmente as formigas convergem para um único caminho (DORIGO; STÜTZLE, 2004).

No segundo experimento, a relação de comprimento entre os dois ramos foi definida de modo que o ramo longo leva o dobro do tempo para ser percorrido. Neste caso, na maioria dos testes, após algum tempo todas as formigas escolheram usar somente o ramo curto como pode ser visto na Figura 7.

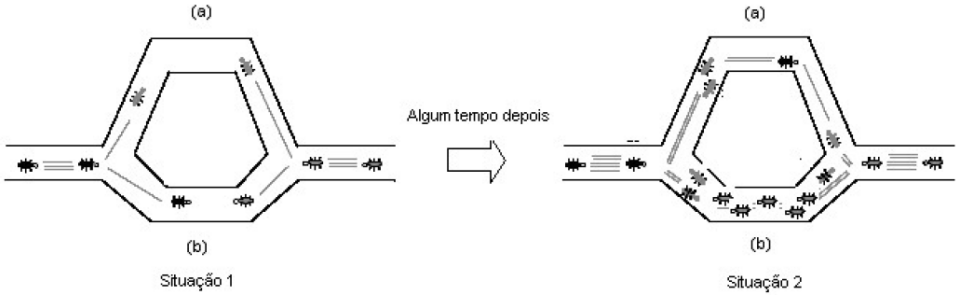


***Figura 7** – Pontes dupla de comprimento diferente*

Fonte: (DORIGO; STÜTZLE, 2004)

As formigas que escolhem o caminho (b) chegam à fonte de alimentos e retornam para a colônia mais rápido do que as formigas que escolhem o caminho (a). Dessa forma, o caminho (b) acumula uma concentração maior de feromônio. A situação 2 da Figura 7 mostra que a concentração maior de feromônio no caminho (b) influencia outras formigas a seguirem este caminho. Porém, algumas formigas continuarão a escolher o caminho mais longo. Este fato ocorre

porque existe um fator aleatório que influencia, também, na decisão da formiga. Entretanto, depois de certo tempo, a quantidade de formigas que escolhe o caminho mais longo diminui significativamente. A consequência deste fato é que a concentração de feromônio torna-se o parâmetro principal na decisão da escolha do caminho (SILVA et al., 2005).

DENEUBOURG et al (1990) apresentam a formulação matemática do experimento. Quando uma formiga atravessa a ponte no ramo A deixa I unidades de feromônio, portanto temos A_i para ramo A e B_i para o ramo B. A próxima formiga que chega à ponte vai escolher entre o ramo A ou B, com probabilidades $prob_A$ e $prob_B$, dependendo de A_i e B_i. Tendo escolhido, em seguida, a formiga contribui para a feromônio no ramo escolhido. Assim:

$$prob_A = \frac{(K+A_i)}{(k+A_i)+(k+B_i)} \qquad (prob_A + prob_B = 1) \qquad (1)$$

$$A_{i+1} = A_i + \delta, \qquad B_{i+1} = B_i + (1-\delta) \qquad (A_i + B_i = i) \qquad (2)$$

Onde δ é uma variável estocástica que assume um valor de 1 ou 0, com probabilidade $prob_A$ e $prob_B$, respectivamente. Em outras palavras, A é aumentado de uma unidade se a formiga escolhe o ramo A ou B é aumentado se a formiga escolhe o ramo B.

A equação (1) é uma função de escolha geral simples, que quantifica a maneira em que uma concentração mais alta no ramo A dá uma maior probabilidade da formiga ir para o ramo A, dependendo dos valores absolutos e relativos de A_i e B_i. O parâmetro N determina o grau de não linearidade da escolha, um valor elevado para N significa que uma ramificação tem apenas concentração de feromônio muito ligeiramente superior ao outro. A formiga seguinte que passa terá uma probabilidade muito elevada de escolhe-lo. O parâmetro k corresponde ao grau de atração atribuída a um ramo não marcado, ou em outras palavras, quanto maior k, maior é a marcação necessária para que o ramo se torna-se o escolhido.

2.6 *Ant Colony Optimization*

Inspirados pelos experimentos de coleta de alimentos por formigas, (DORIGO et al., 1996) desenvolveram um modelo computacional com tais conceitos para resolver o problema do caixeiro viajante. O algoritmo proposto baseia-se em um grupo de "formigas artificiais" que liberam feromônio durante o seu trajeto e seguem "trilhas de feromônio artificial" para encontrar o menor caminho entre todas as cidades (SERAPIAO, 2009).

A Otimização por Colônia de Formigas (*Ant Colony Optimization* - ACO) surgiu na década de 90 e foi proposta por DORIGO e GAMBARDELLA (1997) como uma meta-heurística para resolver problemas de otimização combinatória. É um método de busca distribuída e cooperativa inspirada no comportamento de formigas reais. O algoritmo procura imitar a técnica empregada pelas formigas para estabelecer rapidamente o menor caminho entre uma fonte de

alimento e a colônia (LOPES et al., 2007).

Um algoritmo ACO constitui um sistema baseado em agentes que simula o comportamento natural das colônias de formigas na procura por alimentos, desenvolvendo mecanismos de cooperação e aprendizado (DORIGO; STÜTZLE, 2004). As formigas coordenar as suas atividades através estigmergia (Capitulo 2). Na procura de alimento, as formigas depositam um substância química sobre o solo, essa substância é o feromônio que vai agir como meio de comunicação para o enxame. Ao se deslocarem, as formigas formam uma trilha dessa substância, que se torna mais atraente na medida em que mais formigas escolhem a mesma trilha e depositam mais feromônio (DORIGO; STÜTZLE, 2004). O comportamento coletivo surge a partir dessa comunicação química.

Um dos padrões comportamentais mais surpreendentes exibidos pelas formigas é a capacidade de certas espécies de formigas para encontrar o que os cientistas denominam de caminhos mais curtos. É esse padrão que os cientistas da computação se inspiram para desenvolver algoritmos (DORIGO; STÜTZLE, 2004).

Algoritmos ACO são as técnicas algorítmicas mais bem-sucedidas e amplamente reconhecidas com base em comportamentos de formigas. O seu êxito é evidenciado pela extensa gama de diferentes problemas a que tenham sido aplicados e, além disso, pelo fato dos algoritmos ACO são usados para muitos problemas atualmente resolvidos por algoritmos de alto desempenho. (DORIGO; STÜTZLE, 2004)

2.6.1 Pseudocódigo do ACO

Descrição do pseudocódigo criado por NETO e COELHO (2006). É interessante notar que este algoritmo faz referência a uma lista tabu, que corresponde a um conjunto de pontos já visitados pelo agente e que não devem ser visitados novamente.

- A probabilidade de escolha da ida de uma cidade i para uma cidade j é representada pela equação(3) onde $T_{ij}(t)$ representa a quantidade de feromônio existente no caminho entre as cidades i e j no intante t; N_{ij} representa a visibilidade da cidade j em i e α, β and R são constantes.

- A equação (4) define a forma como feromônio é atualizado. O termo p.T representa a parcela que evapora, senso P uma constante de evaporação tal que 0<P<1. O termo Dt se referencia à quantidade de feromônio depositada pela formiga entre os instantes t e t+n.

- A quantidade de feromônio depositada pela formiga k em um caminho é definida na equação (5), aonde Q representa uma constante e L é o comprimento total percorrido pela formiga.

- A quantidade de feromônio depositada por todas as M formigas em um caminho de i a j é dada pela equação (6).

$$p_{i.j}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum[\tau_{ik}(t)]^{\alpha}.[\eta_{ik}]^{\beta}} \qquad (3)$$

$$\tau_{ij}k(t+n) = p.\tau_{ij}k(t) + \Delta\tau_{ij}k \qquad (4)$$

$$\Delta\tau_{ij}^{k} = \begin{cases} . \frac{Q}{L_k} se\ a\ formiga\ k\ ultiliza\ desta\ trilha\ em\ seu\ caminho. \end{cases} \qquad (5)$$

$$\Delta\tau_{ij} = \Sigma_{k=1}^{m} \Delta\tau_{ij}k \qquad (6)$$

1. Inicialização:

   $t = 0, \quad nc = 0$ (t representa o tempo, NC representa o número de ciclos)

   Para cada caminho (i, j), selecione um valor inicial $\tau_{ij}(t) = c$ para a intensidade do feromônio e $\Delta\tau_{ij} = 0$

   coloque m formigas nos n nós

2. s=1 (s representa o índice da lista tabu)

   Faça de $k = 1$ a m

   Selecione a cidade inicial da k-ésima formiga em tabu(s)

3. Repita até que a lista tabu esteja cheia (n-1 vezes)

   s=s+1

   Faça de k=1 a m

   Escolha a cidade j para mover a formiga, com a probabilidade dada pela equação (3).

   Mova a k-esima formiga para a cidade j

   Insira a cidade j na lista Tabu

4. Faça de k=1 a m

   Mova a k-ésima formiga de tabu(n) a tabu(1)

   Determine o comprimento L_k do caminho percorrido Atualize o feromônio do último caminho encontrado, de acordo com as equações (4), (5) e (6)

2.6.2 Caracteristicas e Aplicações

Segundo ALMEIDA et al (2007), o ACO possui as seguintes características:

- É um algoritmo não-determinístico baseado em mecanismos presentes na natureza, isto

é, ele é baseado no comportamento de formigas para a determinação de caminhos através de suas colônias para procura eficiente de fontes de comida;

- É um algoritmo paralelo e adaptativo, pois uma população de agentes movê-se simultaneamente, de forma independente e sem um supervisor (não há um controle ou supervisão central);

- É um algoritmo cooperativo, pois cada agente (formiga) escolhe um caminho com base na informação (trilhas de feromônios) depositadas por outros agentes que tenham selecionado previamente o mesmo caminho. Este comportamento cooperativo tem ingredientes de autocátalise (catálise provocada por feromônios que se formam no próprio sistema reativo), isto é, o ACS providencia uma realimentação positiva, desde que a probabilidade de um agente escolher o caminho aumenta com o número de agentes que escolheu previamente aquele caminho.

MILLER (2007) em um artigo para revista NATIONAL GEOGRAPHIC BRASIL, descreve alguns usos do algoritmo das formigas em problemas do cotidiano de certas empresas:

> Em Houston, a companhia *American Air Liquide* adotou uma estratégia baseada nas formigas para resolver um problema empresarial. A empresa produz gases para fins industriais e médicos – nitrogênio, oxigênio, hidrogênio – em uma centena de usinas nos Estados Unidos e precisa entregar seus produtos em 6 mil locais, usando gasodutos, trens e 400 caminhões. Os custos de produção variam de acordo com oscilações regionais nos preços de energia. Em algumas regiões, a desregulamentação do mercado de energia (o preço da eletricidade varia de 15 em 15 minutos em algumas áreas do Texas) torna a situação ainda mais complicada.
>
> Com a ajuda do *Bios Group* (atualmente *Nu-Tech Solutions*), uma empresa especializada em inteligência artificial, a *Air Liquide* desenvolveu um modelo digital que envia bilhões de formigas digitais para que descubram onde estão as trilhas de feromônio mais fortes para as rotas de nossos caminhões."
>
> A *Air Liquide* associou a abordagem das formigas a outras técnicas de inteligência artificial de modo a levar em conta todas as permutações entre cronogramas de produção de suas usinas, condições climáticas e rotas de caminhões – milhões de possíveis decisões e resultados por dia. Todas as noites, o modelo computacional é alimentado com previsões de demanda e de custos de produção. "Leva quatro horas para fazer todo o processamento", conta Harper. "Mas todas as manhãs, às 6 horas,

temos um esquema para orientar-nos durante o dia." A empresa, teve reduções de custos significativas. "É enorme o ganho. Enorme."

Outras companhias também estão lucrando ao imitar o comportamento das formigas. Na Itália e na Suíça, frotas de caminhões para entrega de leite e laticínios, óleo para o aquecimento doméstico e alimentos em geral também recorrem às regras das formigas forrageiras para determinar as melhores rotas. Na Inglaterra e na França, companhias telefônicas ampliaram o tráfego de chamadas em suas redes depois de programarem as mensagens para que depositem feromônios virtuais em estações de distribuição, tal como as formigas deixam sinais para que suas companheiras encontrem as melhores trilhas.

Nos Estados Unidos, a companhia aérea *Southwest Airlines* testou um modelo desse tipo para melhorar o atendimento no aeroporto de *Phoenix*. Com 200 aviões decolando e pousando em duas pistas a cada dia, e usando pontos de embarque e desembarque em três terminais, a empresa queria assegurar que cada aeronave entrasse e saísse do aeroporto no menor tempo possível, mesmo que o vôo estivesse adiantado ou atrasado. *Doug Lawson* criou um modelo computacional do aeroporto, atribuindo a cada avião a capacidade de lembrar quanto tempo levou para se aproximar e se afastar de cada portão. Em seguida rodou o modelo de modo a simular a atividade de um dia no aeroporto.

"Os aviões são como formigas em busca do melhor portão", diz ele.Mas em vez de deixar feromônios virtuais ao longo do caminho, cada aeronave lembra-se dos portões mais rápidos e esquece-se dos mais lentos. Após muitas simulações, usando dados verdadeiros para variar os tempos de chegada e partida, cada avião aprendeu a evitar uma espera intolerável na pista. A *Southwest* ficou tão satisfeita que talvez use um modelo similar para agilizar os balcões de check-in.

3 Capítulo

3.1 Projeto dos simuladores

O estudo da robótica de enxame apresenta alguns problemas, o gasto para se criar um enxame de robôs e algo a ser considerado no projeto e também a dificultado de atingir uma arquitetura robótica que se adeque aos algoritmos desenvolvidos.

Uma solução usada nessa área de pesquisa é primeiramente implementar os sistemas em software para que através da simulação apropriada do robô real seja entradas e corrigidas falhas e assim evitar maiores gastos com execuções em sistemas reais (COSTA, 2007a).

3.2 Ferramentas

3.2.1 Blender

Blender, também conhecido como Blender3D, é um programa de computador para modelagem, animação, texturização, composição, renderização, edição de vídeo e criação de aplicações interativas em 3D, tais como jogos, apresentações e outros, através de seu motor de jogo integrado, o Blender Game Engine (FOUNDATION, 2012). Inicialmente era um projeto proprietário de um empresa chamada NaN (*Not a Number*), de Ton Rosendaal. A empresa era especializada em animação e produção de mídia e comunicação visual, isso lá nos anos 90. Porém uma crise econômica abalou-a, o que forçou a empresa a se desfazer de seu produto. Em 2002 Ton Rosendaal, fundou a Blender Foundation e iniciou uma campanha chamada “Free Blender”, para arrecadar dinheiro com intuito de liberar o programa como código aberto. Em 13 de outubro de 2002, o Blender passa a ter uma licença dupla: Blender License (BL) / GNU General Public License (GPL). O Blender possui ainda partes licenciadas sob a Python Software Foundation License (FOUNDATION, 2012).

O programa foi escrito inicialmente em C, e atualmente está escrito em C, C++ e, algumas partes, principalmente scripts embutidos, em Python, são multiplataforma, estando portanto disponível para diversos sistemas operacionais. Atualmente, suporta 25 idiomas, incluindo o português brasileiro, e está na versão 2.66, lançada em 21 de fevereiro de 2013 (FOUNDATION, 2012).

O Blender possui sua própria "Game Engine"(Máquina de Jogos) embutida, que possibilita a criação de aplicações interativas 3D. A *Game Engine* (Máquina de Jogos) do Blender ou a chamada BGE (Blender Game Engine) é utilizada principalmente para jogo, mas também

pode ser utilizada para criar praticamente qualquer tipo de software interativo 3D para outras finalidades, como visualizações de cenários 3D interativos para arquitetura, ou então pesquisas físicas em educação.

O núcleo da estrutura da BGE são os *Logic Bricks* (Blocos Lógicos). A finalidade dos *Logic Bricks* é oferecer uma interface visual simples e fácil de utilizar para fazer o desenvolvimento de aplicações interativas. Existem três tipos básicos de *Logic Bricks*:

- *Sensors* (Sensores), são os receptores dos estímulos do meio. Estes podem ser a fonte de um evento disparador como um Objeto próximo, uma tecla pressionada no Teclado, eventos temporizados, etc. Quando um Sensor é disparado, um pulso positivo é enviado para todos os controladores que estão ligados a ele.

- *Controllers* (Controladores) são como Blocos que coletam dados providos pelos Sensores. O trabalho dos *Controllers* é checar e combinar os Pulsos provindos do sensor para disparar a resposta apropriada como reação. Podem ser do tipo: AND, OR, XOR, NAND, NOR, XNOR EXPRESSION, PYTHON.

- *Actuators* (Atuadores) perfazem ações, como mover, criar Objetos, ou tocar um som. Os Atuadores iniciam as suas funções quando recebem um sinal ou pulso positivo a partir de um (ou mais) dos *Controllers* (Controladores).

Tambem é possivel escrever Jogos ultilizando a linguagem de programação Python, a BGE possui sua própria API Python, separada do restante do Blender, que pode ser utilizada para escrever *scripts* para controlar seu Jogo. Isto é feito criando um Controlador Python e o ligando-o a um Script Python.

3.2.2 Scratch

É uma nova linguagem de programação criada no *Media Lab do MIT* por Michel Resnick que surgiu por volta de 2007. Scratch é muito mais acessível que outras linguagens de programação, por se utilizar de uma interface gráfica que permite que programas sejam montados como blocos de montar, lembrando o brinquedo LEGO. Utiliza uma sintaxe comum a muitas linguagens de programação. E diferente de outras linguagens, não tem nenhum tipo de pontuação obscura. Cada bloco da linguagem contém um comando em separado, que podem ser agrupados livremente caso se encaixem. E os comandos podem ser modificados através de menus. O ambiente de desenvolvimento pode ser baixado gratuitamente em sua página. Já existem versões para Windows, Mac OS X e Ubuntu. E uma versão está sendo feita para o laptop XO (SCRATCH, 2013).

3.2.3 Mouse

Em 9 de dezembro de 1968 o mouse foi apresentado ao mundo pelo inventor Douglas Engelbart (ARRUDA, 2011; ANGELO, 2007; BEZERRA, 2010), resultado de um projeto que durou cinco anos no Instituto de Pesquisa de Stanford. Chamado inicialmente de XY *Position Indicator for a Display System*, seu modelo consistia em uma pequena caixa de madeira com apenas um botão e um cabo que saía de uma das extremidades (Figura 8), lembrando, de alguma forma, o rabo de um rato. Ele permitiu mais fácil manuseio e era mais preciso do que os *trackballs* e *joysticks* existentes até então. Infelizmente, durante os 15 primeiros anos, devido aos computadores daquela época não possuírem interfaces gráficas como as "janelas" e nem os editores de textos terem o cursor de tela, essa invenção ficou sem muita utilidade (BEZERRA, 2010; ANGELO, 2007).

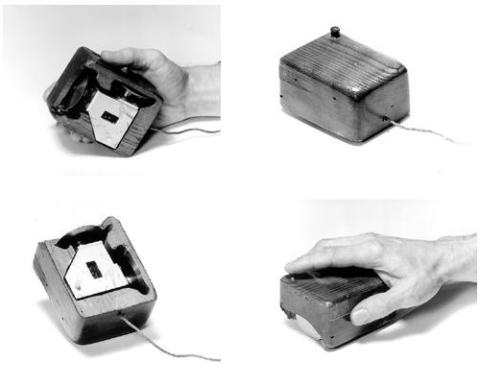


***Figura 8** – O primeiro Mouse*
Fonte: `http://goo.gl/PwGCa`

O mouse ganhou o mercado com os computadores pessoais da Xerox, como o Alto, de 1973, o primeiro PC a usar o conceito de desktop e a ter uma interface gráfica voltada para o uso do "rato" (ARRUDA, 2011). No ano de 1983 com o lançamento do Macintosh, a Apple passou a utilizar o mouse como apontador na tela, tornando-o um dispositivo integrado aos computadores desde então. Ao ver sucesso do mouse, a Microsoft resolveu utilizá-lo no seu novo sistema operacional, o Windows 3.1, onde era impossível navegar pelo sistema sem o dispositivo. Na época Douglas Engelbart não ganhou muita coisa além dos seus minutos de fama, pois vendeu a patente do mouse por míseros US$ 10.000. Somente em abril de 1997, Engelbart recebeu o prêmio Lemelson-MIT (um dos mais importantes no mundo dos inventores) e um prêmio de 500 mil dólares (BEZERRA, 2010).

Tipos de mouses

Atualmente existem diversos tipos e formatos de mouses que com o tempo vieram sendo aperfeiçoados e sofrendo inúmeras modificações. O modelo tradicional possui 3 botões, o esquerdo, o direito e o *scroll*, que é utilizado para rolar a barra de rolagem das páginas ou janelas

(BEZERRA, 2010). São classificados de acordo com a tecnologia de posicionamento, por um lado, e de acordo com a transmissão dos dados para unidade central, por outro (KIOSKEA, 2013).

Distingue-se, assim, várias grandes famílias de mouses:

- Os mouses mecânicos, cujo funcionamento se baseia numa bola (em plástico ou de borracha) embutida num chassis (em plástico) que transmite o movimento a dois rolos;
- Os mouses ópticos, capazes de determinar o movimento por análise visual da superfície sobre a qual deslizam;

Mouse mecânico

Esse mouse tem uma esfera de borracha semiembutida no corpo do mouse. Dentro dele existem dois pequenos rolos (é como se eles representassem os eixos X e Y de um plano cartesiano) que em contato com a esfera rola devido ao movimento do usuário (Figura 9) (BEZERRA, 2010).

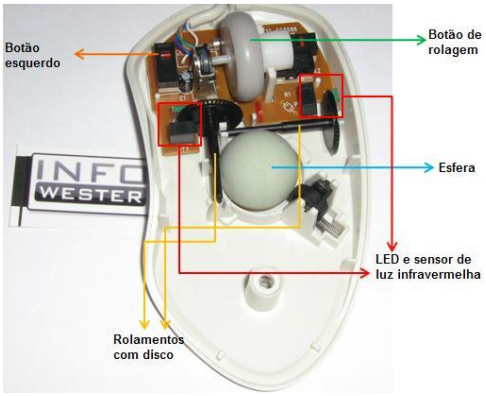


Figura 9 *– Mouse de Esfera*
Fonte: `http://goo.gl/TvlJ7`

Cada um destes rolos comportam um disco entalhado que gira entre um sensor infravermelho e um LED (Diodo electroluminescente). Quando o usuário movimenta o mouse, a esfera gira, fazendo com que os eixos e seus respectivos discos rodem. Quando isso ocorre, num momento a luz infravermelha passa pela perfuração e, no instante seguinte, a luz é bloqueada pela parte não perfurada do disco (Figura 10). Quando a luz passa, o sensor infravermelho retorna um bit (1), quando encontra um obstáculo, o fotodiodo retorna um bit nulo (0) (KIOSKEA, 2013). Um circuito existente no mouse "conta"quantas vezes a luz infravermelha passa pelas perfurações e transmite essas informações à máquina. Quanto mais rápido o usuário movimentar o mouse, mais rápido será a passagem e a não passagem, fazendo com que a seta na tela do computador se movimente em uma velocidade correspondente. Para determinar o sentido

de cada rolamento, isto é, se este gira no sentido horário ou no sentido anti-horário, há mais de uma técnica: alguns mouses utilizam dois pares de LEDs infravermelho e de sensores de luz infravermelha, e comparam as diferenças de passagens de luz entre eles para determinar o sentido. Outros, especialmente mouses mais recentes, conseguem fazer essa distinção através de uma combinação de LED infravermelho e sensor de luz infravermelha mais precisa, fazendo com que apenas um par seja suficiente para cada disco (ALECRIM, 2008). Com a ajuda destas informações, o computador pode conhecer a posição do cursor, e mesmo a sua velocidade.

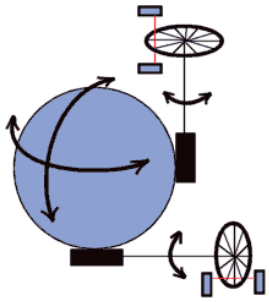


Figura 10 *– Funcionamento da bola*
Fonte: `http://goo.gl/XTwuC`

Mouse ótico

Apesar das pesquisas para os primeiros modelos de mouse ópticos terem começado em 1980, foi só em 1999 que o primeiro modelo comercial desse tipo de dispositivo foi lançado. O IntelliMouse com IntelliEye, da Microsoft, funcionava sobre quase qualquer tipo de superfície e apresentou melhoras muito significativas quando comparado com o mouse mecânico (ARRUDA, 2011).

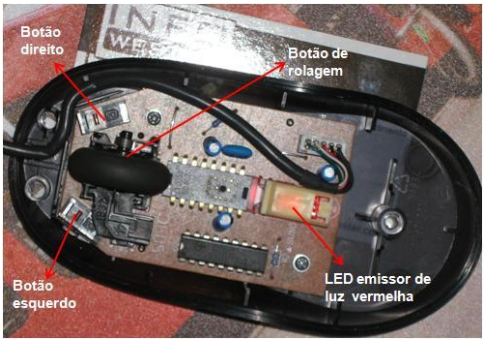


Figura 11 *– Mouse Óptico*
Fonte: `http://goo.gl/XTwuC`

Nesse novo produto as esferas deram lugar um sistema óptico que orienta o cursor da tela de acordo com os movimentos do usuário.

O sistema óptico dos mouses desse tipo é composto, basicamente, por um LED emissor de luz vermelha e um sensor (Figura 11) (geralmente, sensor CMOS, sigla de *Complementary*

Metal Oxide Semiconductor, ou CCD, sigla de *Charge Coupled Device*. Quando o mouse está em contato com uma superfície, a luz é emitida e refletida, isto é, "volta"ao mouse. Quando isso ocorre, o sensor age como se estivesse tirando uma fotografia daquele ponto e envia a imagem a um DSP (*Digital Signal Processor*), que a analisa. Esse processo é repetido constantemente e em uma velocidade muito alta. O DSP faz então uma espécie de comparação e análise dos padrões das imagens e consegue, com isso, entender para onde o mouse está sendo movimentado. O passo seguinte consiste em enviar essas informações ao computador para, finalmente, o cursor na tela ser orientado (ALECRIM, 2008).

Os mouses óticos funcionam em todas as superfícies que não são completamente lisas ou com nuances de cor. As vantagens principais deste tipo de dispositivo indicador em relação aos mouses mecânicos são: maior precisão e menos sujeira.

3.2.4 Sensor de Cor com LDR

Para criar um sensor de cor é necessário entender como as cores funcionam. A cor de um material é determinada pelas médias de frequência dos pacotes de onda que as suas moléculas constituintes refletem. Um objeto terá determinada cor se não absorver justamente os raios correspondentes à frequência daquela cor (ENSINAREVT, 2012). Isso quer dizer que quando a luz chega em um objeto azul, por exemplo, será refletido apenas a luz azul, no caso de um objeto vermelho, será refletido apenas a cor vermelha, e assim sucessivamente (VITALI, 2011).

Para saber que cor tem um objeto, é necessário um sensor capaz de medir intensidades luminosas. O sensor que pode ser usado é o LDR (*Light Dependent Resistor*), que é um dispositivo eletrônico que tem a sua resistência elétrica interna alterada pela incidência de luz.

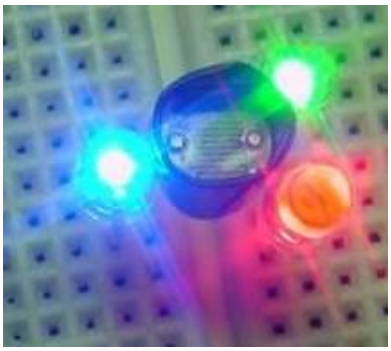


Figura 12 *– Sensor de cor*
Fonte: `http://goo.gl/VDBk1`

O objeto deve ser iluminado com uma luz de uma determinada cor que se pretende identificar (vermelho, verde, azul), e os dados captados pelo LDR. Para melhor resultados, deve ser usado leds RGB de alto brilho (VITALI, 2011). A figura 12 mostra o sensor montado.

Segundo SOCIETYOFROBOTS (2012), sensor de cor deve ser calibrado com as cores

que vai trabalhar. Por exemplo, suponha que o robô deve seguir uma linha branca em um chão cinza. Durante a fase de calibração o robô mede um valor analógico de 95 para o chão cinzento, 112 para a linha branca, e, em seguida, estes valores armazenados na memória. Agora, o robô está na linha, e um sensor lê 108. Como saber a cor?

Usando o método de limiarização, pega-se os dois números calibrados e divide por dois para encontrar o número do meio. Por exemplo, (95 + 112) / 2 = limiar. Qualquer coisa acima desse limite seria a linha branca, e qualquer coisa sob seria o piso cinza.

Para mais de duas cores deve ser usado a correspondência de similaridade. O que determinar quanto semelhante a cor do objeto é do valor calibrado. no exemplo linha branca, usando a correspondência de similaridade e um pouco de matemática gera a equação:

(nova_leitura - leitura_de_calibração) / leitura_de_calibração * 100 = similaridade

Exemplo:

cinza chão = (108 - 95) / 95 * 100 = 13,7

linha branca = (108 - 112) / 112 * 100 = 3,6

comparar: linha branca < chão cinza portanto, o sensor vê uma linha branca

Programar o sensor de cor é bem simples. Por exemplo, um sensor com três LEDs de cores diferentes, este seria o seu pseudocódigo:

```
ligar LED verde
esperar 50ms
sensor de registro de leitura G
desligar LED verde
ligar LED vermelho
esperar 50ms
sensor de registro de leitura R
desligar o LED vermelho
ligar LED azul
esperar 50ms
sensor de registro de leitura B
desligar LED azul
```

Agora, usando um algoritmo de correspondência de similaridade, com números pré-calibrados, o robô pode, então, identificar cores.

A distância entre o objeto a qual pretende-se identicar a cor e o LDR tem forte impacto na calibração, portanto para um melhor funcionamento a calibração tem que ser feita na distância de trabalho do sensor.

3.2.5 Sistema de posicionamento por mouse

Um dos grandes desafios da robótica é determinar com exatidão a localização de um robô móvel durante a sua operação. Existem diversos métodos para determinar a localização de robôs móveis. O método mais usado é a odometria, no qual as velocidades, linear e angular do robô, são integradas ao longo do tempo para determinar a sua posição (MOLINA et al., 2009).

MOLINA et al (2009) apresenta um método de medição e estimação de posição e velocidade, utilizando o sensor de fluxo óptico presente nos mouses comerciais. Os Problemas na determinação da posição de robôs móveis incluem os erros sistemáticos introduzidos pelos dispositivos mecânicos utilizados e erros não sistemáticos causados por terrenos irregulares, ou mesmo pelo escorregamento das rodas no chão, nos casos de terrenos de baixo atrito.

O sensor existente no mouse óptico utiliza o fluxo óptico para a determinação do seu deslocamento. Posicionando corretamente este sensor em um robô móvel é possível determinar a posição do robô sem estar suscetível a erros devidos ao deslize das rodas no piso, por exemplo.

As conclusões realizadas por MOLINA et al (2009), mostra que o sensor do mouse óptico apresenta uma alta variância para o ruído de estimação de velocidade. Como uma solução ótima, é utilizado o Filtro $\alpha - \beta$ que traz minimização da variância do ruído de medição. Entretanto, o sensor apresentou dificuldades para determinar movimento composto por translação e rotação. Como trabalho futuro, é proposto o desenvolvimento de uma estratégia de decisão para determinação do tipo de movimento executado pelo robô (translação, rotação).

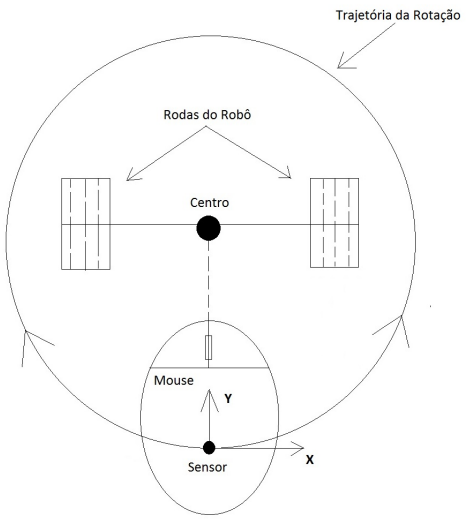


***Figura 13** – Posicionamento do Mouse no robô*

Uma solução para o problema na determinação dos movimentos de translação e rotação seria deslocar o eixo de rotação que nos testes de MOLINA et al (2009) fica no centro do sensor,

para extremidade superior do mouse. A figura 13 mostra como deve ficar a montagem no robô.

Ao executar uma rotação o mouse registra o deslocamento no eixo X, portanto o valor de X corresponde a rotação e Y ao deslocamento. Assim, pode se determinar a posição e a direção do possível robô.

4 Capítulo

4.1 Implementação

Ao longo da pesquisa foram desenvolvidos três simuladores tendo como objetivo chegar a um sistema funcional que pode ser empregado a enxame de robôs. O primeiro simulador desenvolvido nessa pesquisa, que foi batizado de Ants3D, traz o comportamento das formigas de forragear, coletar alimento, formar trilhas e encontro do menor caminho entre o ninho e a fonte de alimento. A construção desse simulador foi de fundamental importância para chegar a um algoritmo que pode ser empregado a enxame de robôs.

O feromônio é o principal mecanismo utilizado pelas formigas para controlar seu comportamento de enxame (Referencia). A representação de feromônios para ser usados por robôs é algo complexo, pois exige uma substancia com as mesmas características do feromônio das formigas e um sensor capaz de detectar essa substância no meio. Para resolver essa dificuldade foram propostas duas hipóteses:

- Unir o Ants3D com robôs reais, onde o simulador fica responsável por controlar a detecção, liberação e evaporação dos feromônios;
- Representar o feromônio por pingo de tinta onde um escala de cor representaria a intensidade do feromônio;

Nos próximos tópicos será descrito a implementação desses simuladores.

4.2 Descrição do Simulador 1

O ants3D (Figura 14) foi desenvolvido todo no Blender 2.49. A vantagem do simulador é que proporciona testes mais rápidos e permite testar várias configurações de modo mais fácil do contrario se fosse feito com robôs.

Figura 14 *– Menu do simulador*

4.2.1 Elementos do simulador

Formiga

A formiga é o elemento de maior importância, ela contém toda a inteligência artificial que representa o comportamento de uma formiga real.

Na figura 15, o círculo amarelo destaca o objeto que representa a formiga, a parte de cor rosa é a cápsula psíquica, ela é responsável pela física e movimento da formiga. A circunferência azul destaca as variáveis da formiga que estão descritas na Tabela 1.

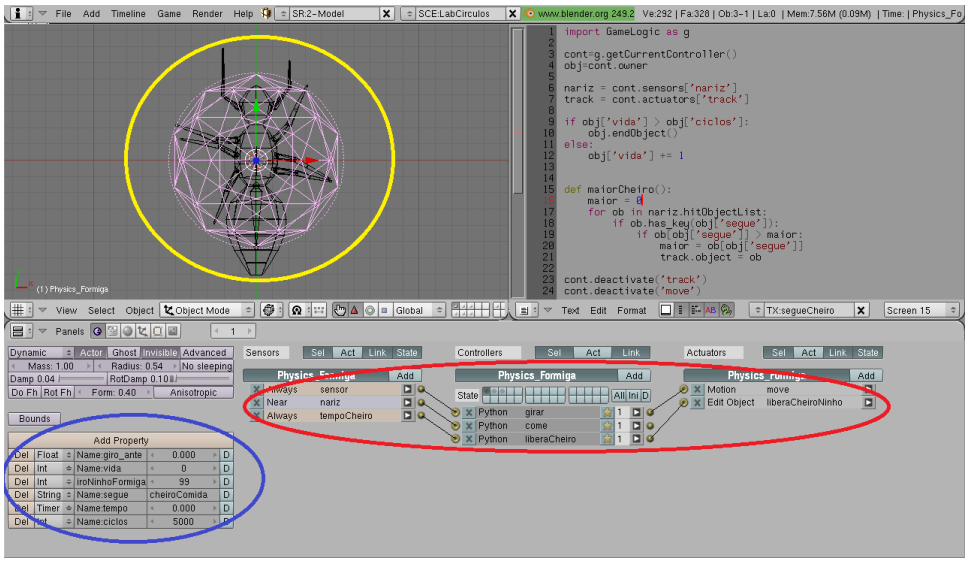


Figura 15 *– Objeto Formiga*

Tabela 1 *– Variáveis da Formiga*

Nome	Tipo	Descrição
giro_ante	float	Armazena o ultimo ângulo de giro
vida	int	Conta o numero de ciclos do algoritmo
cheiroNinhoFormiga	int	Quarda a intensidade do feromônio ninho que formiga libera
segue	int	Nome do feromônio que deve ser seguido
tempo	timer	Usada para marca o tempo de cada direção
ciclos	int	Quantidade de ciclos que formiga pode executar

O círculo vermelho destaca a parte lógica. Essa parte é formada por sensores, controles e atuadores. São dois sensores principais, o Near que representa o sentido do olfato da formiga e gera um campo de colisão em torno do objeto de tamanho configurável e o sensor *collision*, que detecta a colisão com ninho e o alimento.

Lógica da formiga

O algoritmo da formiga se encontra dividido em 6 *scripts*, organizados em dois estados (Figura 16). No primeiro estado, quando não a alimento e nem cheiro de alimento, a formiga anda aleatoriamente, soltando o feromônio ninho. Ao encontra uma trilha de alimento ou alimento, a formiga passa para o segundo estado, onde pegam o alimento e entregam no ninho.

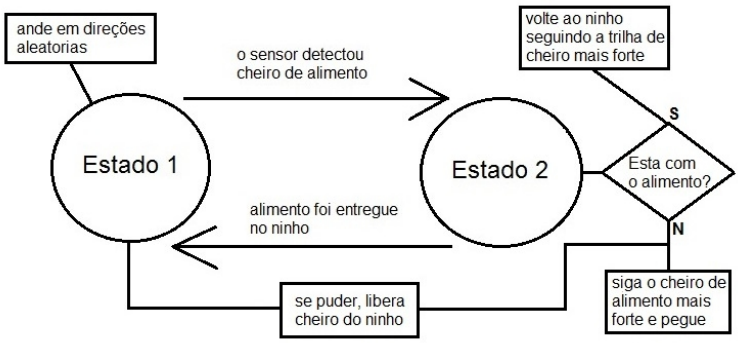


Figura 16 *– Lógica da formiga*

De todos os 6 scripts, o mais importante e o "seguecheiro". Seu trabalho é fazer a formiga seguir as trilhas mais fortes de feromônio. A função abaixo está presente dentro do *scripts*. Basicamente ela pega todos os objetos feromônios que foram detectados pelo Near e seleciona o maior. Depois, o objeto é passado para atuador track que direciona a formiga para o objeto feromônio escolhido.

```
1 def maiorCheiro():
2   maior = 0
3   for ob in nariz.hitObjectList:
4     if ob.has_key(obj['segue']):
5       if ob[obj['segue']] > maior:
6         maior = ob[obj['segue']]
7         track.object = ob
```

Alimento

O alimento representado pelo objeto quadrado verde (Figura 17) que possui duas variáveis: Tabela 2

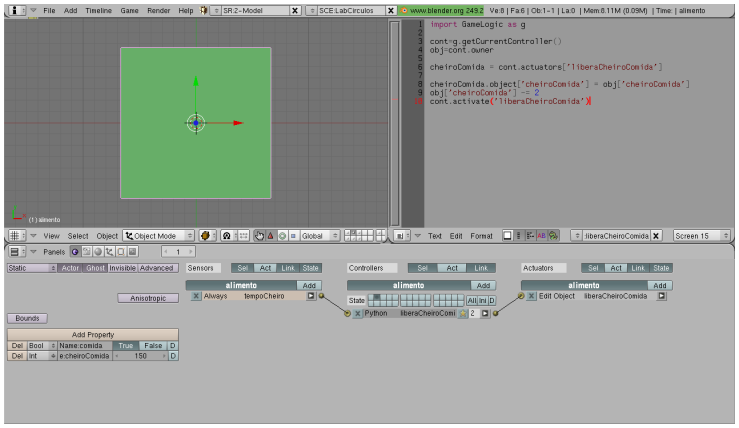


Figura 17 *– Objeto Alimento*

Tabela 2 *– Variáveis do Alimento*

Nome	Tipo	Descrição
comida	bool	Usado pelo sensor da formiga. Identifica como uma comida não encontrada
cheiroComida	int	Define a intensidade da fonte de alimento

Para um melhor desempenho, quem libera o feromônio da comida é o próprio alimento ao invés da formiga. Essa abordagem não comprometeu a teoria do algoritmo das formigas, pois o feromônio da comida só é liberado quando a formiga pega o alimento. Enquanto o alimento não for encontrado por uma formiga ele permanece em um estado inativo, ao ser pego pela formiga muda para o estado onde passa a liberar o feromônio do alimento.

Feromônio Ninho

O feromônio do ninho é representado pelo objeto triangular vermelho (Figura 18). Possui uma variável: Tabela 3

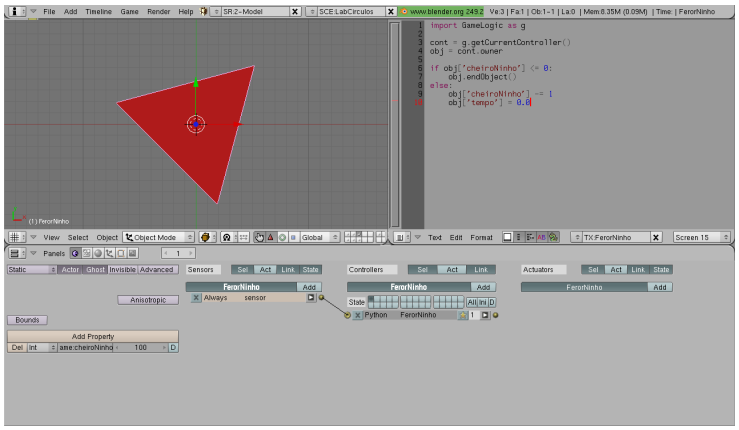


Figura 18 *– Objeto feromônio ninho*

Tabela 3 *– Variáveis da Feromônio Ninho*

Nome	Tipo	Descrição
cheiroNinho	int	Define a intensidade do cheiro do ninho

Ao ser liberado pela formiga ele passa a decrementar a sua variável “cheiroNinho” que quando chega a zero, o objeto é deletado. Essa fução serve para implementar o feedback negativo.

Feromônio Comida

O feromônio do ninho é representado pelo objeto triangular verde (Figura 19). Possui uma variável: Tabela 4

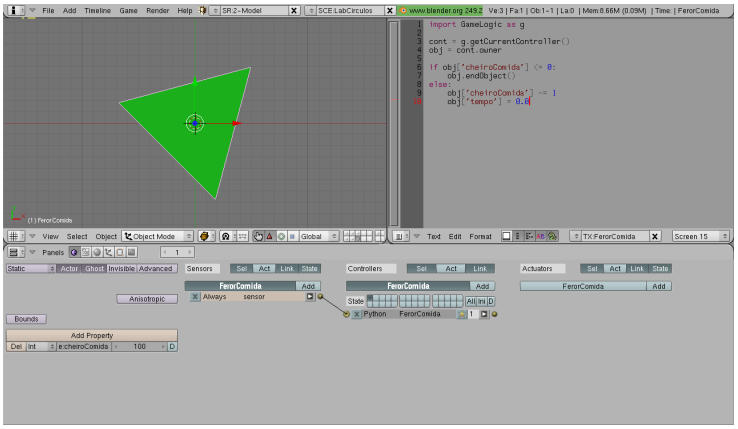


Figura 19 *– Objeto feromônio comida*

Tabela 4 *– Variávies da Feromônio Comida*

Nome	Tipo	Descrição
cheiroNinho	int	Define a intensidade do cheiro da comida

O funcionamento é igual ao feromônio do ninho: decrementa a sua variável “cheiroComida” que quando chega a zero, o objeto é deletado

Ninho

O ninho realiza o trabalho de carregar a variável “cheiroNinhoFormiga” de cada formiga que se aproxima dele (Figura 20).

Para uma aproximação da realidade, o ninho possui uma variável que armazena a energia do mesmo. Essa energia é incrementada quando um objeto alimento é entregue no ninho por uma formiga. A energia mínima do sistema é 500. Se tiver 15 pontos de energia sobrando uma formiga nasce. As variáveis do ninho são: Tabela 5

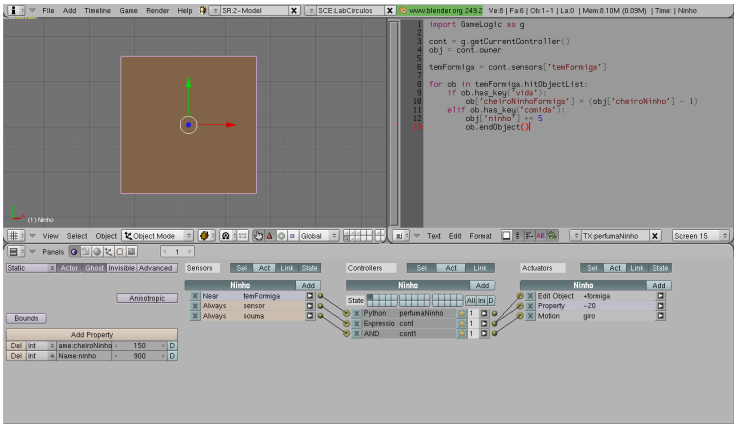


Figura 20 *– Objeto ninho*

Tabela 5 *– Variáveis do Ninho*

Nome	Tipo	Descrição
cheiroNinho	int	Guarda o cheiro do ninho, deve ser o maior cheiro do sistema
ninho	int	Granda a energia do ninho

4.3 Descrição do Simulador 2 - Feromônio de tinta

Através de estudos de como representar o feromônio para um enxame de robôs, uma hipótese foi criada com intuito de representar o feromônio com tinta ou algum tipo de pigmento que pudesse ser depositado no chão. Para testar essa hipótese, foi construído (Figura 21) um simulador no Scratch 1.4.

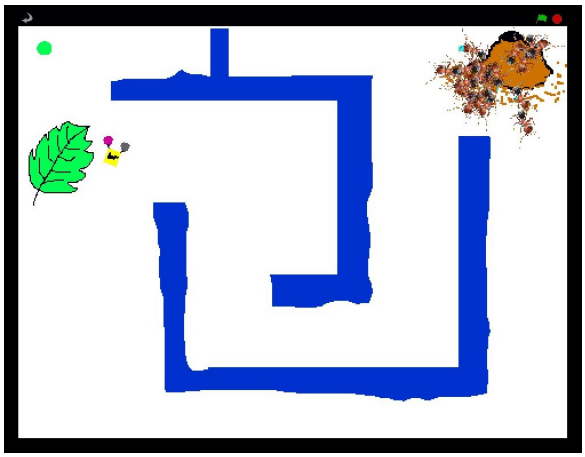


Figura 21 *– Simulador do feromônio de cor*

Foi criado um enxame de formigas que se move de forma aleatória e libera pingos de tinta quando as formigas estão se deslocando a procura de alimento. Os pingos possuem uma determinada cor que representar a intensidade do feromônio, essa cor muda conforme a intensidade do feromônio diminui. A escala de valores das cores e mostrada na Figura 22.

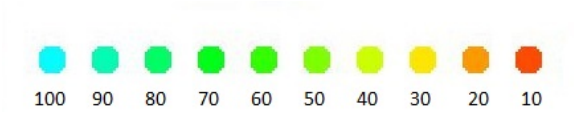


Figura 22 *– Escala de valores do feromônio*

O pseudocódigo abaixo descreve o comportamento da formiga (figura 23-A).

```
1:  enquanto não tocar na comida faça
2:      se passou 3 segundos então
3:          se feromônio for maior que 10 então
4:              decremente 10 unidades
5:          senão
6:              feromônio recebe 0
7:          fim se
8:          Zere o contador de tempo
9:      fim se
10:     se tocar no ninho então
11:         feromônio recebe 100
12:     fim se
13:     Sorteei um ângulo e aplique o giro
14:     Sorteei a quantidade de passos e mova
15:     se tocar na parede então
16:         volte
17:     fim se
18:     Pingue a feromônio
19: fim enquanto
```

O cenário ao qual as formigas e robôs devem trabalhar constitui um pequeno labirinto com dois caminhos possíveis para se chegar a comida. Um caminho é ligeiramente maior que o outro, essa diferença serve para testar a eficácia do sistema de feromônio de cores em achar o menor caminho. A figura 21 mostra o início da simulação.

Um objeto ninho figura 23-D foi criado com objetivo de aumentar a intensidade do feromônio de cada formiga, ao tocar no ninho a formiga recarrega seu feromônio.

O objeto comida figura 23-C tem a finalidade de ser o ponto de parada para a formiga, pois as formigas tem somente a função de criar a trilha para as robôs poderem chegar no ninho pelo menor caminho.

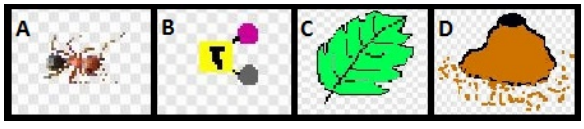


Figura 23 *– Objetos do simulador*

Um enxame de robôs (figura 23-B) fica posicionado no final do labirinto à espera da trilha de feromônio gerada pelas formigas. Cada robô possui dois sensores de cor posicionados a frente como duas antenas. A função do sensor é determinar a direção que o robô deve seguir. A prioridade dos sensores é para as cores mais próximas do ninho, nesse caso são valores mais próximos do 100. Por exemplo: se o sensor da direita detectar um pigmento de cor 50 e o da esquerda detectar um pigmento de cor 80, a prioridade será para a direção da esquerda.

Essa técnica torna possível o robô seguir a melhor trilha deixada pelas formigas e chegar ao ninho. O pseudocódigo do robô pode ser visto abaixo.

```
1:  cor = 100
2:  cont = 1
3:  enquanto não tocar no ninho faça
4:      Guarde a posição de x e y
5:      Mova alguns passos
6:      se cor < 100 E sensor da direita ou da esquerda tocou numa cor maior que 100 então
7:          cor recebe mais 10
8:          Mude cont para 1
9:      fim se
10:     se cont < 36 então
11:         se sensor da direita e da esquerda tocou na cor então
12:             Mude cont para 1
13:         senão
14:             se sensor da direita toco na cor então
15:                 Gire para direita
16:                 Mude cont para 1
17:             senão
18:                 se sensor da esquerda toco na cor então
19:                     Gire para esquerda
20:                     Mude cont para 1
21:                 senão
22:                     volte a posição de x e y inicial
23:                     Gire para um lado qualquer
24:                     Incremente cont com 1
25:                 fim se
26:             fim se
27:         fim se
28:     senão
29:         Decremente cor com 10
30:         Mude cont para 1
31:     fim se
32: fim enquanto
```

4.4 Descrição do Simulador 3 - Robô virtual

Uma forma de representar o feromônio das formigas em robôs, sem exigir sensor de alta custo, é usar feromônios artificiais. Como base para a construção desse sistema foi utilizado Ants3D por tanto foi construído também no Blender, mas foi usado a versão 2.54. O primeiro passo foi construir um robô que pudesse trabalhar em conjunto com o simulador.

4.4.1 Robô

O robô tem o papel de explorar o ambiente, encontrar o alimento e retornar a origem estabelecendo a melhor rota.

Figura 24 *– Robô visto por baixo*

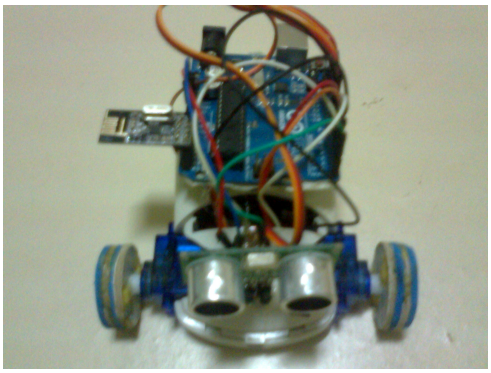


Figura 25 *– Robô visto por cima*

Para a construção do protótipo robô foi utilizado:

- Um sensor ultrassónico HC-SRO4;
- Dois micros servo motor modificado para rotação continua;
- Uma placa Arduino Uno rev3;
- Um mouse de esfera;
- Um sensor de cor;
- Um modulo de comunicação wireless Nrf24l01 de 2.4 ghz;

A função do sensor ultrassônico é proporcionar a percepção do ambiente ao qual o robô foi imerso. Com esse sensor é possível perceber obstáculos é possíveis limitações do meio. Ao ser identificado um obstáculo no ambiente real ele é imediatamente criado no ambiente virtual. Esse processo torna possível a concepção de um mapa virtual do ambiente. Dessa forma, o melhor caminho gerado no ambiente virtual será também o melhor no ambiente real.

O papel do mouse é realizar a sincronização entre o robô e simulador. Como descrito no capítulo 3, o mouse fornece as coordenadas X e Y, velocidade e ângulo de rotação, com essas

informações tornando possível a sincronização em um ambiente real e o simulado.

O robô também possui um sensor de cor para procurar marcadores no chão. Os marcadores são representações do ninho das formigas e o alimento que deve ser coletado. Para minimizar custos do projeto foi construído um sensor de cor simples, esse processo esta descrito no capítulo três. Os marcadores possuem uma cor específica que pode ser detectada pelo sensor, Existem dois tipos de marcadores: o ninho, representado pela cor vermelha e o alimento, representado pela cor verde.

4.4.2 O ambiente virtual

O ambiente virtual deve conter as partes mais importantes do mundo real que será simulado. Os elementos desse sistema são formados por um chão que representa uma possível área a ser explorado pelo robô no mundo real, um marcador que representa o ninho virtual das formigas (esse ponto e a origem do robô) e uma formiga com o mesmo algoritmo das formigas do Ants3D, com o diferença de poder se comunicar com o robô através da serial. A figura 26 mostra a tela inicial do simulador.

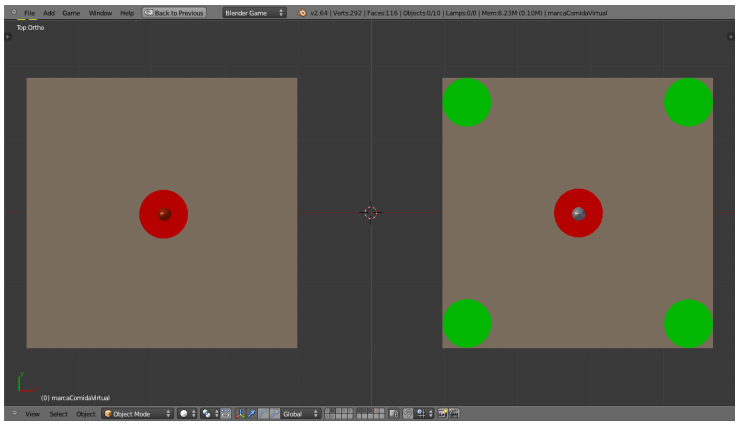


Figura 26 *– Tela inicial do simulador*

4.4.3 Comunicação serial no Blender

O Blender trabalha junto com a linguagem python para criar controles mais precisos de jogos e simulações. Por padrão, o python não vem com suporte à comunicação serial, mas com a instalação do módulo “pySerial” o python passa a ter todas a funcionalidades necessárias para o acesso à porta serial.

Como o Python no Blender é usado na forma de scripts, há uma dificuldade em manter a conexão aberta. Ao término da execução do *script* todas suas variáveis são desalocadas da memória e dessa forma, não há como armazenar a porta que foi aberta em uma variável do

script.

O blender conta com um recurso de variável global chamado de “GlobalDict” que salva o nome e os valores de qualquer tipo de variáveis para que possa ser passado entre as cenas. Com esse recurso é possível salvar as conexões que foram abertas, e reocupera-las quando necessário.

Logo abaixo, o trecho do código responsável por abrir a conexão e salvar no “globalDict “. No inicio é verificado se existe uma conexão aberta salva no “globalDict”. Caso não, é aberta uma conexão e salva no “globalDict”.

```
if(g.globalDict.get("portaR")==None):
  ser = serial.Serial(3,9600,timeout=0.01,writeTimeout=0.01)
  t.sleep(0.5)
  ser.write('a'.encode('ascii'))
  t.sleep(0.1)
  g.globalDict["portaR"] = ser
else:
  ser = g.globalDict["portaR"]
```

As funções para enviar e receber dados da serial são bem simples (abaixo). O módulo “pySerial” trabalha com Strings Unicode que não são viáveis para a comunicação com o Arduino. Os métodos “encode()” e “decode()” são usados para igualar a forma de comunicaçao entre o Blender e Arduino.

```
def envmsn(ser,aux):
  ser.write(aux.encode('ascii'))

def recmsn(ser):
  aux=ser.read()
  return bytes.decode(aux)
```

A comunicação entre o robô e o simulador é feita através de strings com marcadores específicos. Para andar para frente é enviado o “w”, “c” é igual a comida, “g” é para o giro, ele vem seguido do valor da rotação

4.4.4 Funcionamento do simulador

No início o robô se encontra posicionado no marcador referente ao ninho e, de igual forma, está a formiga. O robô logo sai à procura do aliemento, que é representado por um círculo verde, realizando movimentos aleitórios com intuito de percorrer toda a área em volta do ninho.

Todo o movimento do robô é passado para a formiga através de comandos enviados pela serial. Como num espelho, a formiga virtual realiza todos os movimentos do robô deixando uma trilha de feromônio.

Ao encontrar o marcador do alimento, o robô envia uma mensagem para o ambiente virtual, logo um representação do local do alimento é marcada no ambiente simulado.

A formiga passa para seu segundo estado, onde segue as trilhas de feromônio para encontrar o ninho. Nesse momento, o robô passa a ser guiado pela formiga através de comandos envidados pela serial.

5 Capítulo

5.1 Experimentos

Para realizar os testes cada simulador apresenta um cenário que representa um possível ambiente real, cada ambiente tem em comum uma representação do ninho e outra do alimento.

O primeiro simulador (Ants3D) foi submetido a três testes cada um em ambiente diferente onde as formigas virtuais devem encontra o melhor caminho entre a fonte de alimento e o ninho.

O segundo simulador é testada a eficácia da representação do feromônio por pontos de tinta, onde um enxame de formigas virtuais percorre um pequeno labirinto na busca de alimento liberando os pingos de tinta. A trilha de tinta criada é seguida por um enxame de robôs que deve achar a saída do labirinto pelo menor caminho.

Os testes no terceiro simulador teve o objetivo de observar a comunicação entre a meio simulado e a representação do meio real, também foi observado o sincronismo entre os dois meio que é vital para que esse sistema funcione.

5.2 Testes com Ants3D

5.2.1 Testes 1

No teste 1, apresenta um cenário cheio de obstáculos cinzas cilíndricos com infinitas possibilidades de caminhos. No centro, o quadrado marrom representa o ninho que tem a função de produzir formigas e carregar o feromônio do ninho de uma formiga que passe perto dele.

Os quadrados verdes são as fontes de alimento e, para melhorar os testes, elas são inesgotáveis. As 3 fontes foram colocadas de uma forma que a distância da fonte para ninho é a mesma em todas as fontes.

No início são liberadas 10 formigas, que vão procurar as fonte de alimento andando em direções aleatórias (Figura 27).

Com o decorrer do teste, as 3 fontes de alimento são encontradas (Figura 28). O processo de recrutamento das formigas começa a surgir. As formigas que estão carregando o alimento vão depositando o feromônio que representa a trilha de alimento. Formigas que estão, por perto andando aleatoriamente detectam a trilha e começam a segui-la. Esse processo da inicio as novas trilhas.

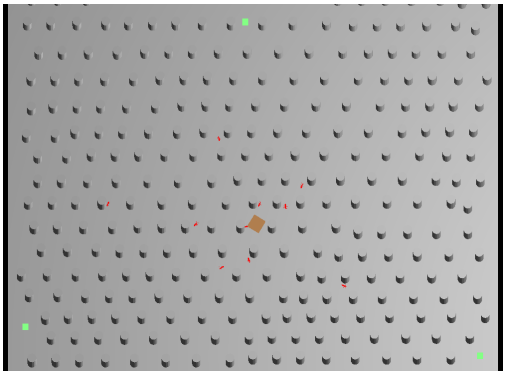


Figura 27 *– Teste 1: inicio*

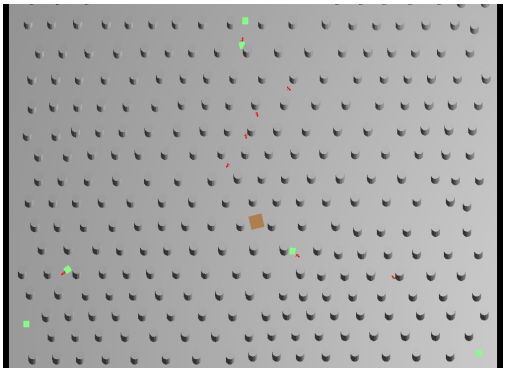


Figura 28 *– Teste 1: encontrado o alimento*

Como as fontes de alimento estão em distâncias iguais, pode ser que mais de uma trilha seja formada, mas sempre uma ficará mais forte devido a flutuações aleatórias no andar das formigas. Na Figura 29 monstra a conclusão do experimento.

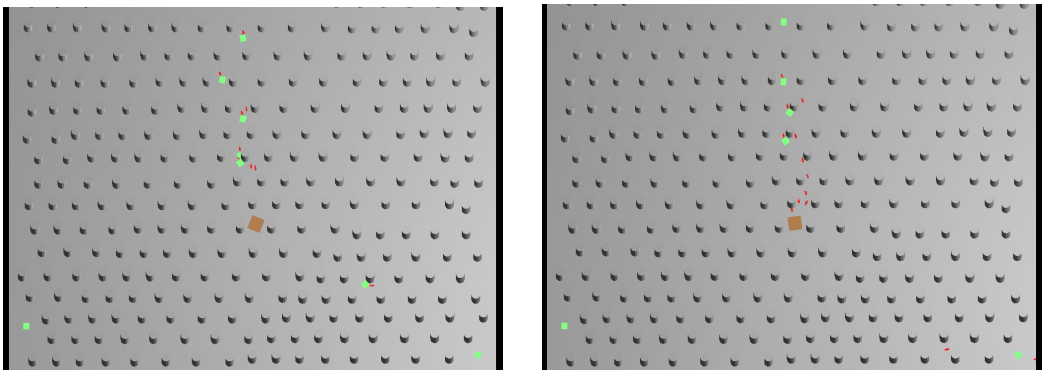


Figura 29 *– Teste 1: Formação da Trilha*

5.2.2 Testes 2

No teste 2, a um cenário com obstáculos circulares grandes. As fontes de alimento foram colocadas com distâncias diferentes, a com menor distância para o ninho está localizada

embaixo. Devido as configurações do cenário, 5 caminhos são possíveis mas só um tem a menor distância.

As formigas se espalham e logo 4 trilhas são formadas (Figura 30), como algumas apresentam distâncias muito grandes se comparadas com as outras, elas logo são deixadas de ser seguidas. Esse processo acontece quando as formigas chegam no ninho, e percebem um trilha como cheiro mais forte, que são as trilha que possui o feedback positivo (Capítulo 2) mais rápido por serem mais curtas (Figura 31). Em pouco tempo o caminho mais curto prevalece (Figura 32).

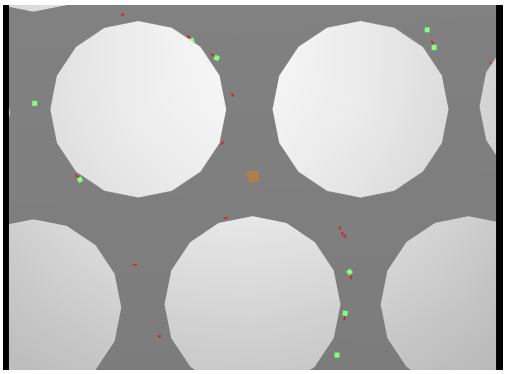


Figura 30 *– Teste 2: formação de 4 caminhos*

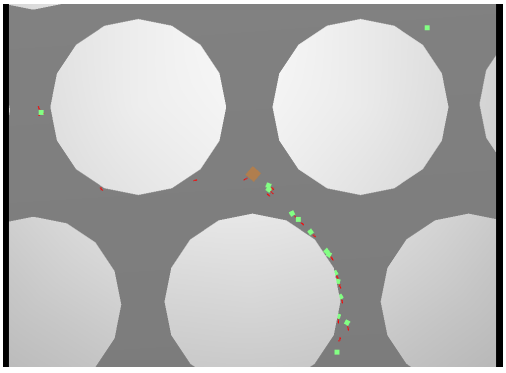


Figura 31 *– Teste 2: formação da trilha*

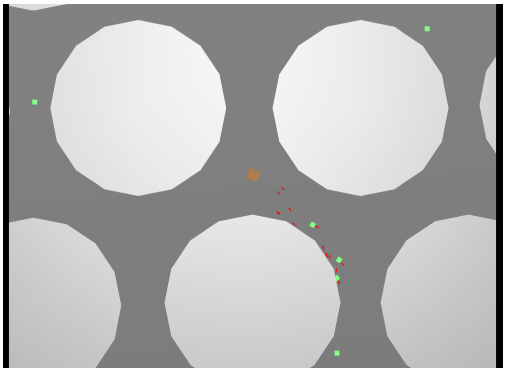


Figura 32 *– Teste 3: Melhor caminho*

5.2.3 Testes 3

O teste 3 é famoso teste da ponte binária descrito por DENEUBOURG et al (1990) que provou que as formigas encontram o menor caminho entre o ninho e o alimento através de pontes duplas. O êxito nesse teste é a prova que o algoritmo das formigas fora implementado da forma correta.

No início as formigas andam aleatórias à procura de alimento percorrendo os dois ramos da ponte, mas um dos caminhos possui uma distância maior, logo trilhas são formadas nos dois lados da ponte (Figura 33).

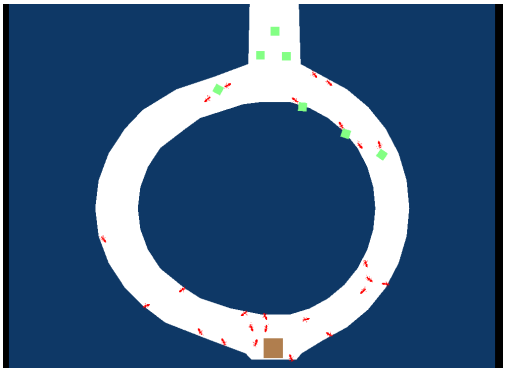


Figura 33 *– Teste 3: Os dois caminhos são testados*

Ao percorrer o caminho de maior distância, a formiga leva mais tempo para chegar ao alimento e voltar ao ninho. Dessa forma, a trilha de feromônio terá um potencial atrativo baixo, pois leva mais tempo para ser fortalecida(feedback positivo) e assim o feedback negativo acaba prevalecendo. No lado mais curto, as formiga levam menos tempo para executar suas tarefas de coleta de alimento, o que torna a trilha formada fortemente atrativa. As formigas que vem pelo caminho mais longo, ao voltar, são atraídas para o caminho mais curto. Em pouco tempo todas as formigas migram para o caminho mais curto como pode ser visto na Figura 34, isso prova o sucesso do algoritmo.

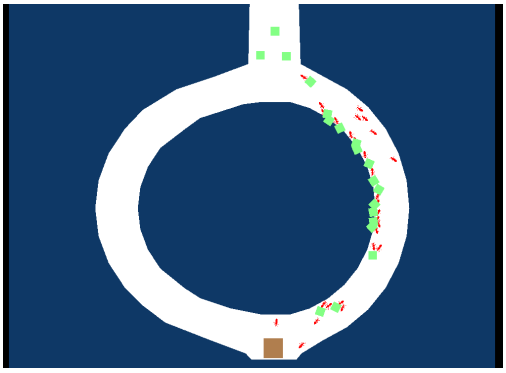


Figura 34 *– Teste 3: melhor caminho*

5.3 Testes com o simulador 2 (Feromônio de tinta)

No primeiro momento as formigas se espalham pelo labirinto deixando pequenos pontos de tinta (Figura 35). Com passar do tempo, todas as áreas do labirinto foram visitadas pelas formigas (Figura 36). É possível notar a diferença nas cores do feromônio. Quanto mais próximo do ninho, as cores do feromônio ficam mais próximas do azul. Essa diferença proporciona o destaque das rotas mais curtas.

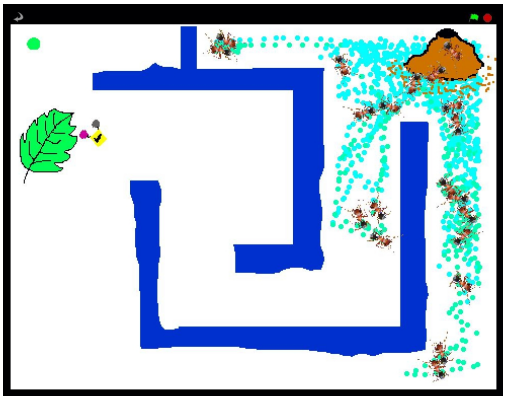


Figura 35 *– formigas forrageando*

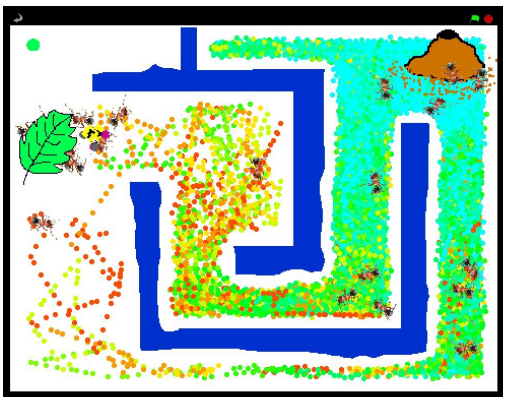


Figura 36 *– formação de duas trilhas*

Ao detectar as trilhas de feromônio os robôs começam a sair (Figura 37). Existem dois caminhos para se chegar ao ninho, mas um dos caminhos possui cores de maior valor é no caso, esse caminho será o mais curto. Os robôs logo entram no caminho mais curto.

Quanto mais longe do ninho, mais colorida fica a trilha devido a tinta não evaporar. Essa questão torna difícil para que os robôs acharem a trilha certa, mas com o tempo eles conseguem. Ao final, todos os robôs chegam ao ninho percorrendo o caminho mais curto (Figura 38).

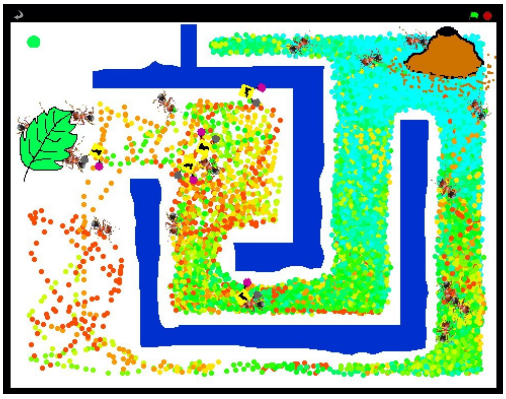


Figura 37 *– robôs começam a seguir os feromônios de cor*

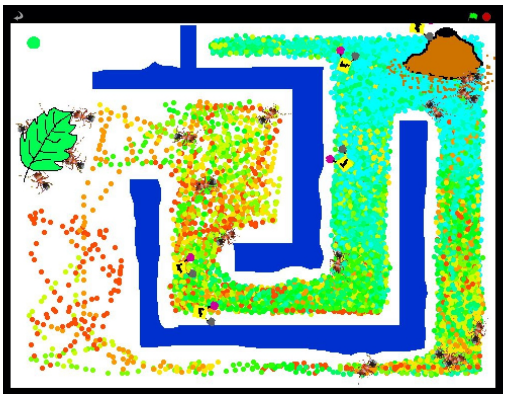


Figura 38 *– O caminho para ninho é encontrado*

5.4 Testes com o simulador 3 (Robô Virtual)

No início, o robô percorre o ambiente real, andando aleatoriamente. A Fomiga no mundo virtual recebe as coordenadas, através da comunicação Serial na porta COM4, do robô real e também se desloca deixando uma trilha de feromônio (Figura 39).

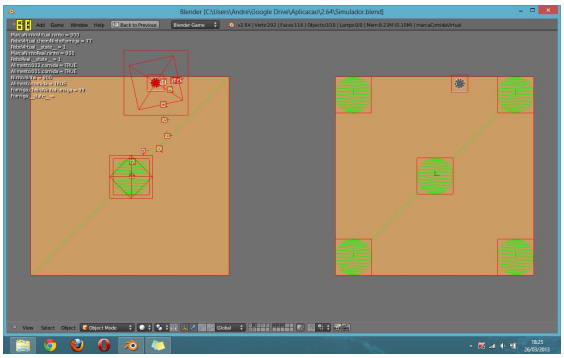


Figura 39 *– Inicio da simulação*

Quando o robô detecta o marcador do alimento é enviada um mensagem para a formiga virtual que imediatamente marca a posição no mundo virtual (Figura 40).

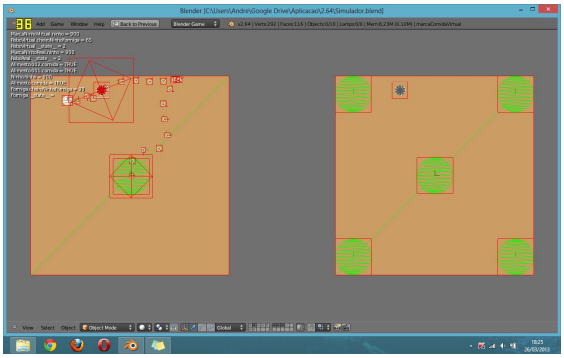


Figura 40 *– Marcação do alimento*

Tendo a posição do alimento, a formiga virtual passa à assumir o controle. A cada ida e volta, a formiga cria uma nova trilha que representa a melhor solução. O robô sendo guiado pela formiga através da comunicação Serial na porta COM4, percorre esse caminho no ambiente real (Figura 41).

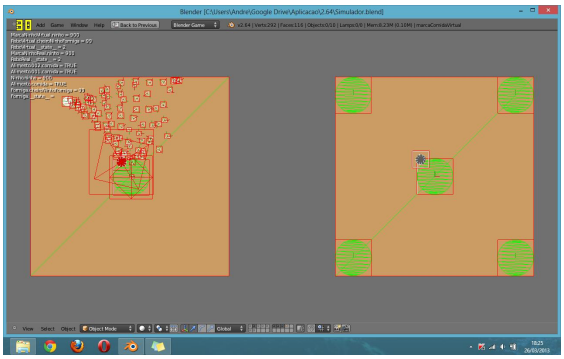


Figura 41 *– Várias trilhas*

Em pouco tempo, devido a evaporação do feromônio, a formiga e o robô estão percorrendo o melhor caminho (Figura 42).

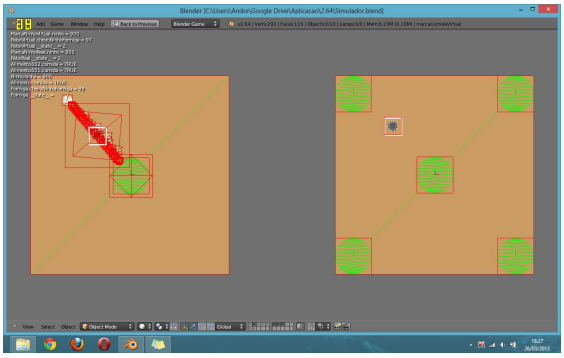


Figura 42 *– melhor caminho*

6 Capítulo

6.1 Conclusão

Para ter um entendimento melhor do algoritmo das formigas foi desenvolvido um simulador de colônia de formiga baseado nos trabalhos de DENEUBOURG et al (1990) e DORIGO & GAMBARDELLA (1997). O Ants3D, feito no Blender 2.49, torno possível criar um algoritmo que executa o comportamento básico, de uma formiga em seu trabalho diário de coleta de alimento. Esse algoritmo apresenta uma logica simples e de fácil adaptação para ser empregado em robôs.

Os experimentos realizados provarão que o algoritmo fora desenvolvido da forma correta, pois em todos os cenários as formigas encontraram o melhor caminho.

Tendo um modelo de algoritmo, o próximo passo era criar uma forma de comunicação para robô que possuísse a mesma eficiência da comunicação por feromônio das formigas. Esse objetivo era de fundamental importância, pois os algoritmos inspirados em colônias de formiga tem sua base nesse mecanismo de comunicação.

A primeira ideia era usar tinta de cores diferentes para representar o feromônio, onde cada cor significaria um nível de intensidade atrativa. Para testar a eficácia dessa hipótese foi construído em simulador no Scratch 1.4.

No simulador o sistema se mostra funcional, os robôs conseguem seguir as trilhas coloridas e achar o melhor caminho, mas no decorrer dos estudo fica evidente que a aplicação real do sistema não seria viável, pois os pingos de tinta quando sobrepostos se misturam gerando novas cores. Não haveria como calibrar o sensor dos robôs para a infinidade de padrões de cores que serão gerados, e também causaria uma grande sujeira um enxame de robôs soltando tinta para todo lado.

Uma forma de tornar esse sistema funcional seria substituir a tinta por alguma substancia que evaporasse ou que apresentasse alguma mudança decremental com o tempo e que fosse capaz de ser medida por algum sensor da robótica.

A segunda ideia consiste em usar um sistema formado por robôs reais controlados pelas formigas artificiais do simulador Ants3D. As formigas ficam com a parte dos feromônios e os robôs vão explorar o ambiente real onde estão imersos. A chave para o funcionamento desse sistema e o sincronismo entre os dois meios: real e virtual. Visando chegar a uma forma de comunicação, foi construído um simulador baseado em um protótipo de robô para esse sistema.

Com o estudo do simulador foi observado que é possível empregar esse sistema, mas é necessário melhoras alguns pontos. Deve-se criar um protocolo de comunicação seguro, pois qualquer falha de comunicação entre os meio (real e virtual) causa uma perda de sincronismo.

Outros pontos só podem ser avaliados com testes reais, mas devido à exigência de sincronismo, pode se prever que o "Sistema de posicionamento por Mouse" não e o suficiente para esse sistema; é necessário somar outros sensores ao robô para assegurar o sincronismo.

O ponto forte desse sistema é o baixo custo e rápida construção dos robôs, e a possibilidade de usar robôs virtuais e reais ao mesmo tempo.

O algoritmo das formigas se mostrou muito viável para ser usados em robôs, pois com códigos simples é possível criar sistema de exploração complexos.

6.2 Trabalhos futuros

Para trabalhos futuros, alguns tópicos devem ser levados em consideração:

- Implementa o sistema de feromônio artificiais em enxames de robôs reais;
- Desenvolvimento de um protocolo de comunicação entre o robô e ambiente virtual;
- Desenvolvimento de um algoritmo de mapeamento tendo como base o sensor ultrassônico;
- Desenvolvimento de algoritmo para desviar de obstáculos;
- Reforça o sistema de localização do robô com outros sensor. Ex: GPS, acelerômetros, giroscópios ...

Referências

ALECRIM, Emerson. *Mouses: funcionamento, tipos e principais características*. 2008. Online, Acesso em 23 mar 2013. Disponível em: <http://www.infowester.com/mouse.php>.

ALMEIDA, T.A. et al. Algoritmo de colônia de formigas e sistema imunológico artificial aplicado ao problema de designação generalizada. 2007. Online, Acessado em 17 mar. 2013. Disponível em: <http://www.dt.fee.unicamp.br/~tiago/courses/computacao_natural/IA013.htm>.

ANGELO, Fernanda. *Mouse completa 43 anos em 2007; conheça história*. 2007. Online, Acesso em 23 mar 2013. Disponível em: <http://tecnologia.uol.com.br/produtos/ultnot/2007/05/21/ult2880u361.jhtm>.

ARRUDA, Felipe. *Como inventaram o mouse*. 2011. Online; Acesso em: 23 mar. 2013. Disponível em: <http://www.tecmundo.com.br/historia/10976-como-inventaram-o-mouse.htm>.

BALAGE, Pedro P. *Aplicações da Inteligência Artificial*. 2008. Online, Acessado em: 10 out. 2012. Disponível em: <http://investiga-ia.blogspot.com.br/2008/10/aplicaes-da-inteligncia-artificial.html>.

BELO, Felipe Augusto Weilemann. *Desenvolvimento De Algoritmos De Exploração E Mapeamento Visual Para Robôs Móveis De Baixo Custo*. Dissertação (Mestrado) — Pontifícia Universidade Católica Do Rio De Janeiro - Puc-Rio, 2006. Online, Acessado em: 12 out. 2012. Disponível em: <http://www.maxwell.lambda.ele.puc-rio.br/9142/91422.PDF>.

BEVILACQUA, R. et al. Estratégias adaptativas em sociedades de formigas. *SBA Controle e Automação*, v. 10, p. 3, 1999. Online, 16 mar. 2013. Disponível em: <http://www.fee.unicamp.br/revista_sba/vol10/V10A232.pdf>.

BEZERRA, David. *A história do Mouse*. 2010. Online, Acesso em 23 mar. 2013. Disponível em: <http://www.iotecnologia.com.br/a-histria-do-mouse>.

BONABEAU, Eric. *Definitions of stigmergy*. 1999. Online, Acessado em 16 mar. 2013. Disponível em: <http://www.stigmergicsystems.com/stig_v1/stigrefs/article1.html?900388>.

BONABEAU, Eric; DORIGO, Marco; THERAULAZ, Guy. Swarm intelligence : from natural to artificial intelligence. 1999.

BOULET, Frédérique. *Computação natural: A filosofia*. 2010. Online, Acessado em 17 mar. 2013. Disponível em: <http://www.xuejava.com/computacao-natural-a-filosofia.html>.

CARVALHO, A. C. P. de L. F. de. Computação bioinspirada. *Revista Eletronica de Ciencias*, n. 22, 2003. Online, Acessado em 08 mar. 2013. Disponível em: <http://www.cdcc.usp.br/ciencia/artigos/art_22/computacaobioinspirada.html>.

CASTRO, L.N. De; ZUBEN, F.J. von. *Recent Developments In Biologically Inspired Computing*. Idea Group Publishing, 2005. Online, Acessado em 17 mar. 2013. ISBN 9781591403128. Disponível em: <http://books.google.com.br/books/about/Recent_Developments_In_Biologically_Insp.html?id=s_Q5YZ2nh2kC&redir_esc=y>.

CASTRO, L.N. de. *Fundamentals of Natural Computing: Basic Concepts, Algorithms, And Applications*. Chapman & Hall/CRC, 2006. (Chapman and Hall/CRC Computer and Information Science Series). ISBN 9781584886433. Disponível em: <http://books.google.com.br/books?id=N6iYpNVP9RgC>.

CASTRO, L. N. Fundamentos da computação natural: uma visão geral. *Fisica da Life Comentários 4*, p. 1–36p, 2007. Online, Acessado em 17 mar. 2013. Disponível em: <http://www.uvm.edu/~cmplxsys/tri/pdf/naturalcomputing.pdf>.

CASTRO, Leandro N.; ZUBEN, Fernando J. Von. *Tópicos em Sistemas Inteligentes II- Tópico 4:Inteligência Coletiva*. DCA/FEEC/Unicamp: [s.n.], 2003. 34 p. Online, Acessado em 06 out. 2012. Disponível em: <http://www.dca.fee.unicamp.br/~vonzuben/courses/ia006.html>.

COELHO, L. dos S.; NETO, R. F. T. Colônia de formigas: Uma abordagem promissora para aplicações de atribuicao quadratica e projeto de layout. *XXIV ENEGEP*, Florianópolis, SC, Brasil, 2004. Online, 16 mar. 2013. Disponível em: <http://www.dep.ufscar.br/admin/upload//ARTIGO_1156534584.PDF>.

COSTA, Danilo Nogueira. *Simulador extensível para navegação de agentes baseado em inteligência de enxames*. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação, 2007. Online, Acessado em: 19 out. 2012. Disponível em: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-16062007-095214/pt-br.php>.

COSTA, Danilo Nogueira. *Simulador extensível para navegação de agentes baseado em inteligência de enxames*. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação - ICMC-USP, 2007. Online, Acessado em. Disponível em: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-16062007-095214/pt-br.php>.

DENEUBOURG, J. L. et al. The self-organizing exploratory pattern of the argentine ant. *Journal of insect behavior*, v. 3, n. 2, 1990. Online, Acessado em 18 mar. 2013. Disponível em: <http://neuro.bstu.by/ai/To-dom/My_research/Courses/2009-semester-project/ACO/A/Ref/76.pdf>.

DORIGO, Marco; GAMBARDELLA, Luca Maria. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, v. 1, n. 1, 1997. Online, Acessado em 18 mar. 2013. Disponível em: <www.idsia.ch/~luca/acs-ec97.pdf>.

DORIGO, Marco et al. The ant system:optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics?Part B*, v. 26, n. 1, 1996. Online, Acessado em 18 mar. 2013. Disponível em: <ftp://iridia.ulb.ac.be/pub/mdorigo/journals/IJ.10-SMC96.pdf>.

DORIGO, Marco; STÜTZLE, Thomas. *Ant Colony Optimization*. United States of America: MIT Press, 2004.

ENSINAREVT. *A Teoria da Cor*. 2012. Online, Acesso em 23 mar 2013. Disponível em: <http://ensinarevt.com/conteudos/teoria_cor/>.

FOUNDATION, Blender. *Source Code and Features*. 2012. Online; Acesso em: 25 mar. 2013. Disponível em: <http://www.blender.org>.

GONCALVES, Luiz M. Garcia. *Robótica, principais tendencias e direções.* 2005. Online, Acessado em: 10 out. 2012. Disponível em: <http://www.comciencia.br/reportagens/2005/10/09.shtml>.

HAMANN, Renan. *Como foi realizado o pouso da sonda Curiosity em Marte?* 2012. Online, Acessado em 16 mar. 2013. Disponível em: <http://megacurioso.com.br/exploracao-espacial/27918-como-foi-realizado-o-pouso-da-sonda-curiosity-em-marte-.htm>.

INOVACAOTECNOLOGICA. *Robótica de enxame: robôs unidos jamais serão vencidos*. 2012. Disponível em: <http://www.inovacaotecnologica.com.br/noticias/noticia.php?artigo=robotica-de-enxame>.

KIOSKEA. *O Mouse*. 2013. Online, Acesso em 23 mar. 2013. Disponível em: <http://pt.kioskea.net/contents/pc/souris.php3>.

LEVY, Pierre. *A Inteligencia Coletiva por uma antropologia do ciberespaço.* São Paulo, Brasil: Loyola, 1998.

LOPES, H. S; MOLLE, V. D.; LIMA, C. R. E. Aplicação da otimização por colonia de formigas ao problema de roteamento de multiplos veiculos de capacidade limitada. *Anais do I Simpósio Brasileiro de Inteligência Computacional.*, Florianópolis, 2007. Online, Acessado em 16 mar. 2013. Disponível em: <http://fei.edu.br/sbai/SBAI2007/docs%5C60100051.pdf>.

MELO, Bruno M. de Souza. *Swarm Intelligence.* 2009. Online, Acessado em 20 set. 2012. Disponível em: <http://oglobo.globo.com/blogs/mundointeligente/posts/2009/06/16/swarm-intelligence-196033.asp>.

MIAZAKI, Mauro. *Sistema de controle multi-robô baseado em colônia de formigas artificiais*. Dissertação (Mestrado) — Instituto de ciências matemática e de computação ICMC-USP, São Carlos, 2007.

MILLER, Peter. *Teoria dos enxames: formigas, abelhas e aves nos ensinam a lidar com a complexidade do mundo*. 2007. Online, Acessado em 11 fev. 2013. Disponível em: <http://viajeaqui.abril.com.br/materias/teoria-dos-enxames>.

MOLINA, L et al. Estimação de posição e velocidade de robôs móveis com restrições holonômicas utilizando. *IX Simpósio Brasileiro de Automação Inteligente*, p. 1–6, 2009. Acesso em 24 mar. 2013. Disponível em: <http://www.gprufs.org/genius/publicacoes/arquivos/138/WO9tpJYG.pdf>.

MOURA, Leonel; RAMOS, Vitorino. *Caminhos, formigas e anarquia/ Parte 2*. aLife Art Architecture: [s.n.], 2001. Online, 16 mar. 2013. Disponível em: <http://www.lxxl.pt/aswarm/caminhos2.html>.

NETO, Roberto Fernandes Tavares; COELHO, Leandro dos Santos. Planejamento de rotas para robôs de inspeção usando um algoritmo híbrido de colônia de formigas e algoritmo cultural. Programa de Pós-Graduação em Engenharia de Produção e Sistemas, Laboratório de Automação e Sistemas, 2006. Online, Acessado em 18 mar. 2013. Disponível em: <www.dep.ufscar.br/admin/upload//ARTIGO_1156534634.PDF>.

NEUFERT, Diego Rodrigo. *Desenvolvimento de um Robô que Percorre o Menor Caminho Aplicando o Algoritimo de Otimização por Colônia de Formigas*. [s.n.], 2006. Online, Acessado em 20/04/2013. Disponível em: <http://www.leandrohsouza.com.br/engcomp/attachments/article/93/D743Z1~4.PDF>.

RUSSELL, Stuart J.; NORVING, Peter. *Inteligência artificial:tradução da segunda edição*. [S.l.]: Elsevier, 2004. ISBN 8535211772.

SCRATCH. *Create and share your own interactive stories, games, music, and art*. 2013. Online, Acesso em 23 mar 2013. Disponível em: <http://scratch.mit.edu/>.

SERAPIAO, Adriane Beatriz de Souza. Fundamentos de otimização por inteligência de enxames: uma visão geral. *Sba Controle & Automação*, v. 20, n. 3, p. 271–304, 2009. ISSN 0103-1759. Online, Acessado em 16 mar. 2013. Disponível em: <http://www.scielo.br/scielo.php?pid=S0103-17592009000300002&script=sci_arttext>.

SIERAKOWSKI, Cezar A. *Inteligência Coletiva Aplicada a Problemas de Robótica Movel*. 160–160 p. Dissertação (Mestrado) — Instituto de Engenharia de Produção e Sistemas, Universidade Católica do Paraná, 2006.

SILVA, E. O. A. et al. Uma abordagem paralela baseada em colonia de formigas para o problema do caixeiro viajante. *Cadernos do IME : Serie Informática*, v. 18, 2005. Online, Acessado em 16 mar. 2013. Disponível em: <http://www.ime.uerj.br/cadernos/cadinf/vol18/Artigo3.pdf>.

SOCIETYOFROBOTS. *COLOR SENSORS TUTORIAL*. 2012. Online, Acesso em 23 mar. 2013. Disponível em: <http://www.societyofrobots.com/sensors_color.shtml>.

VITALI, Mateus. *Sensor de Cor*. 2011. Online, Acesso em 24 mar. 2013. Disponível em: <http://mateusvitali.wordpress.com/2011/06/16/sensor-de-cor/>.

WALL, Mike. *Touchdown! Huge NASA Rover Lands on Mars*. 2012. Online, 16 mar. 2013. Disponível em: <http://www.space.com/16932-mars-rover-curiosity-landing-success.html>.

XAVIER, Rafael Silveira. *O QUE É A COMPUTAÇÃO NATURAL?* 2013. Online, Acessado em: 18 mar. 2013. Disponível em: <http://www.computacaonatural.com.br/?page_id=220>.

APÊNDICE A – Sistema de posicionamento por Mouse

O Mouse é capaz de determinar as coordenadas x e y do movimento realizado pela mão tornando possível manipular objetos no mundo virtual do computador através de movimentos realizados no mundo real. Essa característica foi de grande interesse para esse trabalho devido à necessidade de estabelecer o sincronismo entre o simulador virtual e o robô real.

A.1 Ligação no Arduino

Como o possível robô para esse trabalho foi projetado na plataforma Arduino Uno, as ligações e códigos foram testas apenas no Arduino Uno.

A maioria dos Mouses apresenta quatro conexões principais, são elas: 5 volts, negativo(ground), Data e Clock. O Mouse não deve receber mais de 275 mA. O Arduino apresenta uma saída de 5 volts e 50mA por tanto, não a problema em ligar o Mouse diretamente nele. Os pinos Data e Clock devem ser ligados aos pinos digitais do Arduino.

A.2 Algoritmo

A comunicação com o Mouse deve seguir um protocolo que tem a função de traduzir os dados gerados pela conexão "Data" bem como o envio do sinal para conexão "Clock". A descrição do protocolo pode ser vista aqui: `http://www.computer-engineering.org/`.

No site `http://playground.arduino.cc/` que, contem códigos e tutorias sobre o Arduino, apresenta o código e esquema para utilizar o Mouse com o Arduino. Esse código funcionou em todos os modelos de Mouse de Esfera testado e em alguns modelos de Mouse ópticos. O código retorna o estado dos botões do Mouse e também a variação das coordenadas x e y.

A.3 Montagem no Robô

O esquema da montagem do mouse no robô pode ser vista na Figura 43. Essa forma de posicionar o mouse no robô torna possível medir o deslocamento (eixo Y) e a rotação (eixo X).

Os testes realizados foram feitos da seguinte maneira:

1. O robô foi colocado para se deslocar em linha reta para frete e para traz. Nesse teste observou-se que a coordena Y apresenta variações e a coordenada X fica estática.

2. O robô foi colocado para realizar rotações no sentido horário e anti-horário permanecendo no mesmo eixo. Esse movimento e feito colocando os motores do robô para girarem em sentidos diferentes. Nesse teste observou-se que a coordenada X variava e a coordenada Y permanece estática.

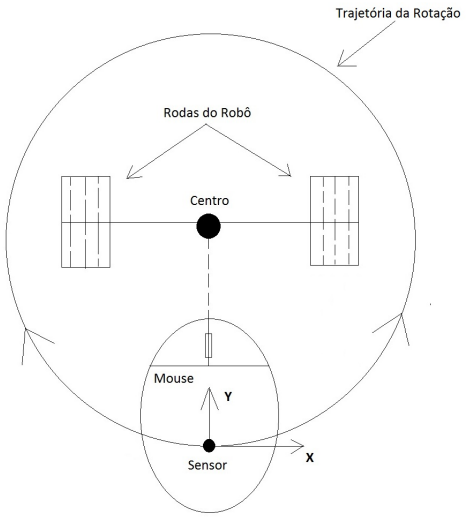


***Figura 43** – Posicionamento do Mouse no robô*

Esses movimentos já satisfazem as necessidades desse projeto por tanto, não foram feito teste com outros tipos de movimento como o "Movimento em Arco" onde, um dos motores do robô fica parado enquanto o outro gira.

A.4 Conclusão

A vantagem de usar o Mouse como mensurador de deslocamento e rotação esta na facilidade de adquirir esse sensor, no preço se comparado com outros sistemas como, GPS e facilidade de uso. Como uma desvantagem, o uso do Mouse é mais adequado para trajetos curtos pois, no caso de algum erro de leitura esse erro será acumulado ao longo da trajetória do robô. Portanto, para robôs que precisam realizar trajetos longos e ter uma medida muito precisa desse deslocamento e necessário acrescentar mais sensores para reforça esse sistema.

APÊNDICE B – Algoritmos Criados

B.1 Principais Algoritmos do Ants3D

```
'''
script do projeto Ants3D implentado no Blender 2.49
Tem a funcao de fazer a formiga seguir a melhor trilha de feromonio

Autor: Andre Lima
email: andre.lima0000@gmail.com
'''
import GameLogic as g

cont=g.getCurrentController()
obj=cont.owner

nariz = cont.sensors['nariz']
track = cont.actuators['track']

if obj['vida'] > obj['ciclos']:
  obj.endObject()
else:
  obj['vida'] += 1

def maiorCheiro():
  maior = 0
  for ob in nariz.hitObjectList:
    if ob.has_key(obj['segue']):
      if ob[obj['segue']] > maior:
        maior = ob[obj['segue']]
        track.object = ob

cont.deactivate('track')
cont.deactivate('move')
maiorCheiro()
if track.object != None:
  cont.activate('track')
  cont.activate('move')
```

segueCheiro.py

```
'''
script do projeto Ants3D implentado no Blender 2.49
Tem a funcao de fazer a formiga se mover em direcoes aleatorias

Autor: Andre Lima
email: andre.lima0000@gmail.com
'''
import GameLogic as g
import random as r

cont = g.getCurrentController()
obj = cont.owner

if obj['vida'] > 5000:
  obj.endObject()
else:
  obj['vida'] += 1

if obj['tempo'] > 0.5:
  lado = r.randint(1,10)
  giro = r.uniform(0.0,1.0)
  if lado <= 4:
    giro = -giro # giro negativo
  elif lado == 5:
    giro = obj['giro_ante'] # giro anterior
  obj.applyRotation([0.0, 0.0, giro ])
  obj['giro_ante'] = giro
  obj['tempo'] = 0.0

cont.activate('move')
```

girar.py

```
'''
script do projeto Ants3D implentado no Blender 2.49
faz a formiga libera o feromonio que indica o caminho para ninho

Autor: Andre Lima
email: andre.lima0000@gmail.com
'''
import GameLogic as g

cont=g.getCurrentController()
```

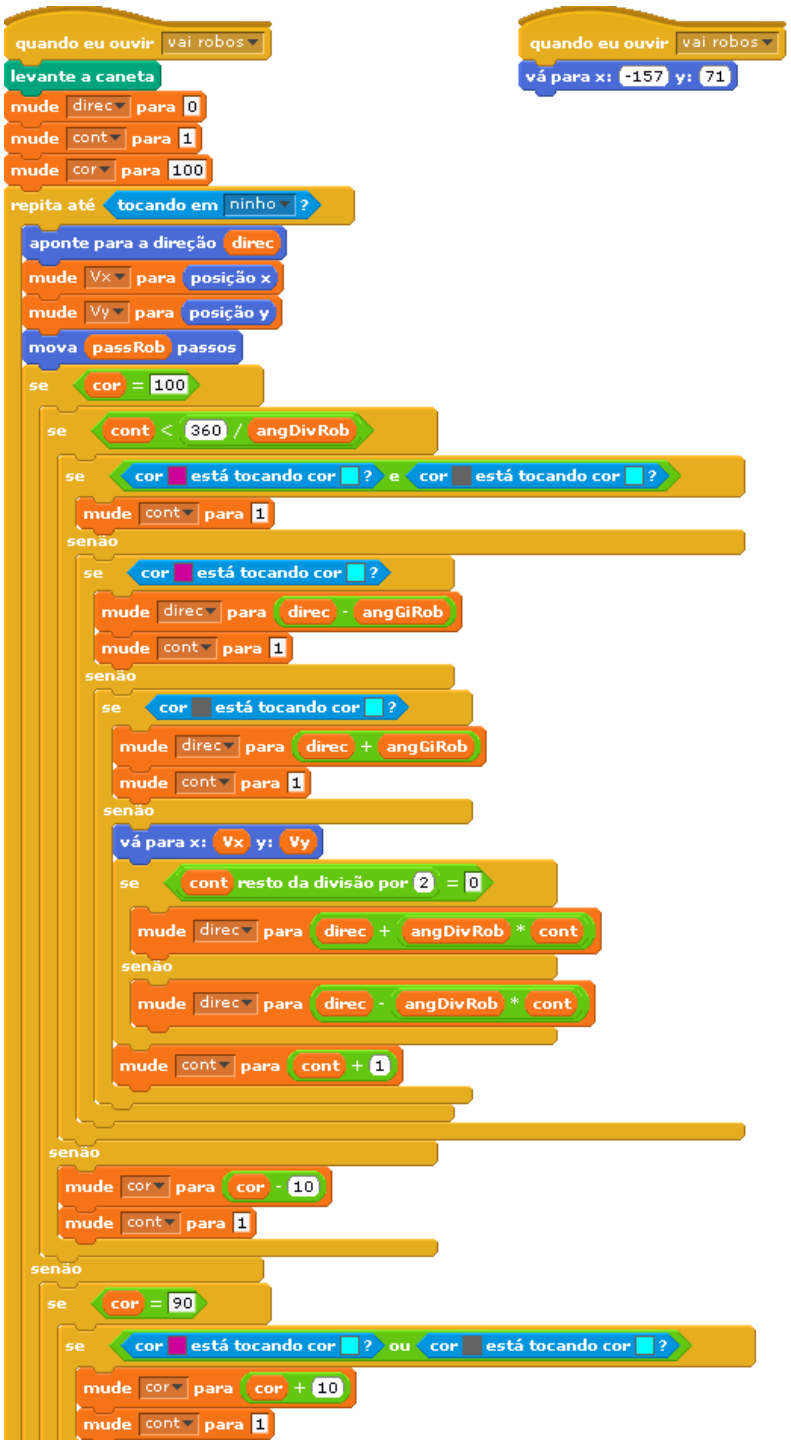
```
obj=cont.owner

cheiroNinho = cont.actuators['liberaCheiroNinho']

if (obj['cheiroNinhoFormiga'] > 0) and (obj['segue'] == 'cheiroComida'):
  cheiroNinho.object['cheiroNinho'] = obj['cheiroNinhoFormiga']
  obj['cheiroNinhoFormiga'] -= 2
  cont.activate('liberaCheiroNinho')
```

liberaCheiro.py

```
'''
script do projeto Ants3D implentado no Blender 2.49
Atua no ninho e tem a funcao de reforca o feromonio da formiga

Autor: Andre Lima
email: andre.lima0000@gmail.com
'''
import GameLogic as g

cont = g.getCurrentController()
obj = cont.owner

temFormiga = cont.sensors['temFormiga']

for ob in temFormiga.hitObjectList:
  if ob.has_key('vida'):
    ob['cheiroNinhoFormiga'] = (obj['cheiroNinho'] - 1)
  elif ob.has_key('comida'):
    obj['ninho'] += 5
    ob.endObject()
```

perfumaNinho.py

B.2 Parte do Algoritmo do Simulador 2 referente ao Robô



B.3 Principais Algoritmos do Simulador 3

```
'''
script do projeto Simulador 3 - robo virtual implementado no Blender 2.54
faz a formiga virtual interpretar os comandos enviadas
pela porta Serial COM4 e executar o movimento

Autor: Andre Lima
email: andre.lima0000@gmail.com
'''
from bge import logic as g
import site
import serial
import time as t

def envmsn(ser,aux):
    ser.write(aux.encode('ascii'))

def recmsn(ser):
    aux=ser.readline()
    return bytes.decode(aux) # chr(aux[0]) coloca na forma normal


def main(cont):
  obj = cont.owner # obj recebe o objeto do controlador (no caso o cubo)

  if(g.globalDict.get("portaV")==None):
    ser = serial.Serial(3,9600,timeout=0.01,writeTimeout=0.01) # porta 0 =
        1 no windows
    t.sleep(0.5)
    ser.write('a'.encode('ascii')) # so para iniciar a comunicacao
    t.sleep(0.1)
    g.globalDict["portaV"] = ser
  else:
    ser = g.globalDict["portaV"]

  aux = recmsn(ser)
  print(aux)
  if(aux[0:1] == "c"):
    cont.activate("comidaVirtual")
  if(aux[0:1] == "g"):
    giro = float(aux[1:len(aux)-2])
    obj.applyRotation([0.0, 0.0, giro])
  if(aux == "w"):
```

```
42      cont.activate("move")
43    else:
44      cont.deactivate("move")
```

girarVirtual.py

```
1  '''
2  script do projeto Simulador 3 - robo virtual implementado no Blender 2.54
3  faz a representacao do robo real interpretar as coordenadas enviadas
4  pela porta Serial COM4 e executar o movimento
5
6  Autor: Andre Lima
7  email: andre.lima0000@gmail.com
8  '''
9  from bge import logic as g
10 import site
11 import serial
12 import time as t
13
14 def envmsn(ser,aux):
15     ser.write(aux.encode('ascii'))
16
17 def recmsn(ser):
18     aux=ser.readline()
19     return bytes.decode(aux) # chr(aux[0]) coloca na forma normal
20
21 def fim(aux,i,c):
22   tam = len(aux)
23   while(i<tam):
24     if(aux[i:i+1]==c):
25       return i
26     i=i+1
27   return 0
28
29 #o segundo indice e tambem uma posicao
30 def converte(aux):
31   i = 1
32   f = fim(aux,i,"y")
33   #print(aux[i:f])
34   x1 = float(aux[i:f])
35
36   i = f+1
37   f = fim(aux,i,"X")
38   #print(aux[i:f])
```

```
39    y1 = float(aux[i:f])
40
41    i = f+1
42    f = fim(aux,i,"Y")
43    #print(aux[i:f])
44    x2 = float(aux[i:f])
45
46    i = f+1
47    f = fim(aux,i,"f")
48    #print(aux[i:f])
49    y2 = float(aux[i:f])
50
51    return ((x1, y1, 0.0),(x2, y2, 0.0),(0.0, 0.0, 1.0))
52
53  def main(cont):
54    obj = cont.owner # obj recebe o objeto do controlador (no caso o cubo)
55
56    if(g.globalDict.get("portaR")==None):
57      ser = serial.Serial(3,9600,timeout=0.01,writeTimeout=0.01) # porta 0 =
          1 no windows
58      t.sleep(0.5)
59      ser.write('a'.encode('ascii')) # so para iniciar a comunicacao
60      t.sleep(0.1)
61      g.globalDict["portaR"] = ser
62    else:
63      ser = g.globalDict["portaR"]
64
65    aux = recmsn(ser)
66    print(aux)
67    if(aux[0:1]=="x"):
68      obj.orientation = converte(aux)
69      if(aux[-1]=="w"):
70        cont.activate("move")
71      else:
72        cont.deactivate("move")
```

segueCheiroReal.py