Guia Soccer Simulation 2D para Iniciantes Versão 1.1

Esse guia tem por objetivo ajudar os menos familiarizados com o RoboCup Soccer Simulator 2D e com o sistema operacional Linux a instalar rapidamente o ambiente de simulação e começar a desenvolver um time.

Caso ocorra algum problema ou caso você queira adicionar mais informações a esse guia, envie um e-mail para jackson@itandroids.com sendo o mais detalhista possível.

1- Instalando o Linux (sim, só roda em Linux):

Recomenda-se SUSE 10.0 em inglês com GNOME (deixar pelo menos 6 GB "livres" para a instalação do SUSE).

Pode fazer o download do SUSE 10 do ftp da UNICAMP:
ftp://ftp.unicamp.br/pub/linux/iso/suse/10/
Ou de um dos mirrors listados em:
http://www.novell.com/products/suselinux/downloads/ftp/mirrors_isos.html
Cuidado com a versão (32 ou 64 bits).

1.1 - Nas opções de intalação, adicionar:

Graphical Base System
GNOME System
C/C++ Compiler and Tools
GNOME Development
Java
E todos os componentes que tiverem Ruby, OpenGL e glut.

Caso não esteja seguro ou precise de ajuda para instalar o SUSE, consulte o Guia de Instalação SUSE, também disponível no site da ITANDROIDS:
http://www.itandroids.com/index.php?c=Users&a=mgu

Obs.: As instruções de instalação do Soccer Simulation 2D são para a utilização da conta root (sei que não é recomendado por motivos de segurança, mas fica mais fácil usando root). Se quiser escrever outro guia usando um usuário normal, ficaremos agradecidos.

2- Preparando diretórios:

Criar um diretório "rc" em /root/ (home do usuário root).
Places->Home Folder
Botão Direito -> Create Folder
Dentro do diretório "rc" criar um diretório "2d"
Dentro do diretório "2d" criar um diretório "fonte"

3 - Baixando Arquivos

Baixar os seguintes pacotes para o diretório /root/rc/2d/
rcssbase-10.0.11.tar.gz
rcssserver-10.0.7.tar.gz
manual.pdf
http://sourceforge.net/project/showfiles.php?group_id=24184

rcssmonitor-9.3.4-0.i386.rpm
http://www.ele.ita.br/~jackson/itandroids/files/rcssmonitor-9.3.4-0.i386.rpm

Baixar o fonte do UVATrilearn para /root/rc/2d/fonte/
http://staff.science.uva.nl/~jellekok/robocup/2003/trilearn_base_sources-3.3.tar.gz

4- Descompactando Pacotes

Decompacte os pacotes .tar.gz no próprio diretório onde eles estão (esqueça o .rpm por enquanto)

(botão direito -> Extract here)
Será criado um diretório para cada pacote:
/root/rc/2d/rcssbase-10.0.11/
/root/rc/2d/rcssserver-10.0.7/
e
/root/rc/2d/fonte/trilearn_base_sources-3.3/

5- Instalando rcssbase e rcssserver

Abra um terminal:
Applications->Sytem->Terminal->Gnome Terminal
Vá até o diretório do rcssbase:
(Observação, o # é a marca do prompt do Terminal, não é para ser digitado junto com o comando).

```
# cd rc/2d/rcssbase-10.0.11/
```

Configure, compile e instale o rcssbase (basta digitar esses comandos e esperar que cada um deles acabe)

```
# ./configure
# make
# make install
```

Agora vá até o diretório do rcssserver:

```
# cd ..
# cd rcssserver-10.0.7
```

Configure, compile e instale o rcssserver:

```
# ./configure
# make
# make install
```

6- Instalando rcssmonitor

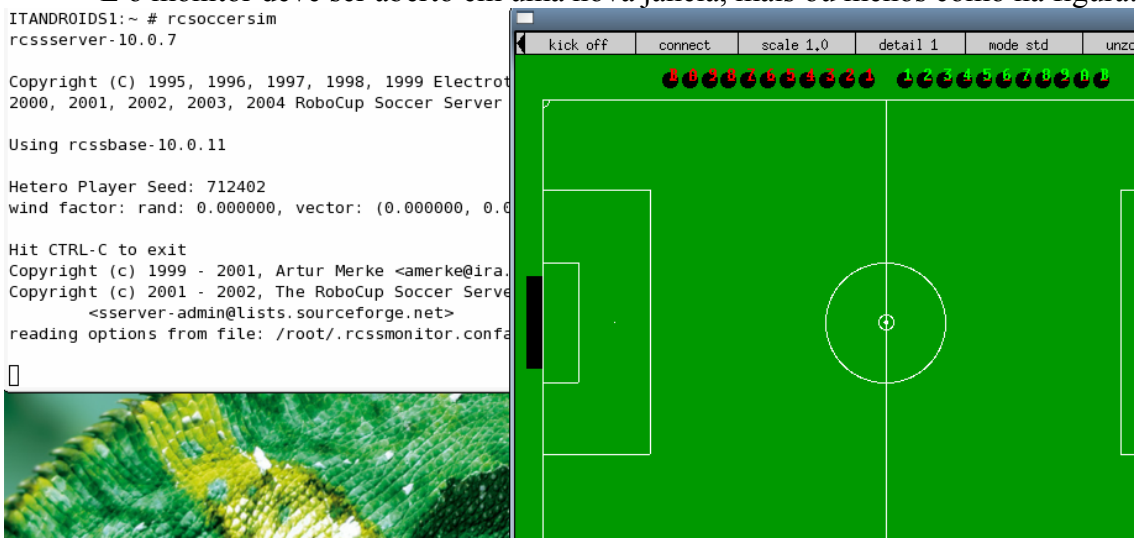
Agora dê um duplo clique no arquivo rcssmonitor-9.3.4-0.i386.rpm
O Yast deve aparecer, clique no botão <Install Packadge with YaST>
Aguarde um pouco e o monitor estará instalado.

7- Testando o servidor

Reinicie seu computador (é Linux também precisa reiniciar).
Abra um terminal:
Applications->Sytem->Terminal->Gnome Terminal
O servidor+monitor pode ser rodado do diretório root mesmo, basta digitar:
rcsoccersim
No prompt você deve ver uma mensagem como essa:

```
rcssserver-10.0.7

Copyright (C) 1995, 1996, 1997, 1998, 1999 Electrotechnical Laboratory.
2000, 2001, 2002, 2003, 2004 RoboCup Soccer Server Maintenance Group.

Using rcssbase-10.0.11

Hetero Player Seed: 712402
wind factor: rand: 0.000000, vector: (0.000000, 0.000000)

Hit CTRL-C to exit
Copyright (c) 1999 - 2001, Artur Merke <amerke@ira.uka.de>
Copyright (c) 2001 - 2002, The RoboCup Soccer Server Maintainance Group.
        <sserver-admin@lists.sourceforge.net>
reading options from file: /root/.rcssmonitor.confa new (v2) monitor connected
```

E o monitor deve ser aberto em uma nova janela, mais ou menos como na figura:



No terminal onde você rodou o servidor aperte <CTRL>+<C>. O servidor deve parar e o monitor deve ser fechar sozinho...

8- Compilando um time

Compilar o time base UVATrilearn é como compilar rcssbase e rcssserver, você só não precisa instalar.

Abra um terminal:
Applications->Sytem->Terminal->Gnome Terminal
Vá até o diretório do UVA:
```
# cd rc/2d/fonte/trilearn_base_sources-3.3/
```
Configure e compile o código (não precisa instalar)
```
# ./configure
# make
```

9- Vendo um jogo

Abra um terminal:
Applications->Sytem->Terminal->Gnome Terminal
Rode o servidor:
```
# rcsoccersim
```

Abra um novo terminal para o time 1:
Applications->Sytem->Terminal->Gnome Terminal
Vá até o diretório do time:
```
# cd rc/2d/fonte/trilearn_base_sources-3.3/
```
Execute o script que carrega o time:
```
# ./start.sh
```
Se você olhar no monitor verá o time "entrando em campo"

Abra um novo terminal para o time 2:
Applications->Sytem->Terminal->Gnome Terminal
Vá até o diretório do time:
```
# cd rc/2d/fonte/trilearn_base_sources-3.3/
```
Execute o script que carrega o time, mas usando um outro nome para o time
```
# ./start.sh localhost TESTE
```
Se você olhar no monitor verá o time "entrando em campo".

Depois dos dois times entrarem em campo, aperte o botão "kick off" do monitor e a partida começará.

Para iniciar o segundo tempo também é preciso apertar "kick off". Cada tempo dura 3000 ciclos

Para encerrar uma simulaçãoo basta apertar <CTRL>+<C> no terminal onde você rodou o servidor.

Caso não funcione direito (isso é, caso você tente iniciar uma nova simulação e o monitor abra a anterior), use o seguinte comando para "matar" a simulação:
```
# killall rcssserver
```

10- Programando seu próprio time

Agora você pode modificar o fonte o UVATrilearn para que ele não fique apenas chutando a bola na direção do gol, mas efetue passes, carregue a bola e drible os adversários.

Caso já não tenha seu software preferido para programar em Linux, recomendo que use o ANJUTA IDE. Ele já vem instalado no SUSE (se você marcou a opção "GNOME Development" conforme descrito anteriormente), mas a versão pré-instalada está com problemas. Felizmente atualizar para uma versão que funciona é fácil.

Se você já tem seu software para programação, pode ir direto para o item <10.3 – Entendendo melhor o fonte do UVATrilearn>

10.1 – Atualizando o Anjuta

Vamos atualizar apenas o Anjuta (se quiser atualizar todo o SUSE os procedimentos são um pouco diferentes, e pode demorar devido aos downloads).

Desktop -> YaST

Na janela do Yast (YaST Control Center @ SEUMICRO) clique em <Online Update>.

Na nova janela (YaST2@SEUMICRO) deixe marcado "Manually Select Patches" e clique em <Next>

O YaST vai pegar informações sobre os updates.

Na Lista que aparecer, deixar só o GNOME Development selecionado (pode deixar o resto selecionado também, mas vai demorar mais)

Apertar <Accept> e esperar

Talvez tenha que confirmar alguma coisa (licenças) durante a instalação.

Pronto, o Anjuta está atualizado e funcionando.

10.2 – Rodando o Anjuta e importando o código do UVATrilearn

Para rodar o Anjuta:

Applications->Development->Integrated Enviroment->Anjuta IDE

Na janela "Anjuta start with dialog" aperte o botão <Import Wizard>

Na janela "Project Import Wizard" aperte o botão <Forward>

Na tela seguinte (Select directory) aperte o botão <Browse> e vá até o diretório onde o fonte do UVATrilearn foi descompactado: (rc/2d/fonte/trilearn_base_sources-3.3/). Em Folders você deve ver:

./
../
src/
windows/

Aperte o botão <Ok>

De volta à janela "Project Import Wizard", tela "Select directory", aperte o botão <Forward>

Na tela seguinte (Project Type) o "Generic/Terminal project" deve estar selecionado, aperte o botão <Forward>

Na tela "Basic Information", complete o Project Author, mude as outras informações se quiser e altere o "Programmming language" para: "Both C and C++" e aperte o botão <Forward>

Na tela seguinte "Project Description" pode colocar uma descrição ou só apertar <Forward>

Na tela "Additional Options" sugiro desmarcar a opção "Include GNU Copyright statement in file headings" aperte <Forward>

Na tela Summary, apenas aperte <Apply>

Pronto, pode agora abrir os arquivos fonte do Anjuta e editá-los. Para gerar um novo binário, use a opção Build->Build All (ou Shift+F11).

10.3 – Entendendo melhor o fonte do UVATrilearn

As duas classes "responsáveis" pela atuação do jogador são WorldModel e Player (a implementação dessas classes está "espalhada" em alguns arquivos .cpp).

O WorldModel é responsável por manter uma imagem do mundo (campo, bola, jogadores) e usar essa imagem do mundo para responder perguntar feitas pelo player.

O player por sua ver é responsável por tomar as decisões e enviar os comandos para o simulador.

Abra o arquivo main (source src/main.cpp)

Ache a função int main(int argc, char * argv[]) //linha 70

Após verificação de argumentos e construção de objetos (lá no final da função) você vai achar a chamada para o Player: bp.mainLoop(); //linha 272

Um pouco acima //linha 242 você pode conferir que bp é da classe Player.

Marque mainLoop da linha 272 (um duplo clique em qualquer lugar da palavra mainLoop deve resolver).

Botão Direito do Mouse (na palavra mainLopp marcada) -> Go -> Tag Definition

O Anjuta vai abrir o arquivo src/Player.cpp e já colocar na definição da função void Player::mainLoop( ) //linha 108

Após alguma inicialização você encontrará o looping principal em: while(bContLoop) //linha 122

O if(WM->updateAll( ) == true) //linha 129 chama o WorldModel e manda que ele se atualize (pode usar o marcar, botão direito-> Go -> Tag Definition, para ver a função updateAll) O WorldModel após fazer vários controles chama funções mais específicas para atualizar as informações sobre o próprio jogador, a bola e os outros jogadores.

Seguindo na função void Player::mainLoop( ), você vai encontrar um switch(formations->getPlayerType( )) //linha 139

Dependendo do tipo do jogador ele irá chamar diferentes funções para decidir qual a sua ação/comando (SOC)

Olhando mais abaixo no código (ou usando o Go-> Tag Definition) você verá que as funções defenderMainLoop( ), midfielderMainLoop( ) e attackerMainLoop( ) chamam uma outra função deMeer5(), enquanto que a função goalieMainLoop( ) chama uma outra função deMeer5_goalie().

Marque um dos deMeer5, botão direito, Go -> Tag Definition

O Anjuta vai abrir o arquivo src/PlayerTeams.cpp na função SoccerCommand Player::deMeer5( ) //linha 60

É aqui que o jogador "pensa" e "decide" o que vai fazer. Resumindo, o jogador faz várias perguntar para o WorldModel (WM->isBeforeKickOff( ), WM->isKickOffUs( ), WM->getPlayerNumber(), WM->isBallKickable(), WM->getStrategicPosition()) e de acordo com a resposta, opta por uma ou outra decisão, chamando funções específicas para "gerar" o comando correspondente à decisão.

Por exemplo:

soc = kickTo(posGoal, SS->getBallSpeedMax()); // kick maximal //linha 115

O jogador tomou a decisão de chutar para o gol e chama a função kickTo para gerar o comando correpondente ao chute para o gol com a máxima velocidade possível.

A idéia do deMeer5_goalie é parecida, só que as perguntas para o WorldModel, as decisões e os comandos são um pouco diferentes.

Na verdade as coisas são um pouquinho mais complicadas que isso, mas já é um excelente começo.

Tente alterar algumas coisas no deMeer5 e no deMeer5_goalie para começar (mudando o critério de decisão ou as ações tomadas), depois imagine novas situações e novas decisões (provavemente você vai ter que criar novas funções no WorldModel para ajudar na tomada de decisão).

Daqui pra frente é com você.

11- Pegando outros times para enfrentar

Binários do Mundial 2006 disponíveis em:

http://ssil.uni-koblenz.de/RC06/2D/Bilogs 2005 2d soccernaries/

Para executar os binários de outros times pode-se montar um script para rodá-lo ou pode tentar alterar os já existentes. Muitos times têm um script que já roda todos os jogadores no server. Tente arquivos de script com nomes diferentes dos padrões “start”, “start1”, “start2” ou “start3”. Por exemplo, para o Brainstormers (disponivel no link indicado) basta usar o script “start_team.sh” que esta na pasta robocup. Ou para o time da tokyotech basta usar o script “start.sh”.

Se estiver usando o linux no modo gráfico do Gnome basta clicar no arquivo e selecionar a opção executar. Caso não se abra essa opção, pode-se usar o terminal, estando na pasta onde está o script basta usar:

#sh script.sh

Onde script.sh é o arquivo alvo.

12- Vendo logs

Alguns links para baixar logs de jogos:

Logs da Competição Brasileira de 2006 disponíveis em:

http://jri.sorocaba.unesp.br/logs/

Logs do Mundial de 2006 disponíveis em:

http://ssil.uni-koblenz.de/RC06/2D/Logs/

Além do logplayer oficial disponibilizado pela Robocup, já se têm disponíveis diversos outros criados por equipes participantes da Robocup. Alguns destes têm certas

funcionalidades a mais para a análise do jogo. Para isto deve-se baixar arquivos .rcg disponíveis nos sites das competicões.

Para ver logs de jogos no Windows XP pode-se optar pelo logalyzer 0.5:
http://dis.ijs.si/andraz/logalyzer/

Para usá-lo, basta baixar o arquivo zip disponível no site, descompactá-lo e então e só clicar no executável. Com o programa aberto escolhe-se a opção open file e abrir o arquivo do jogo escolhido.

Para ver os jogos no linux mesmo, uma boa opção é o TeamAssistant 2003:
http://www.sbcee.net/pres/download.htm

Para este programa, basta baixar a versão para linux, descompactar e já esta pronto para usar, bastando executar o arquivo teamasist. No programa, no menu para Robocup->Open Log e escolhe-se o arquivo.

Para maiores informações sobre os players basta procurar no site por guias ou descrições.

13- Convertendo logs para .swf

Outra opção bastante utilizada, principalmente se é desejável exibir os logs em páginas web é passá-los para o formato flash(.swf). Um dos programas para este trabalho é o robocup2flash, que pode ser baixado em:

http://sourceforge.net/projects/robolog/

Para converter o arquivo .rcg para .swf basta, digitar no terminal:

```
#robocup2flash arquivo.rcg arquivoFlash.swf
```

Onde em arquivo.rcg coloca-se o nome do arquivo a ser convertido e em arquivoFlah.swf o nome do arquivo a ser obtido. Caso não se coloque o nome do arquivo flash o programa irá criar um arquivo de mesmo nome com a extensão diferente.

14- Informações finais

Se der alguma coisa errada, ou se achar que alguma outra informação deve ser adicionada a esse guia, não esqueça de me mandar um e-mail com o máximo de detalhes possível para jackson@itandroids.com.

Esse guia de instalação foi escrito pela Equipe de Competições de Robótica do ITA – ITANDROIDS, com o objetivo de ajudar no desenvolvimento de novas equipes para Soccer Simulation 2D.

Sua distribuição é livre desde que não seja alterado.

A versão atualizada desse guia está disponível no site da ITANDROIDS:
http://www.itandroids.com/index.php?c=Users&a=mgu

Versão 1.1, 23/08/2006.