UM AMBIENTE GRÁFICO PARA DESENVOLVIMENTO DE CONTROLE PARA ROBÔS MÓVEIS UTILIZANDO UMA SIMULAÇÃO 3D

Humberto Cardoso Marchezi*, Dr. rer. nat. Hans-Jorg Andreas Schneebeli*

*Avenida Fernando Ferrari s/número, Goiabeiras
Universidade Federal do Espírito Santo
Vitória, Espírito Santo, Brasil

Emails: hcmarchezi@hotmail.com, hans@ele.ufes.br

Abstract— This article describes the IRCE, an open-source control development environment for a population of mobile robots and their sensors, based on Player/Stage/Gazebo tools. This system allows the control algorithm verification via a 3D simulation. The creation of the virtual world and the control programs is done similarly to RAD environments. The resulting developed controls can be transferred to real robots.

Keywords— Mobile Robots, Simulation, Control, Integrated Environment.

Resumo— Este artigo descreve o IRCE, um ambiente de código-livre para desenvolvimento de controle de uma população de robôs móveis e seus sensores, baseado nas ferramentas do projeto Player/Stage/Gazebo. Este sistema permite a verificação do algoritimo dos controles via uma simulação 3D. A criação do mundo virtual e dos programas de controle é feito similarmente aos ambientes RAD. Os controles desenvolvidos podem ser transferidos para robôs reais.

Palavras-chave— Robôs Móveis, Simulação, Controle, Ambiente Integrado.

1 Introdução

Para que robô um desempenhe tarefas que envolvem interação com o ambiente externo, por meio de sensores, é necessário que este seja programado pelo operador, por exemplo, através de uma linguagem de programação.

O desenvolvimento de um programa de controle deve ser fácil e rápido para que um desenvolvedor possa se concentrar em "o que"o robô deve fazer ao invés de "como fazer". A simulação permite que o desenvolvedor verifique o código de controle de forma rápida e segura, além de possibilitar que sensores ou robôs hipotéticos sejam simulados para terem o seu impacto e eficiência medidos.

A fim de tentar resolver os problemas citados acima, pelo menos parcialmente, vários projetos já foram propostos (Biggs and MacDonald, 2003). Uma alternativa promissora é o projeto Player/Stage/Gazebo (P/S/G) (*The Player Project*, 2006), que consiste num conjunto de ferramentas para o desenvolvimento rápido de aplicações envolvendo robôs e sensores (Gerkey et al., 2003). Suas ferramentas ja são consideradas um padrão *de facto* na comunidade que usa código-aberto em robótica (MacDonald et al., 2005).

O projeto consiste de três aplicativos principais: Player, Stage e Gazebo. O Player é um programa que age como um servidor de controle de robôs (Gerkey et al., 2001). O servidor Player é um servidor de repositório de dispositivos distribuídos que oferece aos controles clientes uma interface orientada a rede para acessar atuadores e sensores de um robô (MacDonald et al., 2005). É ele quem faz a intermediação entre o robôs e seus sensores, que podem ser reais ou virtuais, e o programa de controle. Stage faz a simulação de um população de robôs móveis num ambiente bidimensional. Gazebo faz o mesmo utilizando, por sua vez, um mundo virtual tridimensional. Player também disponibiliza bibliotecas através dos quais pode-se desenvolver controles clientes. O controle desenvolvido pode ser usado para controlar dispositivos reais ou simulados sem necessidade de recompilação ou modificação no seu código-fonte já que a comunicação com os mesmos é intermediada por um servidor Player.

Ao utilizar um cenário 3D, a simulação de um robô passa ter a mais fidelidade já que a representação do mundo virtual é mais próxima da realidade. Por essa razão, apenas o Gazebo será utilizado daqui em diante.

2 Definição do Problema

ha algumas dificuldades na utilização das ferramentas do projeto P/S/G:

- Para fazer uma simulação no Gazebo, o usuário deve escrever manualmente um arquivo (*.world) no formato XML que descreve os robôs a serem simulados e a descrição do mundo virtual em 3D, o que consome tempo.

- O segundo problema é que o projeto P/S/G consiste numa variedade de aplicativos que devem ser executados em vários terminais diferentes, o que pode confundir ou diminuir a produtividade do trabalho de um usuário.

Essas razões são a motivação para a implementação de um ambiente integrado e interativo

para o desenvolvimento de controle de robôs móveis que fizesse a integração entre as ferramentas citadas acima, designado como IRCE.

Os objetivos desse ambiente são:

- Geração Gráfica Interativa de Mundo Virtual;
- Geração Interativa de Arquivo de Configuração;
- Edição e Compilação de Arquivos de Controle de Robô;
- Aumento da Agilidade no Desenvolvimento de Controle para Robôs Móveis;
- Ser de Utilidade no Ensino de Robótica.

3 Trabalhos Relacionados

Webots (Olivier Michel, 2004) é um *software* comercial usado para simulação de protótipos de robôs móveis e transferência de controle para robôs reais. Esse *software* tem um editor para o mundo virtual com simulação realística de física. Também permite ao usuário programar robôs nas linguagens C, C++ e Java via TCP/IP, e transferir o controle para robôs móveis reais. Programadores podem desenvolver software para o controle dos mesmos e testá-los numa simulação até que esteja pronto para ser transferido para um robô móvel real. Apesar de tudo, o webots é um pacote comercial fechado o que impede que pesquisadores possam adequar o sistema às suas necessidades, além do preço de sua licença ainda ser proibitivo para a maioria das instituições de ensino brasileiras.

4 Arquitetura do Ambiente

O ambiente de desenvolvimento consiste num aplicativo capaz de gerar o arquivo do mundo virtual do Gazebo (*.world), os arquivos de configuração (*.cfg) do Player e os programas de controle por meio de interações com o usuário.

No momento da simulação, o ambiente faz chamadas externas para o Gazebo, o Player e os controles conforme pode ser visto na Figura 1.

A execução acontece na seguinte ordem:

- Uma chamada externa ao Gazebo é feita passando o mundo virtual gerado como parâmetro;
- Várias chamadas externas ao Player são feitas, uma para cada arquivo de configuração;
- Cada programa de controle é executado.

O ambiente integrado acima neste artigo faz uso das ferramentas e dos arquivos definidos no projeto Player. Logo, para entender como este ambiente foi construído é necessário antes entender como as ferramentas desse projeto funcionam.

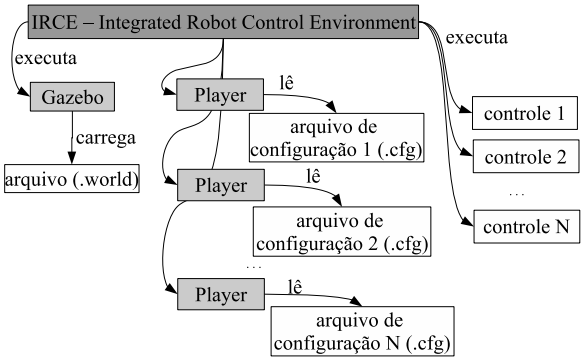

Figura 1: Integração entre IRCE, Gazebo, Players e programas de controle

4.1 O Projeto Player/Stage/Gazebo (P/S/G)

O Player é um servidor para o controle de robôs que funciona como uma interface para dispositivos de robô que abstrai os detalhes dos dispositivos robóticos dos programadores assim como nos sistemas operacionais. Dentro dessa ferramenta, existem três conceitos principais: a *interface*, o *driver* e o dispotivo (*device*) (MacDonald et al., 2005).

A *interface* define a sintaxe e a semântica das mensagens a serem trocadas para uma determinada classe de "entidades". Por exemplo, a *interface position2d* define um robô móvel com rodas e permite que o programador defina a posição e velocidade desse robô assim como receber informações de posição e velocidade.

O *driver* é o trecho de *software* que se comunica com os atuadores, sensores ou algoritmo e traduz as entradas e saídas para obedecer a uma ou mais *interfaces* abstraindo os detalhes específicos da "entidade". Por exemplo, o driver *p2os* pode ser usado para controlar vários robôs do fabricante ActivMedia como a série de robôs Pioneer e o AmigaBot.

O dispositivo pode ser definido como um *driver* ligado a uma *interface* com um endereço único. Todas as mensagem no Player ocorrem entre dispositivos via *interfaces* sendo que o *driver* nunca é acessado diretamente.

Para utilizar o Player, um arquivo de configuração deve ser passado como parâmetro (veja Apêndice A) . Este arquivo contém os dispositivos que serão controlados incluindo o nome do *driver* e a *interface* que implementa.

Para permitir o desenvolvimento de aplicações que controlem robôs, a instalação do Player inclui bibliotecas que permitem se comunicar com os dispositivos declarados no arquivo de configuração em linguagens como C, C++, Tcl e Java (veja exemplo em C++ no Apêndice B). Como a comunicação segue um protocolo aberto, versões para qualquer linguagem de programação podem ser desenvolvidas desde que sigam esse protocolo.

O Gazebo (Koenig and Howard, 2004) é um simulador 3D capaz de simular eventos físicos

como gravidade, detecção de colisão, momento, etc. Esta ferramenta foi feita para interagir com o Player a fim de simular o controle em dispositivos como os robôs Pioneer2AT e Pioneer2DX, SICK LMS 200, além de outros. Para utilizar o Gazebo, primeiro, o desenvolvedor deve escrever um arquivo XML específico (veja o Apêndice C) que descreve o mundo virtual e os dispositivos virtuais, chamados de modelos, que serão simulados.

4.2 A Integração entre Player e Gazebo

A interação, nesse caso, diz respeito a maneira como as ferramentas do projeto P/S/G se comunicam para atingir um certo objetivo. Para executar um controle para um robô Pioneer, só precisamos rodar o Player, devidamente configurado para o Pioneer, e um programa de controle conforme mostrado na Figura 2.

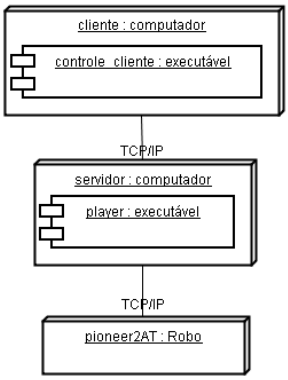

Figura 2: Controle envia comandos e recebe dados de um robô Pioneer através do Player

O programa de controle pode ser testado numa simulação. Para isso, devemos executar o Gazebo com o arquivo do mundo virtual, executar o Player com um arquivo de configuração e executar o programa de controle no final.

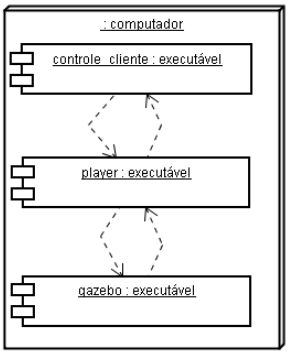

Figura 3: Controle envia comandos e recebe dados do mundo virtual pelo Gazebo através do Player

Conforme pode ser visto na figura 3, durante a simulação, temos que enganar o Player, fazendo-o pensar que esta controlando um robô Pioneer real, quando na verdade é um Pioneer virtual.

O programa de controle, por sua vez, continua imutável para os dois casos, já que referencia apenas *interfaces*.

5 O Ambiente Integrado para Desenvolvimento de Controle de Robôs

O ambiente integrado para desenvolvimento de controle de robôs (IRCE) preenche seis requisitos principais, que estão descritos na Figura 4.

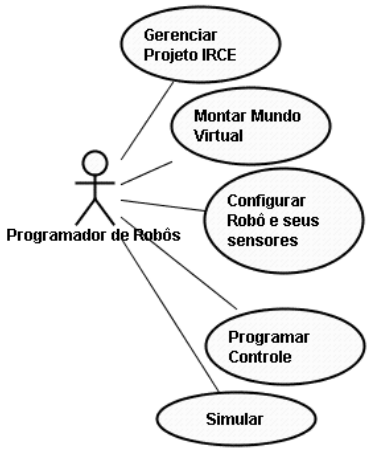

Figura 4: Requisitos do IRCE

A Figura 5 mostra a interface gráfica, que seguiu a analogia dos ambientes de desenvolvimento RAD.

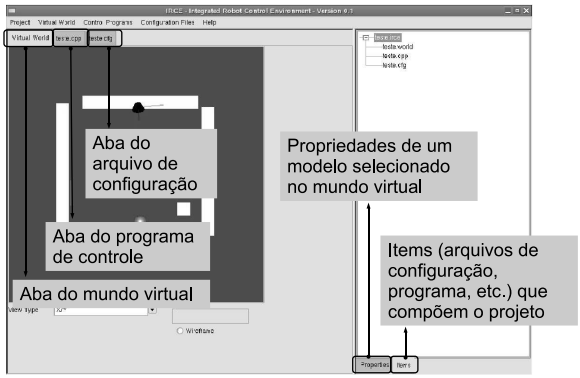

Figura 5: Janela principal do IRCE

As subseções a seguir mostram como cada requisito mencionado na Figura 4 foi implementado no IRCE.

5.1 Montar Mundo Virtual

Esse requisito é implementado pelo menu **Virtual World**.

Para criarmos um mundo virtual no projeto, devemos selecionar o menu (**Virtual World -> New World File**).

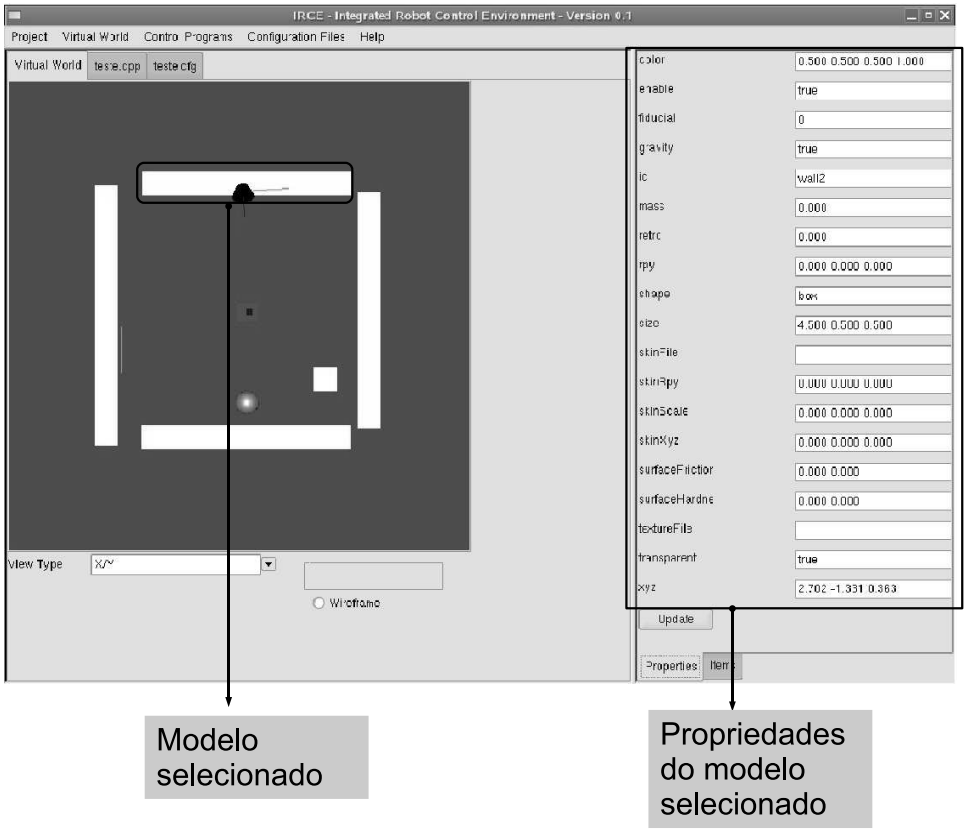

Figura 6: Painel de um mundo virtual em 3D e as propriedades de um modelo 3D selecionado

O sistema requisitará que o usuário informe o caminho onde o arquivo *World* será gravado. A partir do mundo virtual criado, podemos adicionar modelos 3D selecionando (**Virtual World -> Add 3D Model**).

Também é possível visualizar e alterar as propriedades de um modelo, selecionando-o diretamente na área de visualização, conforme mostrado na Figura 6. Para ativar as alterações, é necessário pressionar o botão **Update** abaixo.

5.2 Gerenciar Projeto IRCE

As operações de gerência de projeto (criar, carregar, salvar e editar) estão no menu **Project** no menu principal. Este é o primeiro menu a ser habilitado, pois é partir dele que os outros itens podem ser criados.

O arquivo de projeto IRCE contem a localização de todos os outros arquivos necessários para simulação. Ele referencia um arquivo *world*, vários arquivos de configuração e vários arquivos de programa.

5.3 Configurar Robô e seus Sensores

Alguns modelos 3D podem ser guiados por um controle mas antes é configurá-lo para uso no Player. Isso é feito criando um arquivo de configuração para este modelo 3D. Tal requisito é suprido pelo menu **Configuration**, onde arquivos de configuração podem ser adicionados, alterados ou removidos.

Ao adicionar uma configuração, o usuário pode informar os seus dados de forma mais amigável conforme mostrado na Figura 7.

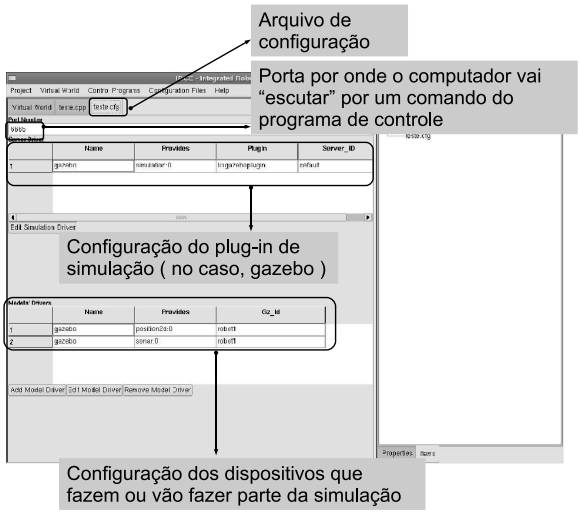

Figura 7: Painel de configuração de dispositivos

5.4 Programar Controle

Após adicionar e configurar um modelo 3D, podemos implementar um programa de controle para ele. Arquivos de programas em C++ podem ser

adicionados, removidos ou compilados no menu (**Control Programs**) implementando assim o requisito de programação de controle.

Ao executar (**Control Programs -> Add Program File**), o sistema requisita o caminho onde o arquivo do programa de controle com extensãp (*.cpp) deve ser gravado.

A compilação do programa é feita através do menu (**Control Programs -> Compile Program File**) e o resultado da compilação, incluindo erros, pode ser visto no painel abaixo, assim como em ambientes de desenvolvimento conforme mostrado na Figura 8.

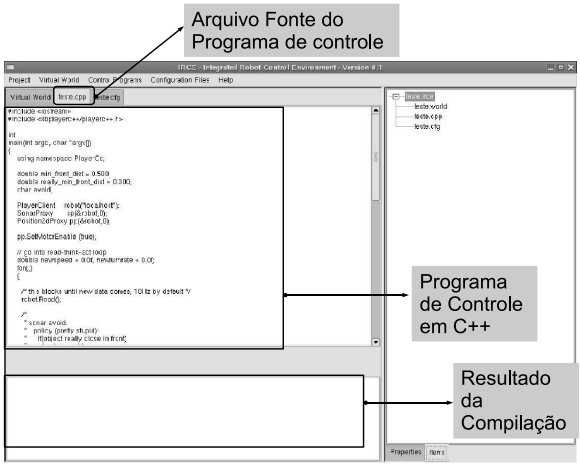

Figura 8: Painel para programação de controle de robôs

5.5 Simular

Finalmente podemos simular o resultado do programa de controle através do menu (**Project -> Simulation**). A partir daí, o ambiente passa o controle para o *Gazebo*.

Para interromper, basta clicar no botão **Ok** na janela de diálogo. Um exemplo de simulação é mostrado na Figura 9.

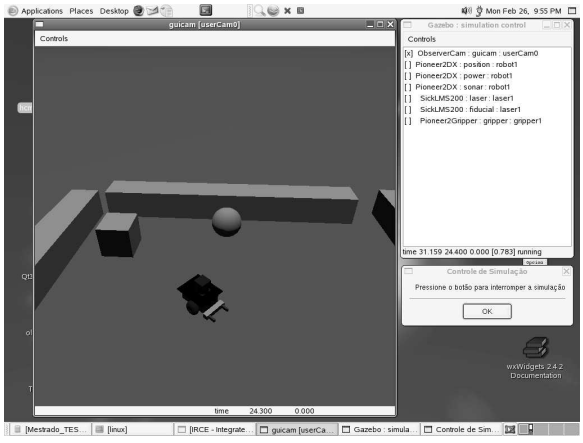

Figura 9: Simulação envolvendo um robô é executada no IRCE

6 Conclusões e Trabalhos Futuros

Esse artigo mostra um ambiente integrado de desenvolvimento de aplicações com robôs móveis. Este ambiente foi construído baseado no projeto P/S/G, que hoje é considerado o mais popular entre a comunidade de robótica que usa ferramentas de código-livre.

Os seguintes requisitos foram implementados no ambiente:

- Gerenciar Projeto IRCE
- Montar Mundo Virtual
- Configurar Robô e seus Sensores
- Programar Controle
- Simular

O protótipo do IRCE implementado é usável e pode ser utilizado pela comunidade robótica como um *software* de código-livre.

Embora o ambiente proposto tenha sido baseado nas ferramentas do projeto P/S/G, ele pode ser extendido para utilizar outros simuladores desde que estes possam ser executados numa linha-de-comando e que o formato do arquivo de entrada seja conhecido.

Por outro lado, as seguintes melhorias podem ser feitas na ferramenta proposta:

- Implementar os modelos 3D restantes suportados pelo Gazebo para serem utilizados nas simulações.
- Implementar a transferência de controle para os respectivos robôs reais.

Foi importante notar durante o desenvolvimento desse projeto como os projetos relacionados a robótica amadureceram rapidamente. O próprio projeto P/S/G também amadureceu bastante e conta com uma comunidade cada vez maior de usuários. Pode-se prever, num futuro muito breve, uma explosão na utilização de robôs móveis dos laboratórios para funções cada vez mais comuns no nosso dia-a-dia e essa ferramenta, assim como várias outras, pode contribuir para que isso aconteça.

Referências

Biggs, G. and MacDonald, B. (2003). A survey of robot programming systems, *Proceedings of the Australasian Conference on Robotics and Automation*, CSIRO, Brisbane, Australia.

Gerkey, B. P., Vaughan, R. T. and Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems, *ICAR 2003*, Coimbra, Portugal, pp. 317–323.

Gerkey, B. P., Vaughan, R. T., Stoy, K., Howard, A., Sukhatme, G. S. and Mataric, M. J. (2001). Most valuable player: a robot device server for distributed control, Vol. 3, pp. 1226–1231.

Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator, *Proceedings of Intelligent Robots and Systems*, Vol. 3, pp. 2149 – 2154.

MacDonald, B. A., Collett, T. H. J. and Gerkey, B. P. (2005). Player 2.0: Toward a practical robot programming framework, *Australasian Conference on Robotics and Automation (ACRA 2005)* .

Olivier Michel, C. L. (2004). Webots tm: Professional mobile robot simulation, *International Journal of Advanced Robotic Systems* **1**(1): 40–43.

The Player Project (2006). `http://playerstage.sourceforge.net`.

A Exemplo de configuração do Player para um robô Pioneer simulado no Gazebo

```
driver
(
  name "gazebo"
  provides ["simulation:0"]
  plugin "libgazeboplugin"
  server_id "default"
)

driver
(
  name "gazebo"
  provides ["position2d:0"]
  gz_id "robot1"
)

driver
(
  name "gazebo"
  provides ["sonar:0"]
  gz_id "robot1"
)
```

B Exemplo de programa de controle em C++ para desvio de obstáculos utilizando um sonar

```
#include <iostream>
#include <libplayerc++/playerc++.h>

int main(int argc, char *argv[]
{
  using namespace PlayerCc;

  // O cliente ouve a porta localhost
  PlayerClient    robot("localhost");

  // Instancia um sensor de sonar do Pioneer
  SonarProxy      sp(&robot,0);

  // Instancia o controle para as rodas do robo
  Position2dProxy pp(&robot,0);

  for(;;)
  {
    double turnrate, speed;

    // le dos proxies dos sensores
    robot.Read();

    // Utilizando dados dos sensores previne colisao
    if((sp[0] + sp[1]) < (sp[6] + sp[7]))
      turnrate = dtor(-20); // vira 20 graus por seg
    else
      turnrate = dtor(20);

    if(sp[3] < 0.500)
      speed = 0;
    else
      speed = 0.100;

    // Envia comandos às rodas do robo
    pp.SetSpeed(speed, turnrate);
  }
}
```

C Exemplo de descrição de um arquivo world

```
<?xml version="1.0"?>
<gz:world xmlns:gz="http://playerstage.sourcefor
ge.net/gazebo/xmlschema/#gz" xmlns:model="http:/
/playerstage.sourceforge.net/gazebo/xmlschema/#m
odel" xmlns:sensor="http://playerstage.sourcefor
ge.net/gazebo/xmlschema/#sensor" xmlns:window="h
ttp://playerstage.sourceforge.net/gazebo/xmlsche
ma/#window" xmlns:param="http://playerstage.sour
ceforge.net/gazebo/xmlschema/#params" xmlns:ui="
http://playerstage.sourceforge.net/gazebo/xmlsch
ema/#params">

  <model:ObserverCam>
    <id>userCam0</id>
    <xyz>5.637 93.859 3.053</xyz>
    <rpy>-0 13 -39</rpy>
    <imageSize>640 480</imageSize>
    <updateRate>10</updateRate>
    <renderMethod>GLX</renderMethod>
  </model:ObserverCam>

  <model:LightSource>
    <id>light1</id>
    <xyz>0.000 0.000 100.000</xyz>
  </model:LightSource>

  <model:GroundPlane>
    <id>ground1</id>
    <color>0.0 0.5 0.0</color>
  </model:GroundPlane>

  <model:Pioneer2DX>
    <id>robot1</id>
    <xyz>8.647 91.267 0.200</xyz>
  </model:Pioneer2DX>

</gz:world>
```