UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MARIANA AKEMI SHIMABUKURO

SIMULADOR E BIBLIOTECA EM *PROCESSING* PARA O AEDROMO

BAURU - SP

2015

MARIANA AKEMI SHIMABUKURO

SIMULADOR E BIBLIOTECA EM *PROCESSING* PARA O AEDROMO

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Faculdade de Ciências, campus Bauru.

Orientador: Renê Pegoraro

BAURU
2015

MARIANA AKEMI SHIMABUKURO

SIMULADOR E BIBLIOTECA EM *PROCESSING* PARA O AEDROMO

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, campus Bauru.
Orientador: Renê Pegoraro

BANCA EXAMINADORA

Profº Dr. Rene Pegoraro
UNESP – Bauru
Orientador

Profº Dr. Wilson Yonezawa
UNESP – Bauru

Profº Dr. Marco Antônio Rahal Sacoman
UNESP - Bauru

BAURU
2015

Dedico este trabalho à minha família, que sempre me deu todos os tipos de suporte e nunca me deixou desistir de meus sonhos e objetivos.

AGRADECIMENTOS

Agradeço à minha família e amigos pelo apoio, principalmente aos meus pais que sempre estiveram presentes em todas as etapas e decisões da minha vida, quem eu sou hoje, todas as minhas realizações e conquistas eu devo a eles.

Agradeço também a entes queridos que hoje não estão mais presentes entre nós, mas sempre serviram de inspiração como meus avôs Kisei e João.

Agradeço em especial algumas pessoas mais próximas que sempre me apoiaram e ajudaram como puderam para a conclusão de mais esta etapa entre eles estão: Azael Sousa, Celso Lisboa, Jéssica Lopes e Thiago Tobaro. Não posso deixar de agradecer também àquela pessoa que me apoiou nos dias mais difíceis durante essa jornada, quem mais me ajuda a renovar as energias após tanto desgaste com seu amor, compreensão e apoio, por isso meu muito obrigada Victor Sawal Jr.

Agradeço também a todos os professores responsáveis pela minha formação, ou melhor, aos meus mestres que sempre me guiaram durante minha trajetória escolar e acadêmica, em especial ao meu querido orientador Rene Pegoraro que sempre foi um grande tutor e uma pessoa cujo caráter e feitos sempre me serviram de espelho.

“*Rise and rise again. Until lambs become lions.”*

Robin Hood the Movie, 2010

RESUMO

Este trabalho apresenta desde a fundação teórica até testes e discussões para a construção de um simulador de robôs móveis e uma biblioteca que encapsula rotinas para facilitar a programação de softwares de controle para os robôs móveis, permitindo que pessoas com pouca ou nenhuma habilidade em programação possa fazer um programa para controlar um robô. Serão expostos e discutidos aspectos do desenvolvimento do simulador e da biblioteca de suporte a programação dos robôs utilizando a linguagem de programação *Processing*. Neste trabalho também compara-se o comportamento do simulador com o ambiente real e a simplificação oferecida pela utilização da biblioteca na construção do software do aluno.

Palavras-chave: AEDROMO, robótica, simulação.

ABSTRACT

This work presents from the theoretical foundation to test and discussions in order to build a mobile robot simulator and a library that encapsulates routines to facilitate a control software for mobile robots programming, allowing people with little or no programming skills to do programs for controlling a robot. Will be presented and discussed aspects of the development of the simulator and the support library programming of robots using the Processing programming language. This work also compares the simulator's behavior with the real environment and the simplification offered by the use of the library in the construction of the student software.

Key-words: AEDROMO, robotics, simulation.

SUMÁRIO

1 Introdução

Assim como os computadores, a robótica é um recurso tecnológico auxiliar utilizável no processo educacional que pode contribuir para o desenvolvimento cognitivo do aluno e habilidades intelectuais específicas. Ela se oferece como uma ferramenta pedagógica interessante sobre vários aspectos e assim deve ser encarada e explorada (PAPERT, 1980).

Alves et al. (2011a) descreve a robótica como uma abordagem de sucesso no âmbito educacional, por ela ser intrinsecamente multidisciplinar, que estimula o trabalho em grupo e promove o retorno de forma visual motivadora.

No entanto, há dificuldades em disponibilizar um número considerável de robôs por número de alunos, além do próprio espaço físico requerido nas atividades. Nesse contexto o uso de simuladores é uma saída interessante, pois os alunos podem testar suas soluções em suas próprias máquinas, em seu próprio tempo e até mesmo fazê-lo em suas casas. E depois de ter entendido o funcionamento do robô e do ambiente, ele pode testar no ambiente real.

Este trabalho apresenta um simulador com visualização tridimensional para o Ambiente Educacional Didático de Robôs Móveis (AEDROMO) e uma biblioteca destinada ao usuário, destacando o uso da linguagem *Processing* no desenvolvimento destes *softwares*.

Uma versão nova do simulador do AEDROMO em *Processsing* será implementada para proporcionar mais flexibilidade ao ambiente como a configuração de quantos robôs e objetos inanimados serão dispostos para a manipulação pelo aluno.

E também a implementação de uma biblioteca em *Processing* para o AEDROMO para ajudar os alunos na fabricação de seus softwares de controle dos robôs móveis do ambiente.

1.1 Objetivos

O objetivo deste trabalho é desenvolver uma versão nova do simulador e a implementação de uma biblioteca em *Processing* para o AEDROMO com

visualização tridimensional e a implementação de uma biblioteca em Processing para o AEDROMO que simplifique a sua utilização por usuários com pouca experiência em programação..

1.2 Justificativa

O desenvolvimento deste trabalho é importante para dar mais flexibilidade ao AEDROMO, pois com a nova versão do simulador, mais objetos podem ser inseridos ao ambiente, criando mais possibilidades a criação de atividades a serem propostas aos alunos. Outro aspecto importante é que esse simulador em *Processing* tem a interface em gráficos tridimensionais, o que dá mais realidade a simulação.

Além disso, a existência de uma biblioteca para o ambiente facilita o desenvolvimento de aplicações pelos alunos, que não necessitam conhecer todas as especificações técnicas da arquitetura do AEDROMO para implementar suas soluções, ou seja, controlar os robôs móveis.

1.3 Estrutura da Monografia

Esta monografia está estruturada a partir deste em seis outros capítulos sendo eles:

Capítulo 2 - AEDROMO: a história, missão, composição, arquitetura e funcionamento do AEDROMO;

Capítulo 3 - Simulação de robôs: revisão bibliográfica sobre outros simuladores de robôs e tecnologias utilizadas para a construção dos mesmos;

Capítulo 4 - *Processing*: apresentação da linguagem de programação *Processing,* suas aplicações e características;

Capítulo 5 - Módulos de Software Desenvolvidos neste Trabalho: características de funcionamento e aplicação do simulador proposto e da biblioteca;

Capítulo 6 - Resultados e trabalhos futuros: discussão e análise dos resultados encontrados no trabalho desenvolvido, e apresentação de propostas de trabalhos futuros relacionados a este trabalho; e

Capítulo 7 - Conclusão: resumo dos resultados e das ideias apresentadas ao percorrer desta monografia.

2 AEDROMO

O Ambiente Experimental Didático com Robôs Móveis (AEDROMO) é um ambiente desenvolvido no Laboratório de Integração de Sistemas e Dispositivos Inteligentes (LISDI) do Departamento de Computação da Faculdade de Ciências da UNESP campus de Bauru onde robôs interagem para realizar tarefas simples. Os robôs neste ambiente são localizados globalmente e controlados remotamente por um computador. O AEDROMO vem sendo desenvolvido como um ambiente de teste para diversas disciplinas. Este ambiente robótico é adaptável para oferecer tarefas em diferentes níveis, integrando objetivos educacionais com suporte tecnológico. Um de seus objetivos é ser aplicável como apoio educacional para o desenvolvimento de ciências, conceitos de lógica e de programação nos níveis fundamental e médio de ensino (ALVES et al., 2011a).

2.1 Objetos e composição do ambiente

O AEDROMO é composto por robôs (como o apresentado na figura 1) e também por um objeto inanimado: bola ou cubo; o qual é utilizado na interação com os robôs por meio de atividades propostas, por exemplo, jogar uma partida de futebol. Uma atividade realizada com o robô tem por objetivo demonstrar a programação do robô e habituar os alunos ao controle dele. Os robôs móveis foram especialmente desenvolvidos pelo LISDI para serem utilizados neste ambiente, eles apresentam pequenas dimensões, com sistema de tração diferencial (*differential drive*) (ALVES et al., 2011a). A tração diferencial consiste em ter o acionamento de cada uma das rodas do robô por comandos independentes.

Figura 1: Robô segurado por um adulto. (ALVES et al., 2011a)

O AEDROMO basicamente é formado por dois robôs, um computador, uma câmera global, do tipo *webcam*, e um transmissor, conforme mostra a figura 2. A flexibilidade de adaptação, uso por várias disciplinas e ciclos de ensino, e o baixo custo são as premissas na concepção e desenvolvimento deste ambiente. Com o intuito de facilitar e incentivar o uso deste ambiente, as suas dimensões também foram pontos norteadores. Os experimentos, neste ambiente, podem ser motivados para fins de pesquisa, aprendizagem ou, meramente, entretenimento. (FERASOLI et al., 2006)

Figura 2: Ambiente do AEDROMO com: webcam; robôs; e área de trabalho. (ALVES et al., 2011a)

2.2 Arquitetura do AEDROMO

O AEDROMO utiliza a arquitetura cliente-servidor. (ALVES et al., 2010). Ou seja, no AEDROMO há dois tipos de computadores: o Computador do Servidor e o Computador do Aluno funcionando como cliente do ambiente. O Computador do Servidor é responsável pela implementação do sistema de visão computacional e pela comunicação com os robôs. Por outro lado, o Computador do Aluno é onde o aplicativo de controle (*Software* do Aluno) é desenvolvido. Este aplicativo é um cliente que recebe do servidor as posições cartesianas de todos os objetos presentes na área de trabalho e envia ao servidor os comandos de acionamento de cada roda do robô sendo controlado, para que o servidor encaminhe estes comandos ao robô via sinal de rádio.

A comunicação entre o cliente e o servidor é dada através de uma rede utilizando o protocolo UDP/IP (*User Datagram Protocol over Internet Protocol*),

permitindo que seja escrito um aplicativo de controle para cada robô em qualquer linguagem de programação que suporte "*socket's*". Adicionalmente, os aplicativos de controle podem ser executados no mesmo computador do servidor ou em diferentes máquinas. Neste computador, as imagens são capturadas pela câmera global e processadas. (ALVES et al., 2011a)

Para processar as imagens é utilizada a técnica de localização chamada Visão Global que usa as imagens adquiridas de uma câmera localizada paralela ao solo para identificar a posição e orientação do robô dadas por (x, y, θ) na qual theta (θ) é o ângulo da orientação do robô. Para facilitar o reconhecimento dos robôs, cada robô possui um marcador fidedigno único com duas cores diferentes. Este sistema é semelhante ao utilizado por competições de futebol de robôs. (ALVES; ROSÁRIO; FERASOLI, 2011)

O modelo cinemático dos robôs móveis de tração diferencial pode ser descrito conforme Eq. (1). Considerando *r* como o raio da roda, *R* como a distância entre as rodas, $\dot{\varphi}_r$ e $\dot{\varphi}_l$ como as velocidades angulares das rodas direita e esquerda, *v* e *ω* como as velocidades linear e angular do robô, e *θ* como a orientação do robô (ALVES; ROSÁRIO; FERASOLI, 2011). E também as equações das posições cartesianas e velocidades de rotação em (2). (ALVES et al., 2011b)

$$\begin{bmatrix} \dot{\varphi}_r \\ \dot{\varphi}_l \end{bmatrix} = \begin{bmatrix} \frac{1}{r} & \frac{R}{r} \\ \frac{1}{r} & -\frac{R}{r} \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{1}$$

$$\begin{aligned} \dot{x}_c &= v\cos(\theta) \\ \dot{y}_c &= v\sin(\theta) \\ \dot{\theta} &= \omega \end{aligned} \tag{2}$$

A programação das aplicações no AEDROMO é realizada em linguagens de alto nível exigindo conhecimentos específicos em linguagens de programação e características do AEDROMO. Assim qualquer simplificação que possa permitir uma utilização mais fácil deste ambiente, alcançando mais pessoas com menor conhecimento em programação, é bem vinda e tem sua importância para a ampliação de sua utilização.

Este capítulo apresentou o funcionamento do AEDROMO, tanto os objetos que o compõe, dois robôs móveis e um cubo e/ou bola; até sua arquitetura, conexão UDP e modelo cinemático do robô. Um simulador para tal ambiente deve possuir características parecidas, seguindo os mesmo modelos de funcionamento.

3 Simulação de robôs

Staranowicz e Mariottini (2011) fazem em seu trabalho uma comparação de diversos simuladores classificados como de código aberto e comerciais. Tais como: Player/Stage (PS), Gazebo, ROS Robot Operating System, Simbad, Carnegie Mellon Robot Navigation Toolkit (CARMEN), Unified System for Automation and Robot Simulation (USARSim), Microsoft Robotics Developer Studio (MRDS) e MissionLab. Após a análise decidem descrever mais profundamente as plataformas mais relevantes em suas opiniões, Player/Stage, Gazebo e Robot Operating System. Uma característica que as três têm em comum é a utilização de *sockets* TCP/IP e UDP na comunicação entre os ambientes e seus clientes. Os clientes podem ser escritos em várias linguagens, como por exemplo, Java, C/C++ e Python, para as quais os simuladores proveem bibliotecas para seus usuários. As simulações podem ser realizadas em gráfico tanto em 3D quanto em 2D.

Browning e Tryzelaar (2003) descrevem uma ferramenta de simulação de múltiplos robôs chamada ÜberSim, especificamente para simulação de jogos de futebol de robôs. No desenvolvimento da ÜberSim, optou-se por seguir fielmente a cinética e a dinâmica dos objetos para uma melhor variedade de robôs, desde a categoria de futebol de robôs *very-small size* até robôs humanoides. A primeira versão do ambiente de simulação já é de fiel a realidade e possui classes reconfiguráveis de robôs movidos por tração diferencial *(differential dri*ve) e robô*s omni-directional* de três rodas.

O principal componente utilizado no desenvolvimento do ÜberSim, é uma biblioteca de ferramentas que auxiliam na física da simulação chamada *Open Dynamics Engine*[1] (ODE), criada por Russell Smith. ODE oferece razoavelmente precisão dinâmica de corpos rígidos usando um integrador Euler, um número de tipos de juntas e rotinas para colisão. As colisões podem ser elásticas, ou se o contato contínuo for desejado, o ponto de contato pode sofrer forças de fricção, alterando a direção dos corpos.

Kitano et al. (1997) descreve o funcionamento do simulador de futebol de robôs para uma competição chamada Robot World Cup Initiative (RoboCup). Os

1http://www.q12.org/ode. Acesso em 21/04/2014.

clientes do simulador podem ser implementados em qualquer linguagem e se conectam por meio de *sockets* UDP/IP. Os clientes, cada um dos robôs em campo, são independentes e a única informação que recebem sobre seus companheiros em campo do simulador é a posição deles relativas aos objetos como bola e traves.

A interface gráfica é em 2D e a colisão dos objetos é identificada por meio da sobreposição de objetos no campo de futebol, uma vez detectada, para tratar a colisão os objetos são movidos até que não haja sobreposição e a velocidade é multiplicada por -0,1.

3.1 Primeira versão do Simulador do AEDROMO

O simulador do AEDROMO, também conhecido como ambiente virtual, e desenvolvido pelo GISDI, tem o intuito de melhorar a portabilidade e usabilidade do AEDROMO de modo que o mesmo possibilite que o *software* do aluno seja testado, sem que se faça necessária a presença do ambiente real (arena com os robôs físicos). O *software* (programa do aluno) apresenta o mesmo funcionamento em ambos ambientes. Sendo que o simulador se comunica com o cliente através de uma conexão UDP, pela qual fornece as coordenadas dos robôs e objeto no ambiente virtual para o computador cliente e recebe os pacotes com os comandos de acionamento de cada uma das rodas do robô.

A conexão estabelecida entre o ambiente virtual e o cliente tem a mesma interface, assim é exatamente igual à conexão entre o ambiente real e o cliente. Ou seja, para o *software* do cliente não há diferença alguma na semântica ou na sintaxe do código. A única alteração necessária é indicar o IP do computador que hospeda o ambiente virtual ao invés do IP da máquina que hospeda o ambiente real (figura 3), sendo que o simulador pode ser executado na mesma máquina do programa do aluno. O simulador é muito mais prático para aluno que tem uma instância do ambiente virtual em execução no seu próprio computador permitindo a verificação do funcionamento do seu código conforme sua vontade antes de testá-lo no ambiente real.

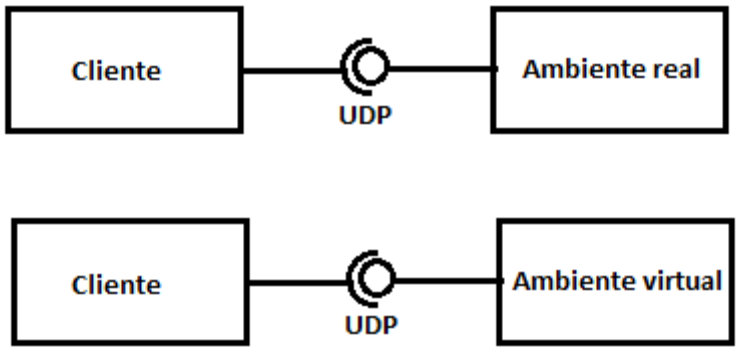

Figura 3: Conexão dos ambientes real e virtual

O ambiente virtual representa os mesmos elementos do ambiente real, e a interação de seus objetos é dada de forma muito parecida com a visão global da câmera presente na arena real, a figura 4 apresenta uma imagem da versão anterior (primeira versão) do simulador, implementado em Java, em uma simulação de uma partida de futebol de robôs no AEDROMO.

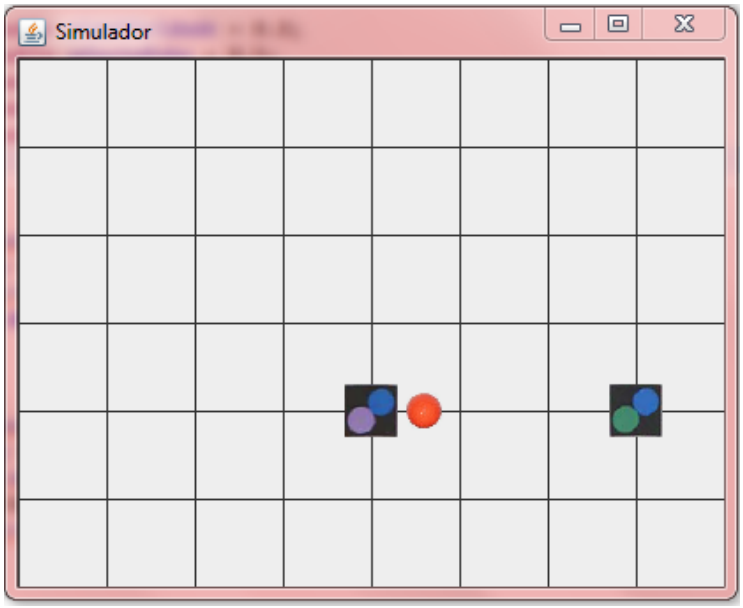

Figura 4: Primeira versão do Simulador do AEDROMO

Cada um dos objetos representados na figura 4 (robôs e a bola) são elementos independentes dentro da arena (*threads* no simulador) que executam concorrentemente, ou seja, cada um deles é responsável por seu próprio movimento e execução dos comandos recebidos do programa do cliente. E no caso da bola, sua

direção depende da interação com os robôs e o ambiente. A arena tem conhecimento de seus objetos e ações e ela cuida da colisão entre os mesmos.

A velocidade dos robôs é considerada constante com a sua cinética (discutida no item 1.1.2) e com aceleração instantânea, desconsidera-se a sua dinâmica pelo fato da massa dos robôs ser muito pequena e a aceleração ocorrer em um intervalo de tempo menor do que as imagens são capturadas pela câmera; isso é semelhante ao ambiente real onde a velocidade do robô real é pequena e também pode ser considerada constante.

Nesta versão do simulador, o método de colisão entre os objetos está implementado de forma simples, no qual é verificada a sobreposição das imagens, e feito o cálculo (direção e sentido da velocidade) de forma a afastar o robô da bola e a bola do robô, considerando o ângulo da colisão, lembrando que a velocidade (módulo da velocidade) é sempre considerada baixa e constante. A tela do simulador é atualizada a uma taxa de 30 fps (*frames per second*).

Finalmente, após comparar alguns dos simuladores existentes na literatura (no item 1.2) (Staranowicz e Mariottini (2011), Browning e Tryzelaar (2003) e Kitano *et a*l (1997)) pode-se notar algumas características em comum com o simulador do AEDROMO como a utilização de sockets UDP na comunicação entre os ambientes e seus clientes. A implementação descrita por Kitano et *a*l. (1997) a verificação da colisão é semelhante a do ambiente virtual do AEDROMO, ou seja, monitorando a sobreposição das imagens.

3.2 Processing

Processing é uma linguagem de programação, um ambiente de desenvolvimento e uma comunidade *online*. Desde 2001*, Processing* tem promovido a disseminação de *software* nas artes visuais dentro da tecnologia. Inicialmente criado para servir como um caderno de esboço de *softwa*re e ensinar os fundamentos de programação de computadores dentro de um contexto visual, *o Processing* evoluiu para uma ferramenta de desenvolvimento para os profissionais. Hoje, existem dezenas de milhares de estudantes, artistas, designers,

pesquisadores e amadores que utiliza*m Processing* para aprendizagem (didático), protótipos, e produção.[2]

Reas e Fry (2010, p. 5) explicam que o *Processing* é um dialeto da linguagem de programação Java; a sintaxe é praticamente igual, mas o *Processing* adiciona características personalizadas relacionadas ao gráfico e interação. Os elementos gráficos do *Processing* estão mais relacionados ao *PostScript* (a base do PDF) e *OpenGL* (um ambiente gráfico 3D). Por causa dessas características compartilhadas, aprender *Processing* pode ser um jeito de começar a programar em outras linguagens e a usar diferentes ferramentas.

A abordagem de programação do *Processing* é utilizar rotinas já definidas. Em essência essa linguagem e suas bibliotecas adicionais fazem uso de Java, que por sua vez tem elementos idênticos à linguagem de programação C. Essa herança permite ao *Processing* fazer uso de mais de 30 anos de melhoramentos de linguagens de programação e facilita o entendimento do *Processing* para muitas pessoas que já estejam familiarizadas a programação de *softwares*. (REAS;FRY, 2006)

O *Processing* torna fácil a fabricação de *softwares* para desenho e animação. Simplifica a extensão e integração de mídias (áudio, vídeo e eletrônica). Oferece maneiras muito simples para se trabalhar com recursos gráficos 2D e 3D. Há também versões compatíveis com plataformas de dispositivos móveis e microcontroladores. (REAS;FRY, 2006)

Na comunidade online do *Processing*, seus membros compartilham programas, contribuem com código fonte, respondem a questões do fórum de discussões, além de desenvolverem bibliotecas que estendem as possibilidades de aplicação da linguagem. A comunidade já tem mais de 70 bibliotecas que facilitam diversas áreas entre elas: visão computacional, visualização de dados, música, redes de computadores e eletrônica.[3]

O *Processing* tem primitivas para gráficos em 3D e uma biblioteca baseada em OpenGL, tornando os recursos gráficos simples para serem utilizados, pois o

2 http://processing.org. Acesso em 4/3/2014.
3 http://wiki.processing.org. Acesso em 4/3/2014.

intuito do *Processing* é ser fácil de ser utilizado principalmente por usuários iniciantes em programação.

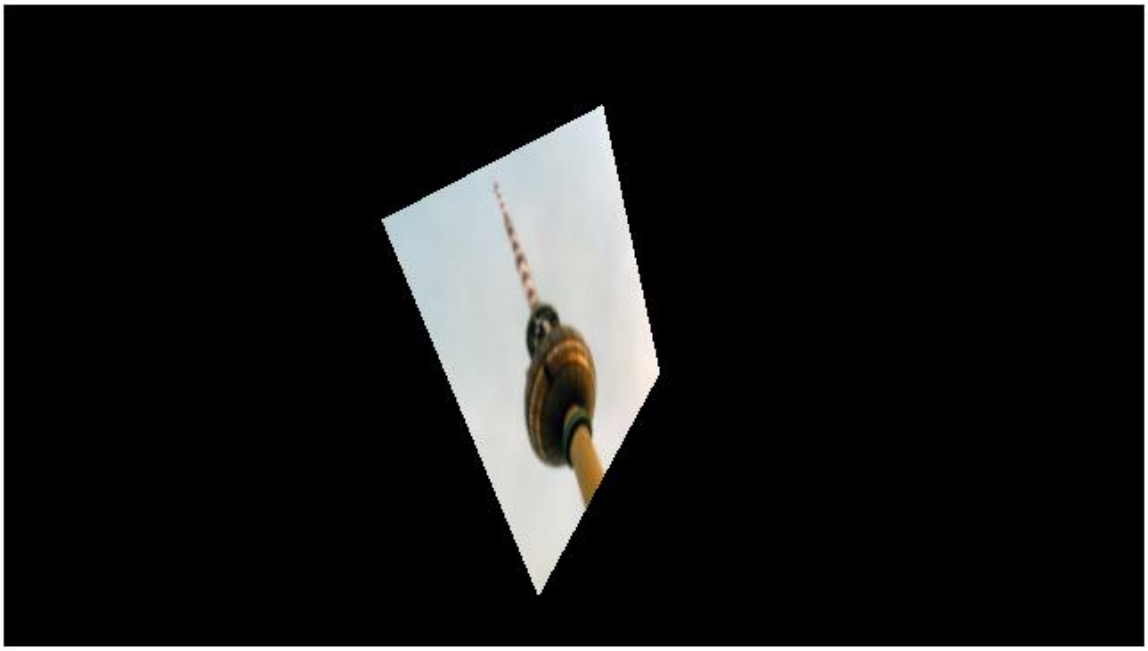

Figura 5: Execução de exemplo de renderização de imagem no espaço 3D. (http://processing.org/examples/texturequad.html. Acesso em 08/05/2014)

A figura 5 mostra a execução de uma texturização de imagem no espaço tridimensional e pode ser observada a simplicidade do código em *Processing* que gerou essa tela. O código fonte do Algoritmo 1 e a tela do programa (figura 6) foram extraídos de um dos exemplos do site oficial do *Processing*[4]

Algoritmo 1: texturização no espaço 3D utilizando primitivas do *Processing.* (http://processing.org/examples/texturequad.html. Acesso em 08/05/2014)

```
PImage img;

void setup() {
  size(640, 360, P3D);
  img = loadImage("berlin-1.jpg");
}

void draw() {
  background(0);
  translate(width / 2, height / 2);
  rotateZ(PI/6);
  rotateY(map(mouseX, 0, width, -PI, PI));
  beginShape();
```

4 http://processing.org/examples/texturequad.html. Acesso em 08/05/2014.

```
  texture(img);
  vertex(-100, -100, 0, 0,   0);
  vertex( 100, -100, 0, 400, 0);
  vertex( 100,  100, 0, 400, 400);
  vertex(-100,  100, 0, 0,   400);
  endShape();
}
```

Focando em simulação com o *Processing* podemos relatar as seguintes bibliotecas criadas para facilitar o desenvolvimento de simuladores de corpos: JBox2D, Fisica, BoxWrap2D e Toxiclibs, elas possuem métodos que encapsulam eventos físicos como a colisão de objetos.

JBox2D[5] é na verdade uma biblioteca Java baseada em LiquidFun[6] da Google (biblioteca de corpos rígidos e fluídos também baseada na Box2D) e Box2D (biblioteca em C++ desenvolvida por Eric Catto) que faz simulação 2D de corpo rígido para jogos (CATTO, 2011). É comum usar o JBox2D em *Processing* pois Java é suportado pelo *Processing*. O JBox2D suporta funcionalidades como por exemplo, física de corpos rígidos, gravidade, tratamento de contato persistente, tração de escorregamento, motores, sensores, detecção de colisão, dinâmica, cinemática e estática de corpos.

BoxWrap2D[7] é uma biblioteca onde foi construída uma camada sobre a biblioteca JBox2D para permitir seu funcionamento em *Processing*. Seu funcionamento é basicamente baseado em um mundo 2D onde seus objetos são criados e obedecem as leis da física, como a gravidade.

Toxiclibs[8] é uma biblioteca para o *Processing*, que visa a re-usabilidade de código e auxílio em diversas aplicações como simulação, visualização de dados, animação e design de interação/interface. Diferente das bibliotecas anteriormente citadas Toxiclibs suporta também gráficos 3D.

5 http://jbox2d.org/. Acesso em 21/4/2014.
6 http://google.github.io/liquidfun/. Acesso em 21/4/2014.
7 http://wiki.processing.org/w/BoxWrap2d#Interacting_with_bodies. Acesso em 21/4/2014.
8 http://toxiclibs.org/about/. Acesso em 21/4/2014.

bRidid[9] que disponibiliza classes para facilitar a manipulação de uma biblioteca suportada pelo Java chamada jBullet[10] no *Processing*. O bRigid permite interação de corpos rígidos em 3D.

Fisica[11] é uma biblioteca do *Processing* criada por Ricard Marxer. Ela é baseada na biblioteca para o Java chamada JBox2D com o intuito de tornar mais simples a criação de modelos físicos utilizando uma API orientada a objetos similar a PPhy2D (atualmente sem manutenção) também para o *Processing*. O mundo do Fisica é 2D, onde se cria objetos que são inseridos a esse mundo e interagem de acordo com as configurações de gravidade, massa, atrito, etc (PROCESSING.ORG, 2012). A biblioteca Fisica foi utilizada no desenvolvimento deste simulador, uma vez que os objetos do simulador não se movem no eixo z, uma biblioteca com mundo 2D pode ser utilizada no calculo das forças e movimentos nos eixos x e y. O principal motivo da escolha dessa biblioteca foi que sua interface é simples de ser utilizada e sua documentação é completa, ajudando nas implementações.

Para justificar o uso d*o Processing* no *retrofitting*[12] do simulador do AEDROMO, considera-se algumas características d*o Processing,* que por ter sido usado para visualização de dados e design gráfico, jogos e simulações possui uma variedade de bibliotecas disponíveis. Outra característica presente do *Processing* e interessante para construção de simuladores é a existência de primitivas de renderização de gráficos tridimensionais, baseada na biblioteca OpenGL. E por último, o *Processing* é de fácil manipulação até mesmo por pessoas com pouca experiência em programação.

9 http://www.lab-eds.org/bRigid/. Acesso em 7/11/2014.

10 http://jbullet.advel.cz/. Acesso em 21/01/2015.

11 http://www.ricardmarxer.com/fisica/. Acesso em 21/4/2014.

12 [1]*Retrofitting* (“Prover a algo (automóvel, computador, ou fabricado, por exemplo) alguma parte (dispositivos ou recurso) que não existia no atual momento no qual aquilo foi originalmente fabricado.”). Fonte : http://www.thefreedictionary.com/retrofit. Acesso em 4/03/2014.

4 Módulos de Software Desenvolvidos neste Trabalho

Principalmente dois módulos de software foram desenvolvidos, um simulador e uma biblioteca de auxilio ao usuário. A versão do simulador desenvolvida é composta por até dois robôs móveis e por objetos inanimados (cubos ou bolas), tanto a quantidade de robôs quanto a de objetos inanimados é determinada antes da execução do simulador em um arquivo de configuração, que será melhor explicado ao decorrer do capítulo e a biblioteca possui algumas funções que encapsulam as diretrizes de conexão, ou o modo como os robôs são controlados para facilitar o desenvolvimento de aplicação pelos alunos para o AEDROMO.

4.1 Simulador

A principal vantagem desta nova versão do simulador, em relação a anterior, é oferecer a visualização em três dimensões. Esta versão utiliza as facilidades de renderização 3D disponíveis no *Processing* de forma que ele passe uma sensação mais concreta do ambiente ao usuário.

Este simulador é flexível quanto a sua configuração no servidor, antes de iniciar sua execução pode-se definir a decoração da arena (desenho da superfície onde os objetos são posicionados) e em um arquivo de configuração os atributos: altura e largura da arena, a quantidade de objetos inanimados já definindo seu tipo (bola ou cubo), sua cor e suas coordenadas iniciais na arena; e também a quantidade de robôs juntamente à suas coordenadas iniciais. Tanto as dimensões quanto as coordenadas inseridas são dadas em milímetros. A figura 6 mostra um exemplo de um arquivo de configuração.

```
1  <?xml version="1.0"?>
2  <AEDROMO>
3    <arena largura="800" altura="600"></arena>
4    <elementos>
5      <robo id="0" x="300" y="200"></robo>
6      <robo id="1" x="400" y="300"></robo>
7      <objeto id="2" tipo="bola" x="50" y="70" red="255" green="0" blue="0"></objeto>
8      <objeto id="2" tipo="cubo" x="400" y="200" red="0" green="255" blue="0"></objeto>
9    </elementos>
10 </AEDROMO>
```

Figura 6. Exemplo de um arquivo de configuração.

Neste simulador tanto a detecção quanto o tratamento de colisões foram realizados por meio de uma biblioteca física disponível no site do *Processing* chamada Fisica apresentada nos capítulos ateriores. A implementação consiste em adicionar um corpo rígido para cada um dos objetos, quadrados para os robôs e cubos; e círculos para a bola. São utilizados formas 2D porque o Fisica é uma biblioteca 2D. Como o simulador apesar de representar objetos 3D, funciona exclusivamente sobre o plano da arena, ou seja, em z=0, ignorando-se assim o terceiro eixo, esta biblioteca Fisica representa o ambiente real.

Já o cálculo das velocidades em x e y; e o ângulo dos robôs são feitos de acordo com a cinemática dos robôs móveis com tração diferencial apresentada nos capítulos anteriores. Veja a seguir o cálculo simplificado implementado no simulador (algoritmo 2):

Algoritmo 2: Cálculo da cinemática dos robôs e manipulação da biblioteca Fisica.

```
//  Divide-se pelo frameRate pois trata-se do deslocamento em um dTempo =
1/frameRate
dAngulo = (-velocidadeAngularRoda*raioRoda) / distanciaEntreRodas / frameRate;
corpoFisica.adjustRotation(dAngulo);
velocidade = (-velocidadeAngularRoda * raioRoda)/2;
velocidadeEmX = velocidade*cos(corpoFisica.getRotation());
velocidadeEmY = velocidade*sin(corpoFisica.getRotation());
corpoFisica.setVelocity(velocidadeEmX, velocidadeEmY);
```

4.2 Biblioteca

Foi desenvolvida também uma biblioteca em *Processing* para uso pelo aluno no desenvolvimento de seu software. A intensão é facilitar o uso do ambiente, pois esta biblioteca encapsula todas as funções que realizam a conexão com o servidor (sendo ele o simulador ou o ambiente real), as funções que tratam a troca de pacotes e as conversões de dados. Especificamente, esta biblioteca disponibiliza uma lista de funções apresentadas na tabela 1.

Tabela 1. Lista de comandos disponíveis na biblioteca.

Comando (parâmetros):retorno	**Descrição**
conexao(número total de elementos, ou seja, quantidade de objetos mais a quantidade de robôs, porto do robô, IP do servidor do ambiente (real ou virtual));	Se conecta ao simulador ou ao ambiente real (número do IP indicado) e cria um objeto de conexão que será utilizado para o envio de comandos, recebe o porto do robô o qual será controlado, e a quantidade de objetos presentes na arena para a criação da estrutura de dados necessária para a comunicação entre cliente e servidor.
recebeEstado() : vetor de coordenadas dos objetos;	Uma função de recebimento das coordenadas dos objetos interagindo no ambiente. Esta função retorna um vetor dos objetos presentes no ambiente e suas coordenadas e ângulos atuais.
executaComando(velocidade da roda esquerda, velocidade da roda direita) : sem retorno;	Aciona cada um dos motores de acordo com o comando recebido, por exemplo, 11_{16}, roda direita e esquerda ligadas para frente, fazendo o robô se mover para frente; 10_{16}, roda direita ligada para frente e esquerda parada, fazendo o robô girar no eixo da roda esquerda; e 91_{16}, roda direita ligada para trás e direita ligada para frente, fazendo o robô girar no próprio eixo.
vaiPara(coordenada na arena de destino do robô) ;	Faz o robô ir diretamente para uma coordenada específica dentro da arena (coordenada recebida por parâmetro).

.

5 Resultados e trabalhos futuros

O simulador desenvolvido tem uma interface muito simples, ele apresenta uma imagem dinâmica do que está ocorrendo com os robôs no espaço da arena.

A verificação do simulador foi realizada apenas pela comparação entre o ambiente real e o virtual. Na figura 7 está uma comparação entre o ambiente real e o virtual quanto a sua visualização.

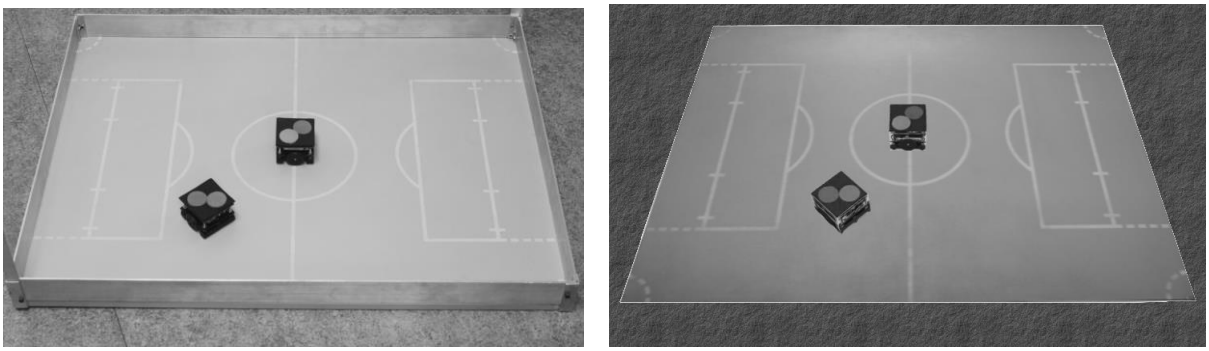

Figura 7. Comparação da visualização do ambiente real (à esquerda) com a gerada pelo simulador (à direita).

O simulador apresenta resultados interessantes quando comparando o deslocamento controlado dos robôs com o ambiente real. Considerando a execução de um trajeto de um robô saindo da posição (10, 10) e se deslocando até a posição (70, 50) nos dois ambientes (real e simulado), o resultado obtido pode ser apreciado na figura 8. Nesta figura observamos que os trajetos, representados pelas linhas pretas, realizados nos dois ambientes são semelhantes. As diferenças observadas estão relacionadas a deficiências construtivas do robô real que dificultam a realização de um percurso em linha reta.

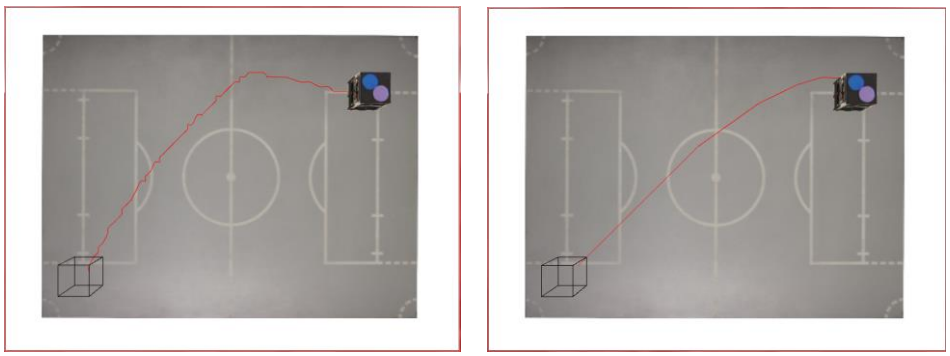

Figura 8. Comparação do deslocamento realizado pelo robô no ambiente real (à esquerda) e no simulado (à direita).

Outra comparação interessante é programar o robô para realizar um quadrado, se movimentando nos pontos P1(550, 150), P2(550, 450), P3(250, 450) e P4(250, 150) respectivamente como pode ser visto no algoritmo 3, onde temos o código executado. E na figura 9, onde temos a comparação dos trajetos do ambiente real à esquerda e do ambiente virtual à direita. As diferenças de trajetos podem ser explicadas pelos ruídos presentes no ambiente real o que torna o trajeto mais "quadrado" enquanto o trajeto do ambiente virtual é mais suave e "arredondado".

Algoritmo 3. Código executado para forma o quadrado.

```
Conexao cn;

void setup() {
  cn = new Conexao(4, 7878, "127.0.0.1");
}

boolean chegou = false, chegou2 =false, chegou3=false, chegou4=false;

void draw() {
  if (!chegou) {
    if (cn.vaiPara(550, 150)) {
      chegou = true;
    }
  }else if(!chegou2){
      if (cn.vaiPara(550, 450))
        chegou2 = true;
  }else if (!chegou3){
      if (cn.vaiPara(250, 450))
      chegou3 = true;
  }else if(!chegou4)
        if (cn.vaiPara(250, 150))
          chegou4 = true;
}
```

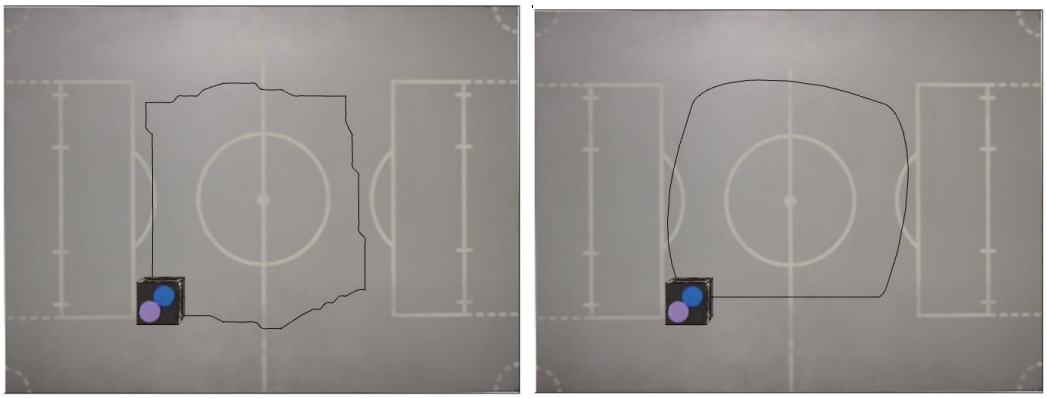

Figura 9. Comparação do deslocamento em forma de quadrado realizado pelo robô no ambiente real (à esquerda) e no simulado (à direita).

Por último a bola foi posicionada ao centro da arena, o robô foi programado saindo do ponto A para ir até a bola em B, e consequentemente consequente empurrando-a até o gol no ponto P(700,300). No algoritmo 4 pode-se observar o código executado, quanto na figura 10, pode-se observar as diferenças de trajetos nos ambientes real e virtual, o trajeto da bola em vermelho e o do robô em preto. Mais uma vez as diferenças de percurso são causadas pelo maior precisão do ambiente virtual que não contém ruídos nem erros de calibração, porém ambos conseguem executar o objetivo com a mesma programação.

Algoritmo 4. Código executado para levar a bola ao gol.

```
1   Conexao cn;
2
3   void setup() {
4     cn = new Conexao(4, 7878, "127.0.0.1");
5   }
6
7   boolean chegou = false;
8
9   void draw() {
10  if (!chegou) {
11      if (cn.vaiPara(400, 300)) {
12        chegou = true;
13      }
14    }else cn.vaiPara(700,300);
15  }
16
```

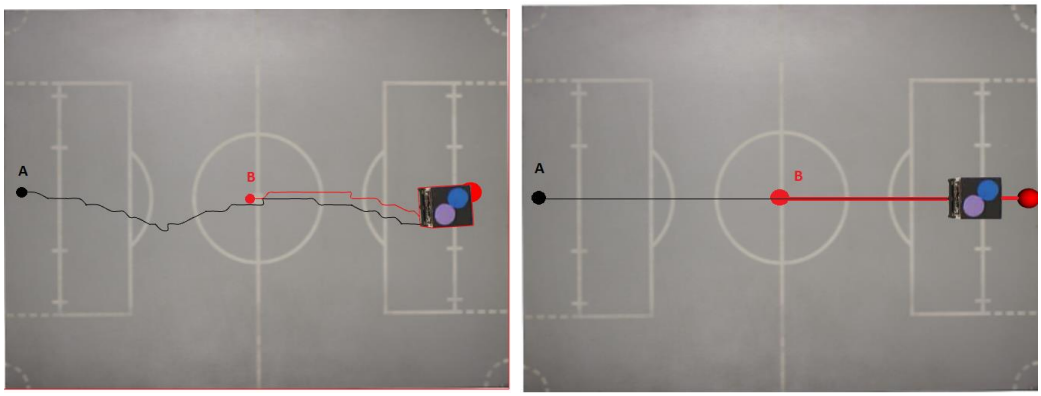

Figura 10. Comparação do deslocamento realizado pelo robô e pela bola, ao arrastá-la até o gol tanto no ambiente real (à esquerda) quanto no simulado (à direita).

Quanto à biblioteca, a simplificação na criação do software do aluno fica evidente quando se compara o número de linhas de código para executar o percurso descrito acima (representado na figura 8). O código escrito em Java puro tem 177 linhas, o no *Processing* utilizado a biblioteca (algoritmo 5) tem 8 linhas de código.

Algoritmo 5. Programa utilizando a biblioteca.

```
1  Conexao cn;
2  void setup() {
3    cn = new Conexao(4, 7879, "127.0.0.1");
4  }
5
6  void draw() {
7   cn.vaiPara(700, 500);
8  }
9
```

Como trabalho futuro, no simulador seria interessante adicionar ruídos para que seu comportamento se aproxime do comportamento do ambiente real. Pode-se também parametrizar mais variáveis, como o ângulo inicial de cada objeto e a velocidade dos robôs. E também é preciso ajustar o ambiente real para ele ser configurável quanto ao número e tipo de objetos inanimados.

6 Conclusão

A melhoria oferecida pela visualização tridimensional no simulador, leva aos alunos uma maior proximidade com o ambiente real, tornando mais concretas suas experiências obtidas através de simulador.

A biblioteca possibilita que um maior número de pessoas possa utilizar o AEDROMO, uma vez que não exige delas conhecimentos específicos de comunicação e controle dos robôs, fazendo com que a pessoa se concentre na solução da atividade proposta.

Assim, a biblioteca e o simulador apresentados para o AEDROMO pode ampliar as possibilidades práticas educativas deste ambiente, proporcionando um maior alcance desta ferramenta para o ensino da robótica, bem como para apoio a outras disciplinas, como o ensino de programação.

REFERÊNCIAS

ALVES, Silas F. R.; FERASOLI Filho, Humberto; PEGORARO, Renê; CALDEIRA, Marco A. C.; YONEZAWA, Wilson M.; ROSÁRIO, João. Educational Environment for Robotic Applications in Engineering. In: Eurobot Conference 2011 - 4th International Conference on Research and Education in Robotics, 2011, Praga - República Tcheca. **Anais:** Research and Education in Robotics - EUROBOT 2011. Berlin - Alemanha: Springer, 2011. v. 161. p.17-28, 2011a.

ALVES, Silas F. R.; FERASOLI Filho, Humberto; PEGORARO, Renê; CALDEIRA, Marco A. C.; YONEZAWA, Wilson M.; ROSÁRIO, João. Proposal of educational environments with mobile robots. In: **Robotics, Automation and Mechatronics (RAM)**, 2011 IEEE Conference on. IEEE, p. 264-269, 2011b.

ALVES, Silas F. R.; FERASOLI Filho, Humberto; PEGORARO, Renê; ROSÁRIO, João. Environment for Teaching and Development of Mobile Robot Systems. **Electronics, Robotics and Automotive Mechanics Conference (CERMA)**, pp. 302–307, 2010.

ALVES, Silas FR; ROSÁRIO, João M.; FERASOLI Filho, Humberto. Plataforma de Software para Robôs Móveis Autônomos. In: **I CTDR Concurso de Teses e Dissertações de Robótica / Latin American Robotics Symposium / Brazilian Robotics Symposium**, Fortaleza, CE. 2011.

BROWNING, Brett; TRYZELAAR, Erick. Übersim: a multi-robot simulator for robot soccer. In: **Proceedings of the second international joint conference on Autonomous agents and multiagent systems**. ACM, 2003. p. 948-949, 2003.

CATTO, Erin. **Box2D v2.2.0 User Manual.** 2011. < http://www.box2d.org/manual.html#_Toc258082967 >. Acesso em 21/4/2014.

FERASOLI Filho, Humberto; PEGORARO, Renê; CALDEIRA, Marco A. C.; ROSÁRIO, João. AEDROMO- An Experimental and Didactic Environment with Mobile Robots. Proceedings. **The 3rd International Conference on Autonomous Robots and Agents**, 2006.

KITANO, Hiroaki; ASADA, Minoru; KUNIYOSHI, Yasuo; NODA, Itsuki; OSAWA, Eiichi. Robocup: The robot world cup initiative. In:**Proceedings of the first international conference on Autonomous agents**. ACM, 1997. p. 340-347, 1997.

PAPERT, S. **Mindstorms: children, computers, and powerful ideas**, New York, NY, USA: Basic Books, Inc. 1980.

PROCESSING.ORG. **Fisica / JBox2D physics engine.** 2012. < http://wiki.processing.org/w/Fisica_/_JBox2D_physics_engine >. Acesso em 21/04/2014.

REAS, Casey; FRY , Ben. **Getting Started with Processing**. First Edition. Sebastopol, CA, USA. O'Reilly Media/Make, p. 5, 2010.

REAS, Casey; FRY, Ben; Processing: programming for the media arts. **AI & Society**, 20(4):526-538, 2006.

STARANOWICZ, Aaron; MARIOTTINI, Gian Luca. A survey and comparison of commercial and open-source robotic simulator software. In: **Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments**. ACM, 2011. p. 56, 2011.

APÊNDICE A – Relatório técnico de desenvolvimento

Roteiro de desenvolvimento técnico

Na primeira parte do desenvolvimento do projeto o foco foi trabalhar na parte gráfica, onde a arena e seus objetos seriam devidamente desenhados em gráficos tridimensionais. O design escolhido foi a renderização dos objetos 3D utilizando fotos dos objetos do ambiente real, assim a aparência do simulador seria muito parecida com o AEDROMO, dando a impressão de estar executando no ambiente real e observar por meio da câmera global.

Ao início foi incorporado ao simulador a leitura de um arquivo de configurações com extensão .csv (Comma separated variables), mas decidiu-se que um arquivo com a extensão xml seria mais adequado, então o arquivo configuração.xml foi adotado, no qual é armazenado as especificações iniciais para a montagem do simulador como as dimensões da arena de trabalho dos robôs, a quantidade de objetos, suas coordenadas iniciais e cores; projetamos então um layout a ser seguido para a criação deste arquivo

Após a leitura do arquivo de configuração, a interface do simulador já começa a ser desenhada de acordo com as especificações do usuário.

O próximo passo definido foi implementar a tecnologia de *sockets*, tornando possível realizar a conexão cliente e servidor. Para essa etapa foi seguido a lógica de programação já existente na versão do simulador em Java. Na qual é feita a criação de uma *thread* que prepara a comunicação para cada um dos robôs no ambiente.

Depois de estabelecida a conexão, é feita a troca de dados entre clientes e servidor, consequentemente é feita a atualização das coordenadas dos robôs e objetos e ao atualizar a tela os objetos são movimentados de acordo com o programa do cliente.

O ponto alto do desenvolvimento foi a implementação da biblioteca para colisão dos objetos, primeiro foram realizados testes com a biblioteca bRigid (3D), porém sem resultados positivos pois a biblioteca apresenta uma interface complicada. A biblioteca que acabou sendo utilizada foi Fisica (2D), como os corpos só interagem em z=0 pudemos utilizar uma biblioteca 2D sem alteração da lógica.

Depois fizemos os cálculos das componentes de velocidade em x e y, juntamente com o ângulo dos robôs seguindo as equações de cinemática de robôs móveis com tração diferencial apresentadas durante o desenvolvimento deste trabalho.

Por último fizemos a implementação da biblioteca na classe Java Conexao.java, que contem um construtor com os parâmetros de conexão e possui os métodos de movimentação dos robôs ora recebendo as coordenadas destino ora recebendo os comando de liga/desliga de cada um dos motores dos robôs (direito/esquerdo).

APÊNDICE B – Artigo publicado no 5th Workshop of Robotics in Education

Simulador do Ambiente Educacional Didático de Robôs Móveis - AEDROMO

Mariana Shimabukuro, Rene Pegoraro

Departamento de Computação – Universidade Estadual Paulista "Júlio de Mesquita Filho" – UNESP – Bauru – SP - Brazil

mah.akemi21@gmail.com, pegoraro@fc.unesp.br

Abstract. *This paper describes a simulator for an educational environment to control mobile robots and a support library for robot programming in the 'Processing' programming language. Finally, a comparison between the behavior of the simulator and the real environment was made, and the simplification offered by the usage of its library to build students' software can be exemplified.*

Resumo. *Este artigo apresenta aspectos do desenvolvimento de um simulador de um ambiente didático com robôs móveis e uma biblioteca de suporte a programação dos robôs utilizando a linguagem de programação 'Processing'. Finalmente, compara-se o comportamento do simulador com o ambiente real e a simplificação oferecida pela utilização da biblioteca na construção do software do aluno.*

1. Introdução

ALVES et al. (2011a) descreve a robótica como uma abordagem de sucesso no âmbito educacional, por ela ser intrinsecamente multidisciplinar, que estimula o trabalho em grupo e promove o retorno de forma visual motivadora. Assim como os computadores, a robótica é um recurso tecnológico auxiliar utilizável no processo educacional que pode contribuir para o desenvolvimento cognitivo do aluno e habilidades intelectuais específicas. Ela se oferece como uma ferramenta pedagógica interessante sobre vários aspectos e assim deve ser encarada e explorada [PAPERT 1980].

Há dificuldades de disponibilizar um número mínimo de robôs por número de alunos, além do espaço físico requerido. Nesse contexto o uso de simuladores é uma opção, pois os alunos podem testar suas soluções sem utilizar robôs reais. Depois de ter entendido o funcionamento do robô e do ambiente, ele pode testar no ambiente real.

Este artigo apresenta um simulador com visualização tridimensional para o Ambiente Experimental Didático com Robôs Móveis (AEDROMO) e uma biblioteca destinada ao usuário, usando a linguagem *Processing*. Este artigo está organizado como segue: o item 2 descrever as tecnologias utilizadas; no item 3 são apresentados as características do novo simulador e da biblioteca de auxílio ao software do aluno, no item 4 estão os resultados do desenvolvimento e finalmente no item 5 a conclusão.

2. Fundamentação

O desenvolvimento deste trabalho está apoiado no AEDROMO, seu simulador já existente e na linguagem de programação *Processing*.

2.1. AEDROMO

O AEDROMO é um ambiente desenvolvido no Laboratório de Integração de Sistemas e Dispositivos Inteligentes (LISDI) do Departamento de Computação da Faculdade de Ciências da UNESP, campus de Bauru, onde robôs interagem para realizar tarefas simples. Os robôs neste ambiente são localizados globalmente e controlados remotamente por um computador. O AEDROMO vem sendo desenvolvido como um ambiente de teste para diversas disciplinas. Este ambiente robótico é adaptável para oferecer tarefas em diferentes níveis, integrando objetivos educacionais com suporte tecnológico. Um de seus objetivos é ser aplicável como apoio educacional para o desenvolvimento de ciências, conceitos de lógica e de programação nos níveis fundamental e médio de ensino [ALVES et al. 2010].

O AEDROMO pode apresentar diversas opções para atividades didáticas, como para pesquisa e entretenimento. Ele tem a capacidade de reconhecer dois robôs e objetos coloridos através de uma câmera e possibilita a realização de diversas atividades, onde cada robô é controlado por um programa desenvolvido por um aluno. O objetivo das atividades é experimentar alguns aspectos da robótica e da computação. As atividades propostas em [ALVES et al. 2011b] incluem as tarefas: (i) Colecionando Coisas, a tarefa é recolher um objeto para uma área reservada. (ii) Caça e Caçador, os dois robôs são colocados no ambiente e com objetivos diferentes. O primeiro foge do segundo enquanto o segundo tenta pegar o primeiro. (iii) Resolvendo o Labirinto, consiste no robô encontrar um caminho em um labirinto virtual. (iv) Jogo de Tênis, o objetivo está em empurrar a bola para o outro lado da linha. Uma outra tarefa de grande apelo entre os jovens é o futebol, onde dois robôs e uma bola interagem na arena em uma partida. Cada uma dessas atividades é acompanhada por uma arena com uma decoração diferente e específica, tornando mais concreta a atividade em desenvolvimento. Neste trabalho, como apresentado na figura 2, a decoração referente ao futebol foi utilizada.

A programação das aplicações no AEDROMO é realizada em linguagens de alto nível exigindo conhecimentos específicos em linguagens de programação e características relacionadas à própria conexão de rede do AEDROMO. Assim, qualquer simplificação que possa permitir uma utilização mais fácil deste ambiente é bem vinda e tem sua importância para a ampliação de sua utilização.

2.1.1 Objetos e composição do ambiente

O AEDROMO é composto por um ou dois robôs (como o apresentado na Figura 1) e também por um objeto inanimado: bola ou cubo; o qual é utilizado na interação com os robôs por meio de atividades propostas. Os robôs móveis foram especialmente desenvolvidos pelo LISDI para serem utilizados neste ambiente, eles apresentam pequenas dimensões, com sistema de tração diferencial (*differential drive*). O AEDROMO basicamente é formado por dois robôs, um computador, uma arena, uma câmera global do tipo *webcam* e um transmissor.

2.1.2 Arquitetura do AEDROMO

O AEDROMO utiliza a arquitetura cliente-servidor. [ALVES et al. 2010]. Ou seja, no AEDROMO há dois tipos de programas: o Servidor e o Software do Aluno funcionando como cliente do ambiente. O Servidor é responsável pela implementação do sistema de visão computacional e pela comunicação com os robôs. Por outro lado, o Software do Aluno é onde o aplicativo de controle é desenvolvido para realizar a tarefa proposta. Este aplicativo é um cliente que recebe do Servidor as posições cartesianas de todos os objetos presentes na área de trabalho e envia ao Servidor os comandos de acionamento de cada roda do robô

sendo controlado, para que o servidor encaminhe estes comandos ao robô via sinal de rádio.

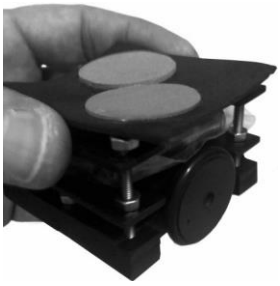

Figura 9. Robô segurado por um adulto [ALVES et al. 2011b].

O sistema de visão computacional utiliza as imagens adquiridas de uma câmera com seu plano de imagem paralelo ao solo para identificar a posição e orientação de cada robô, dadas por (x, y, θ) na qual theta (θ) é o ângulo da orientação do robô. Para facilitar o reconhecimento dos robôs, cada robô possui um marcador único. [ALVES; ROSÁRIO; FERASOLI 2012]

O simulador do AEDROMO, também conhecido como ambiente virtual, desenvolvido pelo GISDI, tem o intuito de melhorar a portabilidade e acessibilidade do AEDROMO de modo que o mesmo possibilite que o Software do Aluno seja testado, sem que seja necessária a presença do ambiente real (arena com os robôs físicos). O software do aluno deve apresentar o funcionamento semelhante em ambos ambientes – simulado e real. O simulador substitui o programa Servidor, fazendo com que a comunicação estabelecida entre o ambiente virtual e o cliente tenha a mesma interface, assim é exatamente igual à conexão entre o ambiente real e o cliente.

A comunicação entre o cliente e o servidor é dada através de uma rede utilizando o protocolo UDP/IP (*User Datagram Protocol over Internet Protocol*), permitindo que seja escrito um aplicativo de controle para cada robô em qualquer linguagem de programação que suporte "*socket's*". Adicionalmente, os aplicativos de controle podem ser executados no mesmo computador do servidor ou em diferentes máquinas [ALVES et al. 2011b]. Porem, a escolha da linguagem de programação tem impacto no que se deseja ensinar, este trabalho foca na utilização da linguagem de programação *Processing*.

2.2. PROCESSING

Processing é uma linguagem de programação, um ambiente de desenvolvimento e uma comunidade *online*. Inicialmente criado para servir como um caderno de esboço de *software* e ensinar os fundamentos de programação de computadores dentro de um contexto visual, *Processing* evoluiu para uma ferramenta de desenvolvimento para profissionais. Hoje, existem dezenas de milhares de estudantes, artistas, designers, pesquisadores e amadores que utilizam *Processing* para aprendizagem (didático), protótipos, e produção.13

13

http://processing.org. Acesso em 4/3/2014.

Reas e Fry [2010, p. 5] explicam que o *Processing* é um dialeto da linguagem de programação Java; a sintaxe é praticamente igual, mas o *Processing* adiciona características personalizadas relacionadas ao gráfico e interação.

A abordagem de programação do *Processing* é utilizar rotinas já definidas. Em essência essa linguagem e suas bibliotecas adicionais fazem uso de Java, que por sua vez tem elementos idênticos à linguagem de programação C. *Processing* torna fácil a fabricação de softwares para desenho e animação. Simplifica a extensão e integração de mídias (áudio, vídeo e eletrônica). Oferece maneiras muito simples para se trabalhar com recursos gráficos 2D e 3D. [REAS;FRY 2006]

A escolha do *Processing,* para o desenvolvimento deste trabalho, se deu devido às facilidades que este ambiente oferece. Ele é fácil de instalar, disponível para Windows, Linux e Mac OS X, tem uma interface muito simples para o usuário com apenas seis botões, diversas rotinas e bibliotecas prontas e tem código aberto. Além de permitir que classes Java sejam importadas diretamente a um projeto, flexibilizando o desenvolvimento.

3. Desenvolvimento

O foco deste trabalho está na descrição de uma biblioteca e de uma nova versão do simulador utilizando a linguagem *Processing.*

3.1. O Novo Simulador do AEDROMO

A principal vantagem desta nova versão do simulador, em relação a anterior, é oferecer a visualização em três dimensões. Esta versão utiliza as facilidades de renderização 3D disponíveis no *Processing* de forma que ele passe uma sensação mais concreta do ambiente ao usuário.

Este simulador é flexível quanto a sua configuração, antes de iniciar sua execução pode-se definir a decoração da arena (desenho da superfície onde os objetos são posicionados) e em um arquivo de configuração os atributos: altura e largura da arena, a quantidade de objetos inanimados já definindo seu tipo (bola ou cubo), sua cor e suas coordenadas iniciais na arena; e também a quantidade de robôs juntamente à suas coordenadas iniciais. Tanto as dimensões quanto as coordenadas inseridas são dadas em centímetros.

Da mesma forma que a versão anterior, apesar da dinâmica ser pouco considerada neste simulador, ele simplifica a realização de testes dos algoritmos em desenvolvimento.

3.2. Biblioteca de Suporte

Foi desenvolvida também uma biblioteca em *Processing* para uso pelo aluno no desenvolvimento de seu software. A intensão é facilitar o uso do ambiente, pois esta biblioteca encapsula todas as funções que realizam a conexão com o servidor (sendo ele o simulador ou ambiente real) e as funções que tratam a troca de pacotes e as conversões de dados. Especificamente, esta biblioteca disponibiliza uma função de indicação de conexão: conecta(IP do servidor do ambiente (real ou virtual), número do robô); uma função de recebimento das coordenadas dos objetos interagindo no ambiente: recebeEstado(vetor de coordenadas dos objetos); e duas funções de acionamento do robô sob controle, uma que aciona cada um dos motores: acionaMotores(velocidade da roda esquerda, velocidade da roda direita); e outra que faz o robô ir diretamente para uma coordenada específica dentro da arena: vaiPara(coordenada na arena de destino do robô);.

4. Resultados

O simulador usa uma interface muito simples, ele apresenta uma imagem dinâmica do que está ocorrendo no espaço da arena. A figura 2 apresenta uma comparação entre o ambiente real e o virtual quanto a sua visualização. Para a verificação do funcionamento do simulador considerou-se apenas a comparação visual do comportamento dos robôs entre o ambiente real e o virtual.

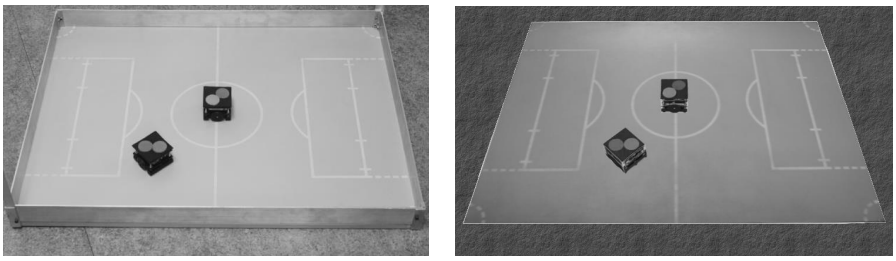

Figura 10. Comparação da visualização do ambiente real (à esquerda) com a gerada pelo simulador (à direita).

Na figura 3 é apresentado o deslocamento de um robô no ambiente real e no virtual, considerando a execução de um trajeto de um robô saindo da posição (10cm, 10cm) e se deslocando até a posição (70cm, 50cm) nos dois ambientes (real e simulado). Nesta figura, observamos que os trajetos, representado pelas linhas pretas, realizado nos dois ambientes, são semelhantes. As diferenças observadas estão relacionadas a deficiências construtivas do robô real que dificultam a realização de um percurso em linha reta.

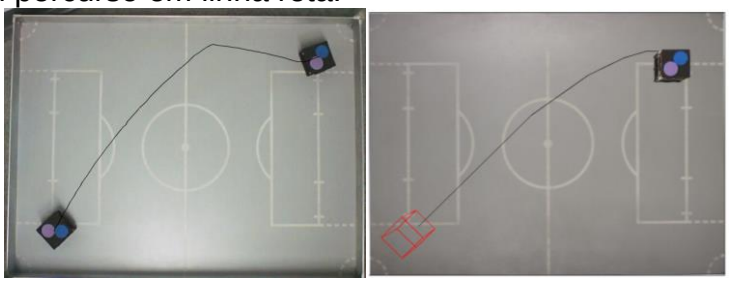

Figura 11. Deslocamento realizado pelo robô no ambiente real (à esquerda) e no virtual (à direita).

Paralelamente a biblioteca desenvolvida, esconde detalhes de pouca importância ao usuário, mas fundamentais ao sistema. Assim, a simplificação na criação do software do aluno fica evidente quando se compara o número de linhas de código para executar o percurso descrito (representado na Figura 3). O código ao qual o aluno tem acesso, em Java puro tem 177 linhas contra 24 linhas com a biblioteca no *Processing*.

5. Conclusão

Com a melhoria oferecida pela visualização tridimensional no simulador, os alunos sentem maior proximidade com o ambiente real, tornando mais concreto suas experiências obtidas através de simulador.

A biblioteca possibilita que um maior número de pessoas possa utilizar o AEDROMO, uma vez que não exige delas conhecimentos específicos de comunicação e controle dos robôs, fazendo com que a pessoa se concentre na solução da atividade proposta.

Assim, a biblioteca e o simulador apresentados para o AEDROMO podem ampliar as possibilidades práticas educativas deste ambiente, proporcionando um maior alcance desta ferramenta para o ensino da robótica e assuntos relacionados à programação.

Referências

ALVES, S. F. R.; FERASOLI Filho, H.; PEGORARO, R.; ROSÁRIO, J. (2010) Environment for Teaching and Development of Mobile Robot Systems. Electronics, Robotics and Automotive Mechanics Conference (CERMA), pp. 302–307.

ALVES, S. F. R.; FERASOLI Filho, H.; PEGORARO, R.; CALDEIRA, M. A. C.; WONEZAWA, W. M.; ROSÁRIO, J. (2011a) Educational Environment for Robotic Applications in Engineering. In: Eurobot Conference 2011 - 4th International Conference on Research and Education in Robotics, 2011, Praga - República Tcheca.

ALVES, S. F. R. ; FERASOLI FILHO, H. ; PEGORARO, R. ; CALDEIRA, M. A. C. ; YONEZAWA, W. M. ; ROSÁRIO, J. M. (2011b) Ambiente Educacional de Robótica Direcionado a Aplicações em Engenharia. In: X Simpósio Brasileiro de Automação Inteligente, São João del-Rei.

ALVES, S. F. R. ; ROSÁRIO, J. M. ; FERASOLI FILHO, H. (2012) Plataforma de Software para Robôs Móveis Autônomos. In: I CTDR Concurso de Teses e Dissertações de Robótica / Latin American Robotics Symposium / Brazilian Robotics Symposium, Fortaleza, CE

PAPERT, S. (1980) Mindstorms: children, computers,and powerful ideas, New York, NY, USA: Basic Books, Inc.

REAS, C.; FRY , B. (2010) Getting Started with Processing. First Edition. Sebastopol, CA, USA. O'Reilly Media/Make, p. 5.

REAS, C.; FRY, B. (2006) Processing: programming for the media arts. AI & Society, 20(4):526-538.

APÊNDICE C – Resumo publicado no Congresso de Iniciação Científica

Simulador do Ambiente Educacional Didático de Robôs Móveis - AEDROMO

Mariana Shimabukuro, Rene Pegoraro, UNESP – Bauru, Departamento de Computação, Bacharelado em Ciência da Computação, mas_akemi@hotmail.com, PIBIC

Palavras Chave: *AEDROMO, robótica, simulação.*

Introdução

Silas[1] descreve a robótica como uma abordagem de sucesso no âmbito educacional, por ela ser intrinsecamente multidisciplinar, que estimula o trabalho em grupo e promove o retorno de forma visual motivadora. Assim como os computadores, a robótica é um recurso tecnológico auxiliar utilizável no processo educacional que pode contribuir para o desenvolvimento cognitivo do aluno e habilidades intelectuais específicas[2].

No entanto, além do próprio espaço físico requerido, há dificuldades em disponibilizar um número razoável de robôs por número de alunos. Nesse contexto o uso de simuladores é uma opção interessante.

Este resumo apresenta um simulador com visualização 3D para o Ambiente Educacional Robótico de Robôs Móveis (AEDROMO) e uma biblioteca destinada ao usuário, destacando o uso da linguagem *Processing*[3] no desenvolvimento.

Objetivos

Desenvolver um simulador e uma biblioteca em *Processing* para o AEDROMO.

Material e Métodos

No desenvolvimento deste trabalho foi utilizada a linguagem de programação *Processing* para implementação do simulador e da biblioteca após o estudo sobre o AEDROMO, simulação e *Processing* na literatura.

Resultados e Discussão

O simulador tem uma interface muito simples, ele apresenta uma imagem dinâmica do que está ocorrendo com os robôs no espaço da arena. Para a verificação do simulador considerou-se apenas a comparação visual do comportamento dos robôs entre o ambiente real e o virtual.

Na figura 1 é apresentado o deslocamento de um robô no ambiente real e no virtual, considerando a execução de um trajeto de um robô saindo da posição (10cm, 10cm) e se deslocando até a posição (70cm, 50cm) nos dois ambientes (real e simulado). Nesta figura, observamos que os trajetos, representado pelas linhas pretas, realizado nos dois ambientes, são semelhantes. As diferenças observadas estão relacionadas a deficiências construtivas do robô real que dificultam a realização de um percurso em linha reta.

Paralelamente a biblioteca desenvolvida, esconde detalhes de pouca importância ao usuário, mas fundamentais ao sistema, tais como comunicação com o servidor ou simulador e transformação de coordenadas. Assim, a simplificação na criação do software do aluno fica evidente quando se compara o número de linhas de código para executar o percurso descrito (representado na Figura 1). O código ao qual o aluno tem acesso, em Java puro tem 177 linhas contra 24 linhas com a biblioteca no *Processing*.

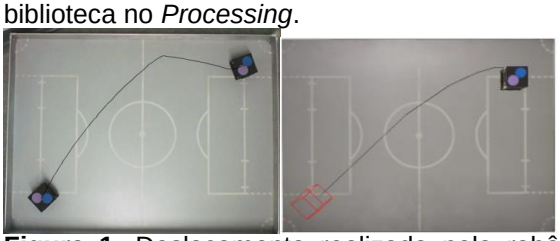

Figura 1. Deslocamento realizado pelo robô no ambiente real (à esquerda) e no virtual (à direita).

Conclusões

A biblioteca e o simulador apresentados para o AEDROMO podem ampliar as possibilidades práticas educativas deste ambiente, proporcionando um maior alcance desta ferramenta para o ensino da robótica e assuntos relacionados à programação.

Agradecimentos

Agradecimento à ProPe da UNESP e ao CNPq.

[1] ALVES, Silas F. R.; FERASOLI Filho, Humberto; PEGORARO, Renê; CALDEIRA, Marco A. C.; WONEZAWA, Wilson M.; ROSÁRIO, João. Educational Environment for Robotic Applications in Engineering. In: Eurobot Conference 2011

[2] PAPERT, Seymour. Mindstorms: children, computers,and powerful ideas, New York, NY, USA: Basic Books, Inc. 1980.

[3] http://processing.org/. Acesso em 20/08/2014.