8
Referências bibliográficas

[1] IAGNEMMA, K., DUBOWSKY, S. **Mobile Robots in Rough Terrain: Estimation, Motion Planning, and Control with Application to Planetary Rovers**. New York: Springer, 2004.

[2] IAGNEMMA, K., DUBOWSKY, S. **Traction Control of Wheeled Robotic Vehicles in Rough Terrain with Application to Planetary Rovers**. Massachusetts Institute of Technology, Department of Mechanical Engineering, Cambridge, USA, 2004.

[3] LAMON, P., SIEGWART, R. **Wheel Torque Control in Rough Terrain - Modeling and Simulation**. Proceedings of the IEEE International Conference on Robotics and Automation, Barcelona, 2005.

[4] SILVA, Alexandre F. Barral. **Modelagem de Sistemas Robóticos Móveis para Controle de Tração em Terrenos Acidentados**. Pontifícia Universidade Católica do Rio de Janeiro – PUC-RIO, Departamento de Engenharia Mecânica, abril de 2007.

[5] SANTOS, Auderi V. **Controle de Capotagem e Deslizamento de Sistemas Robóticos Móveis em Terrenos Acidentados**. Pontifícia Universidade Católica do Rio de Janeiro – PUC-RIO, Departamento de Engenharia Mecânica, maio de 2007.

[6] CAMPION, G., CHUNG, W. **Wheeled Robots**. Springer Handbook of Robotics Siciliano, Khatib (Ed.). Nº 17, Págs. 391-410, Springer, 2008.

[7] BURGARD, W., HEBERT, M. **World Modeling**. Springer Handbook of Robotics Siciliano, Khatib (Eds.). Nº 36, Págs. 391-410, Springer, 2008.

[8] CHAKRABORTY, N., GHOSAL, A. **Kinematics of Wheeled Mobile Robots on Uneven Terrain**. Department of Mechanical Engineering, Indian Institute of Science, maio de 2004.

[9] SWEET, A. (University of California, Berkeley), BLACKMON, T. (NASA Ames Research Center), GUPTA, V. (NASA Ames Research Center). **Simulation of a Rover and Display in a Virtual Environment**. University of California, Berkeley, 1999.

[10] CLARAty, NASA.
Disponível em <http://claraty.jpl.nasa.gov/man/overview/index.php>.
Acesso em 05 de agosto de 2010.

[11] Rover Graphical Simulator (RGS), NASA.
Disponível em <http://www.techbriefs.com/content/view/1782/34/>.
Acesso em 05 de agosto de 2010.

[12] Universal Mechanisms, Laboratory of Computational Mechanics Bryansk State Technical University, Rússia.
Disponível em <http://www.umlab.ru/>. Acesso em 05 de agosto de 2010.

[13] GNU General Public License.
Disponível em <http://www.gnu.org/licenses/licenses.html#GPL>.
Acesso em 05 de agosto de 2010.

[14] NASA Tech Briefs. **Rover Graphical Simulator - Technical Support Package**. NPO-35223. NASA's Jet Propulsion Laboratory, Pasadena, California.

[15] GAZEBO - 3D Multiple Robot Simulator with Dynamics.
Disponível em <http://playerstage.sourceforge.net/index.php?src=gazebo>.
Acesso em 05 de agosto de 2010.

[16] BAUER, R., BARFOOT, T., LEUNG, W., RAVINDRAN, G. **Dynamic Simulation Tool Development for Planetary Rovers**. Jet Propulsion Laboratory, California Institute of Technology, Pasadena, Califórnia 91109.

International Journal of Advanced Robotic Systems, Vol. 5, Nº 3, ISSN 1729-8806, Págs. 311-314, 2008.

[17] Design-Simulation. Working Model 2D.
Disponível em <http://www.design-simulation.com/wm2d/index.php>.
Acesso em 05 de agosto de 2010.

[18] THUEER, T., KREBS, A., SIEGWART, R., LAMON, P. **Performance Comparison of Rough-Terrain Robots – Simulation and Hardware**. Journal of Field Robotics - Special Issue on Space Robotics, Vol. 24, Ed. 3, Págs. 251 – 271, março de 2007.

[19] Autonomous System Labs. CRAB Rover.
Disponível em <http://www.asl.ethz.ch/robots/>. Acesso em 20 de agosto de 2010.

[20] LINDEMANN, R.A., BICKLER, D.B., HARRINGTON, B.D., ORITZ, G.M., VOORHEES, C.J. **Mars Exploration Rover Mobility Development - Mechanical Mobility Design**, Development, and Testing. Robotics & Automation Magazine, IEEE, Vol. 13, Págs. 19-26, 2006.

[21] Russell Smith , Open Dynamics Engine.
Disponível em <http://ode.org/>. Acesso em 05 de agosto de 2010.

[22] Adams Multibody Dynamics.
Disponível em <http://www.mscsoftware.com/Contents/Products/CAE-Tools/Adams.aspx>. Acesso em 20 de agosto de 2010.

[23] Robotics Research Group, The University of Texas, Austin. Disponível em <http://www.robotics.utexas.edu/rrg/>. Acesso em 20 de agosto de 2010.

[24] DUDEK, G., JENKIN, M. **Computational Principles of Mobile Robotics**. United Kingdom, Cambridge, 2000.

[25] ROAMS (Rover Analysis, Modeling and Simulation). Disponível em <http://dshell.jpl.nasa.gov/ROAMS/index.php>. Acesso em 05 de agosto de 2010.

[26] ESTLIN, T., YEN, J., PETRAS, R., MUTZ, D., CASTAÑO, R., RABIDEAU, G., STEELE, R., JAIN, A., CHIEN, S., MJOLSNESS, E., GRAY, A., MANN, T., HAYATI, S., DAS, H. **An Integrated Architecture for Cooperating Rovers**. Artificial Intelligence, Robotics and Automation in Space, Proceedings of the Fifth International Symposium, ISAIRAS '99, held 1-3 June, 1999 in ESTEC, Noordwijk, the Netherlands. Edited by M. Perry. ESA SP-440. Paris: European Space Agency, Págs. 255-262, 1999

[27] VOLPE, R., NESNAS, I., ESTLIN, T., MUTZ, D., PETRAS, R., DAS, H. **The CLARAty Architecture for Robotic Autonomy**. Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109.

[28] NESNAS, R.I.A.D., SIMMONS, R., GAINES, D., KUNZ, C., DIAZ-CALDERON, A., ESTLIN, T., MADISON, R., GUINEAU, J., MCHENRY, M., SHUL, I., APFELBAUM, D. **CLARAty: Challenges and Steps Toward Reusable Robotic Software**. Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109. International Journal of Advanced Robotic Systems, Vol. 3, No. 1, ISSN 1729-8806, Págs. 23-30, 2006

[29] NESNAS, I.A.D., WRIGH, A., BAJRACHARYA, M., SIMMONS, R., ESTLIN, T., KIM, W.S. **CLARAty: An Architecture for Reusable Robotic Software**. Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109.

[30] NESNAS, I.A.D. **CLARAty: A Collaborative Software for Advancing Robotic Technologies**.

[31] USC Robotics Research Lab. **Player, Stage and Gazebo**. Disponível em <http://www-robotics.usc.edu/?l=projects:playerstagegazebo>. Acesso em 05 de agosto de 2010.

[32] NVidia White Paper. **Accelerating MATLAB with CUDA™ Using MEX Files**. WP-03495-001_v01, setembro de 2007.

[33] RDS - Microsoft Robotics Developer Studio 2008. Disponível em <http://msdn.microsoft.com/en-us/library/bb483024.aspx>. Acesso em 05 de agosto de 2010.

[34] PhysX AGEIA Technologies, Inc. . Disponível em Disponível em: <http://www.nvidia.com/object/physx_new.html>. Acesso em 05 de agosto de 2010.

[35] Microsoft DirectX. Disponível em <http://www.microsoft.com/games/en-US/aboutGFW/pages/directx.aspx>. Acesso em 05 de agosto de 2010.

[36] JAIN, A., BALARAM, J., CAMERON, J., GUINEAU, J., LIM, C., POMERANTZ, M., SOHL, G. **Recent Developments in the ROAMS Planetary Rover Simulation Environment**. Jet Propulsion Laboratory, California Institute of Technology.

[37] CLARAty Movies, NASA. Disponível em <http://claraty.jpl.nasa.gov/man/overview/movies/index.php>. Acesso em 05 de agosto de 2010.

[38] KREBS, A., THUEER, T., CARRASCO, E., SIEGWART, R. **Towards Torque Control of the CRAB Rover**. Autonomous Systems Lab, ETH Zurich.

[39] CLARAty Download, NASA. Disponível em <http://claraty.jpl.nasa.gov/man/software/download/index.php>. Acesso em 05 de agosto de 2010.

[40] CLARAty Public License, NASA. Disponível em <http://claraty.jpl.nasa.gov/man/software/license/public_src/index.php>. Acesso em 05 de agosto de 2010.

[41] SMITH, R. Open Dynamics Engine v0.5 User Guide, fevereiro de 2006. Disponível em <http://ode.org/ode-latest-userguide.html>. Acesso em 05 de agosto de 2010.

[42] SIGGRAPH. **Performance OpenGL: Platform Independent Techniques**. Course # 37, 2002.

[43] Khronos Group, OpenGL: Overview. São Francisco. Disponível em <http://www.opengl.org/about/overview/>. Acesso em 05 de agosto de 2010.

[44] Autodesk, Alias|Wavefront Maya. Disponível em <http://usa.autodesk.com/adsk/servlet/pc/index?siteID=123112&id=13577897>. Acesso em 05 de agosto de 2010.

[45] Amsterdam Blender Institute, Blender, Amsterdam. Disponível em <http://www.blender.org/>. Acesso em 05 de agosto de 2010.

[46] OMG, UML - Unified Modeling Language Specification, Versão 1.5, formal/03-03-01, março de 2003.

[47] SIMPACK AG. Disponível em <http://www.simpack.com/home-simpack.html>. Acesso em 05 de agosto de 2010.

[48] RioBotz COMBAT TUTORIAL, Versão 2.0, março 2009.

[49] MADDEN, J.D. **Mobile Robots: Motor Challenges and Materials Solutions**. Science, Vol. 318, 16 de novembro de 2007.

[50] PAPADOPOULOS, E., REY, D. **The Force-Angle Measure of Tipover Stability Margin for Mobile Manipulators**. Vehicle System Dynamics, Vol. 33, 1ª Edição, Págs. 29-48, janeiro de 2000.

[51] PETERS, S.C., IAGNEMMA, K. **An Analysis of Rollover Stability Measurement for High-Speed Mobile Robots**. Proceedings of 2006 IEEE International Conference on Robotics and Automation, Orlando, Flórida, Págs. 3711–3716, maio de 2006.

[52] PETERS, S.C., IAGNEMMA, K. **Stability Measurement of High-Speed Vehicles**. Vehicle System Dynamics, Vol. 47, 6ª Edição, Págs. 701-720, junho de 2009.

[53] PACEJKA, H.B. **Tire and Vehicle Dynamics**. SAE – Society of Automotive Engineers, Inc., 2ª Edição, Págs. 156-215, 2006.

[54] WONG, J.Y. **Theory of Ground Vehicles**. John Wiley & Sons, Inc., 4ª Edição, 2008.

[55] SOBCZYK, S., MARIO, R., PERONDI, E.A., CUNHA, M.A.B. **A Continuous Approximation of the LuGre Friction Model**. 20th International Congress Of Mechanical Engineering – COBEM 2009, Gramado, RS, Brazil, 15 e 20 de november de 2009.

[56] SILVA, A.F.B., SANTOS, A.V., MEGGIOLARO, M.A., Neto, M.S. **A Rough Terrain Traction Control Technique for All-Wheel-Drive Mobile Robots**. ABCM, 2010.

[57] The DARTS Simulation Laboratory, NASA. Disponível em <http://www-robotics.jpl.nasa.gov/facilities/facility.cfm?Facility=7>. Acesso em 05 de agosto de 2010.

[58] AESCO GbR Automotive Engineering, Software & Consulting. Disponível em < http://www.english.aesco.de/>. Acesso em 05 de agosto de 2010.

[59] GLUT. **The OpenGL Utility Toolkit**. OpenGL. Disponível em < http://www.opengl.org/resources/libraries/glut/ >. Acesso em 05 de agosto de 2010.

[60] KOENIG, N. **Stage and Gazebo - The Instant Expert's Guide**. USC Robotics Research Labs, 15 de setembro de 2004.

[61] CANUDAS DE WIT, C., OLSSON, H., ASTROM, K.J., LISCHINSKY, P. **A New Model for Control Systems with Friction**. IEEE Transactions on Automatic Control, Vol. 40, Nº 3, Págs. 419-425, 1995

[62] JAZAR, NAKHAIE, G. **Vehicle Dynamics: Theory and Application**. Springer, 2008.

Anexo A:
Organização dos Arquivos do Aplicativo VirtualBotz 3D

O aplicativo está organizado em pastas (ou diretórios) para facilitar a compreensão de suas partes. Este **Anexo** mostra apenas uma visão abrangente de sua organização. Para maiores detalhes, consulte os outros anexos.

Tabela 9 – Pastas do aplicativo

Pasta	Descrição
VirtualBotz_Version_2.000.5/	
_library/	pasta com as bibliotecas em c++
Botz/	pasta com as classes de negócio
Graph/	pasta com as classes para recursos gráficos (classes de negócio)
Math/	pasta com as classes para operações vetoriais e matriciais (classes de negócio)
OpenGL/	pasta com as classes de visualização em OpenGL
Tools/	pasta com as bibliotecas de uso geral
Build/	
Heightmaps/	pasta com os arquivos de Mapas de Alturas para construção de terrenos
Obj_files/	pasta com os arquivos de objetos 3D e de materiais para rodas e veículos
Report/	pasta com os arquivos de dados gerados pelo aplicativo durante a simulação
Scripts/	pasta com os arquivos de configuração das simulações do aplicativo
Tests/	pasta com os arquivos de testes para comparações de resultados
Textures/	pasta com os arquivos de texturas do terreno

Videos/	pasta com os arquivos de vídeo gerados pelo aplicativo durante as simulações
Projects/	
VS6/	pasta com o projeto para a versão em Microsoft Visual Studio *6.0*
VS9/	pasta com o projeto para versão em Microsoft Visual Studio *9.0*

Anexo B:
Diagrama de Classes (UML) do Simulador

As classes do simulador estão organizadas nas seguintes pastas, nos respectivos arquivos de código mostrados na tabela a seguir:

Tabela 10 – Localização e descrição de todas as bibliotecas do aplicativo

Localização em pastas	Descrição
_library/	
typedefs.h	biblioteca para declarações / definições de tipos globais usados no sistema
globalvars.h *globalvars.cpp*	variáveis globais
main.cpp	programa principal
Graph/	
CVRImage.h *CVRImage.cpp*	biblioteca gráfica para leitura de bitmap padrão e OpenGL
Math/	
CMtx33.h *CMtx33.cpp*	biblioteca de matemática para manipulação de matriz 3x3
CVec3.h *CVec3.cpp*	biblioteca de matemática para manipulação de vetor de três posições
Botz/	
CVRAvi.h *CVRAvi.cpp*	biblioteca para Audio Video Interleave
CVRControl.h *CVRControl.cpp*	biblioteca responsável pelo controle do veículo robótico
CVRCorner.h *CVRCorner.cpp*	biblioteca base para criação do veículo robótico

CVRMobileRobot.h *CVRMobileRobot.cpp*	biblioteca responsável pela criação do veículo robótico (chassi + rodas)
CVRMotor.h *CVRMotor.cpp*	biblioteca responsável pela criação dos motores
CVRMotorDataSheet.h *CVRMotorDataSheet.cpp*	biblioteca para criação da folha de dados dos motores
CNumericalMethod.h *CNumericalMethod.cpp*	biblioteca para o método de integração do sistema
CVRState.h *CVRState.cpp*	biblioteca de histórico dos estados do sistema
CVRTerrain_.h *CVRTerrain_.cpp*	biblioteca para criação de terrenos
CVRThread.h *CVRThread.cpp*	biblioteca para manipulação de processos
CVRToolsFiles.h *CVRToolsFiles.cpp*	biblioteca responsável pela leitura dos scripts do aplicativo
CVRWheel.h *CVRWheel.cpp*	biblioteca para criação das rodas do veículo robótico
OpenGL/	
COGLComponent.h *COGLComponent.cpp*	biblioteca para superclasse “componente” OpenGL
COGLTerrain.h *COGLTerrain.cpp*	biblioteca OpenGL para o terreno
COGLObject.h *COGLObject.cpp*	biblioteca OpenGL para objetos 3D
COGLTexture.h *COGLTexture.cpp*	biblioteca OpenGL para texturas
COGLWheel.h *COGLWheel.cpp*	biblioteca OpenGL para rodas
COGLMobileRobot.h *COGLMobileRobot.cpp*	biblioteca OpenGL para veículos robóticos

OGLSettings.h *OGLSettings.cpp*	biblioteca de configuração para visualização em OpenGL
Tools*/*	
ToolsArrays.h	biblioteca para manipulação de vetores
ToolsRandom.h *TooslRandom.cpp*	biblioteca para obtenção de números aleatórios

- Classes relacionadas com o terreno

A Figura 61 mostra o diagrama de classes em UML em duas camadas da modelagem do terreno.

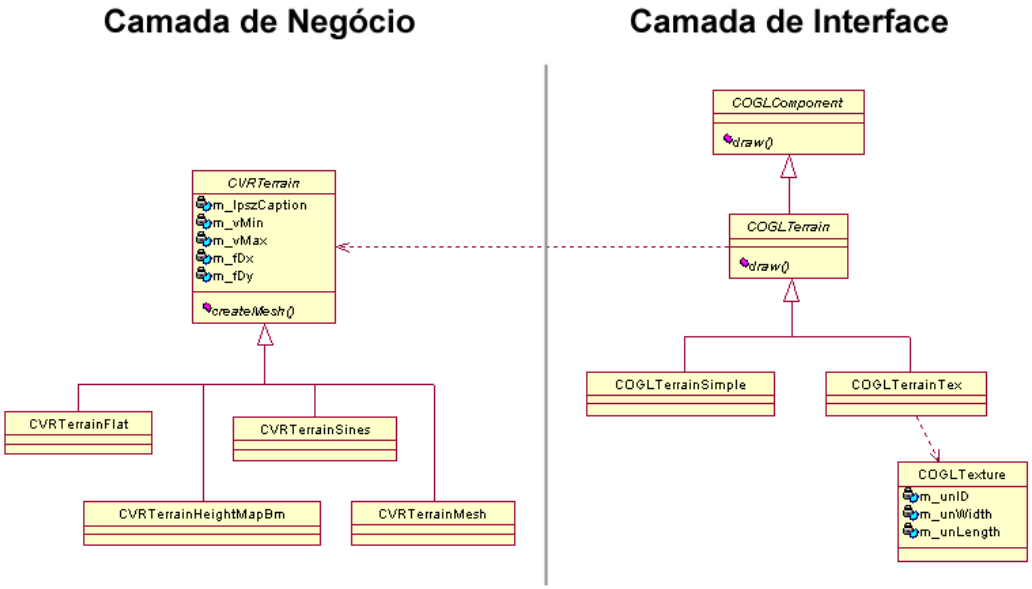


Figura 61 – Diagrama de classes (UML) do terreno

- Classes relacionadas com o veículo robótico

A Figura 62 mostra o diagrama de classes em UML em duas camadas da modelagem do veículo robótico.

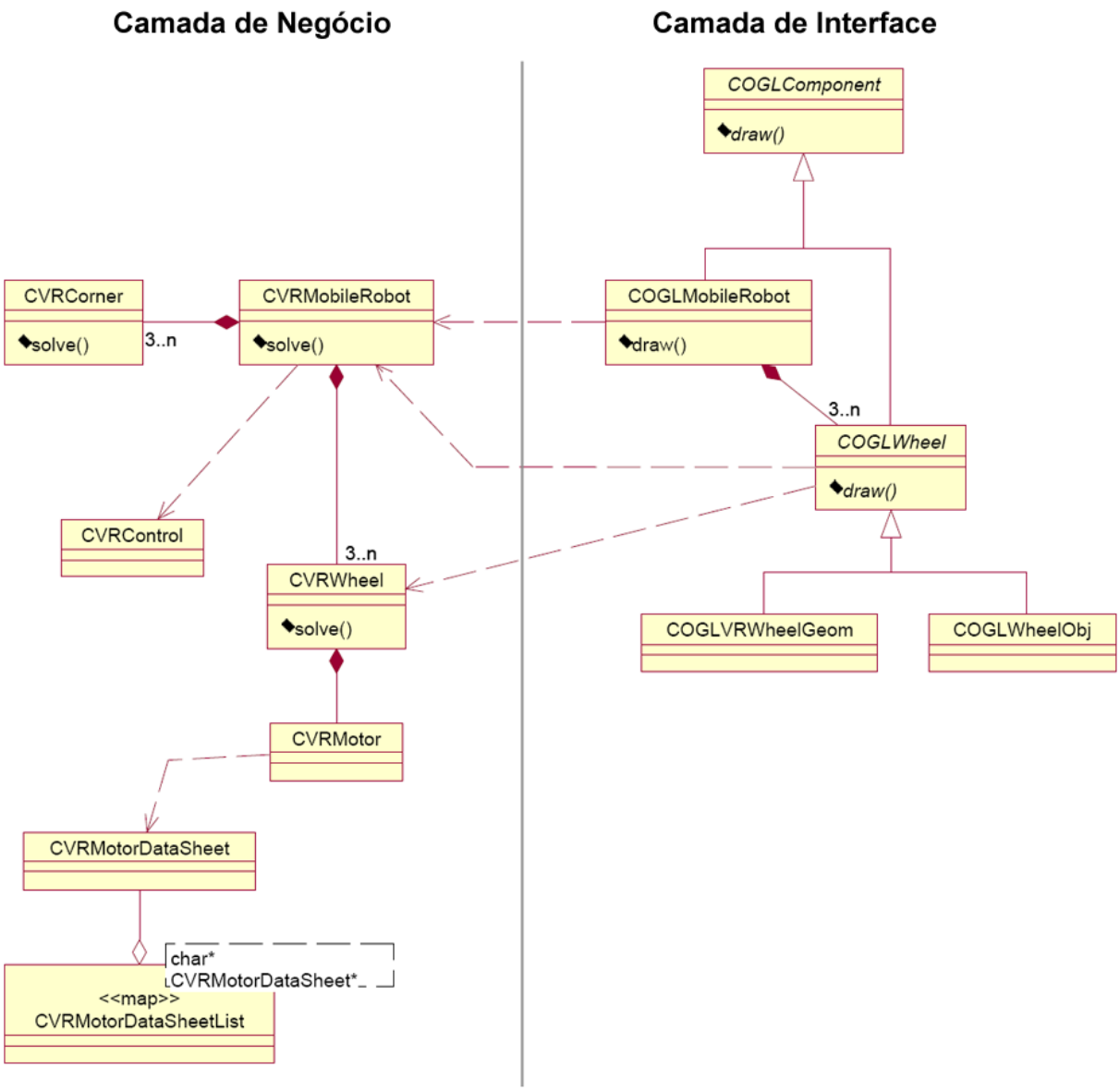


Figura 62 – Diagrama de classes (UML) do veículo robótico

- Classes extras

A Figura 63 mostra o diagrama de classes em UML da modelagem para as classes de operações matriciais.

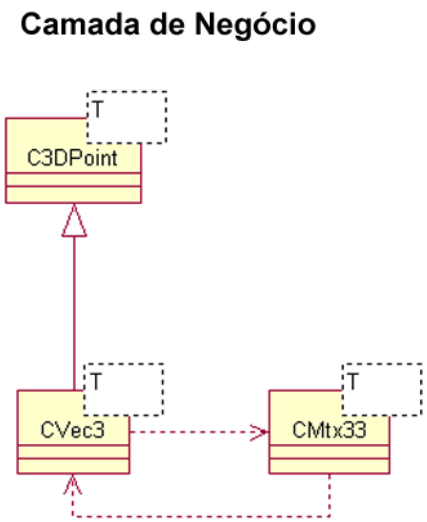


Figura 63 – Diagrama de classes (UML) para matrizes 3x3 e vetores 3x1/1x3

Anexo C:
Comandos do Joystick (Analógico)

O simulador deve funcionar com qualquer tipo de joystick analógico. Para isso, devem ser instalados os drivers que acompanham o próprio joystick. Os joysticks com dois *pads* (ou manetes) de comandos (Figura 64) seguem a seguinte configuração: os comandos do *pad* da esquerda são para o veículo robótico, e os comandos do *pad* da direita, para o mundo virtual, com as opções rotação, translação e *zoom*, selecionadas uma por vez ao pressionar o botão de comando.

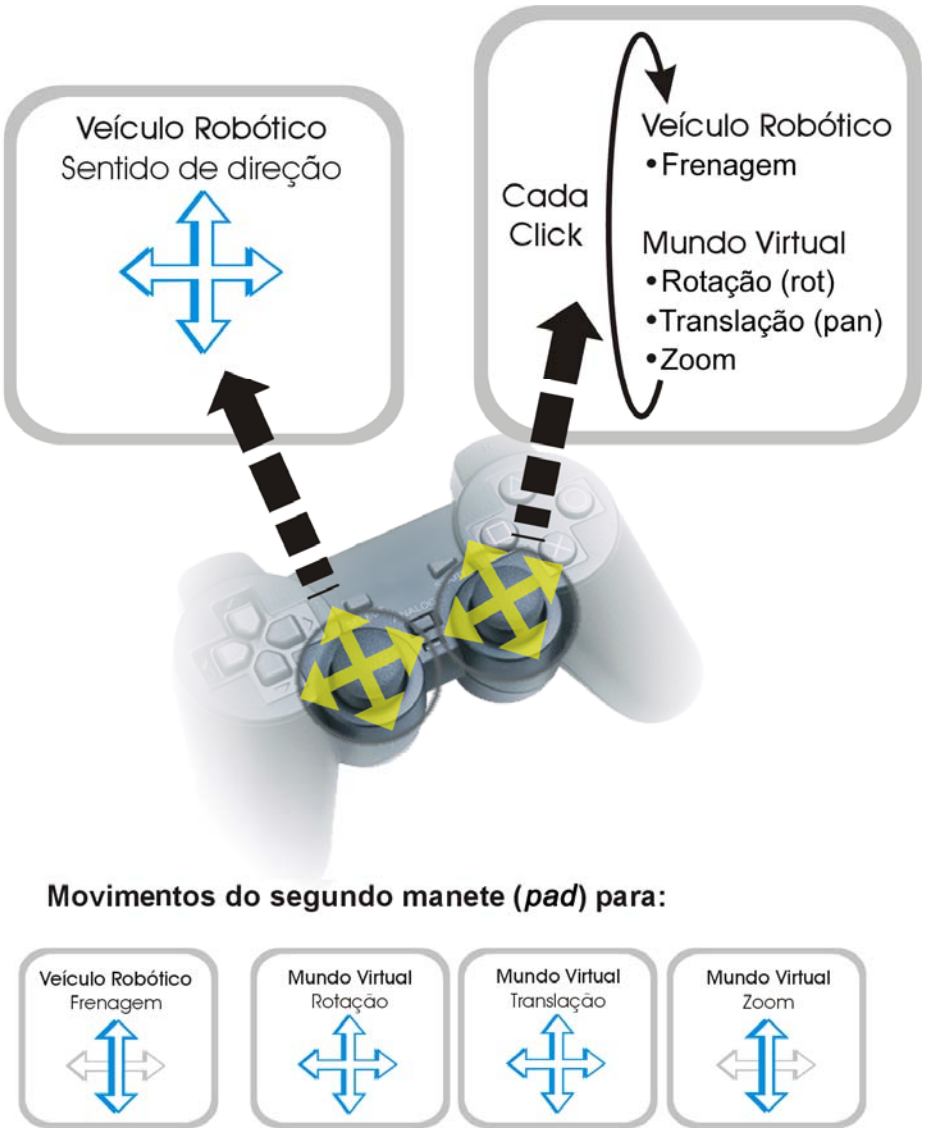


Figura 64 – Comandos dos manetes do joystick

O simulador também utiliza os quatro botões da direita do joystick analógico (Figura 65) após a parada de uma simulação (tecla "**P**"). Eles servem para avançar ou recuar o veículo robótico para as posições do centro de massa guardado no histórico da simulação. Essa operação pode também ser realizada através do teclado (ver item "**D.8. Posicionar o veículo robótico pelo centro de massa ao longo da trajetória após a simulação**" do **Anexo D**).

Ao pressionar uma vez o botão 1 (com o triângulo verde), a movimentação de recuo ou avanço do veículo robótico é feita automaticamente. Para o recuo do veículo robótico, deve-se pressionar o botão 4 (com o quadrado rosa) e, para o avanço, o botão 2 (com o círculo vermelho).

Figura 65 – Comandos dos botões do joystick

Anexo D:
Teclas de Funcionalidades do Simulador

O simulador tem algumas teclas de funcionalidades que estão em destaque na Figura 66.

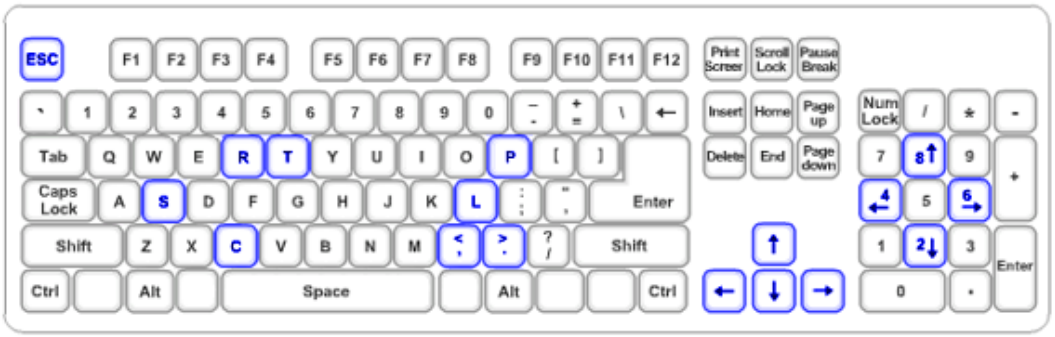


Figura 66 – Teclas de funcionalidades do simulador (teclado padrão internacional)

D.1. Terminar execução do simulador

A tecla "**Esc**" termina a execução do aplicativo, após salvar todos os arquivos, liberar todos os recursos de memória e inclusive restaurar o VSync (sincronização vertical da placa de vídeo), se estiver sendo utilizado.

D.2. Alternar visão da simulação pelas câmeras interna e externas do veículo robótico

A tecla "**C**" (ou "**c**") alterna entre os três tipos de câmeras existentes no aplicativo, que são as seguintes: 1) observador fixo em um ponto no espaço, que, porém, permite rotações e translações do mundo virtual com o mouse a fim de permitir uma visão panorâmica da área de teste (Figura 67); 2) câmera interna posicionada no interior do veículo robótico (Figura 68), cujo observador se movimenta conforme a dinâmica do veículo robótico, inclusive nas capotagens; 3) câmera posicionada externamente ao veículo a uma distância fixa (Figura 69).

Para a segunda e a terceira câmeras, não é permitida a utilização do mouse para manipulação do mundo virtual.

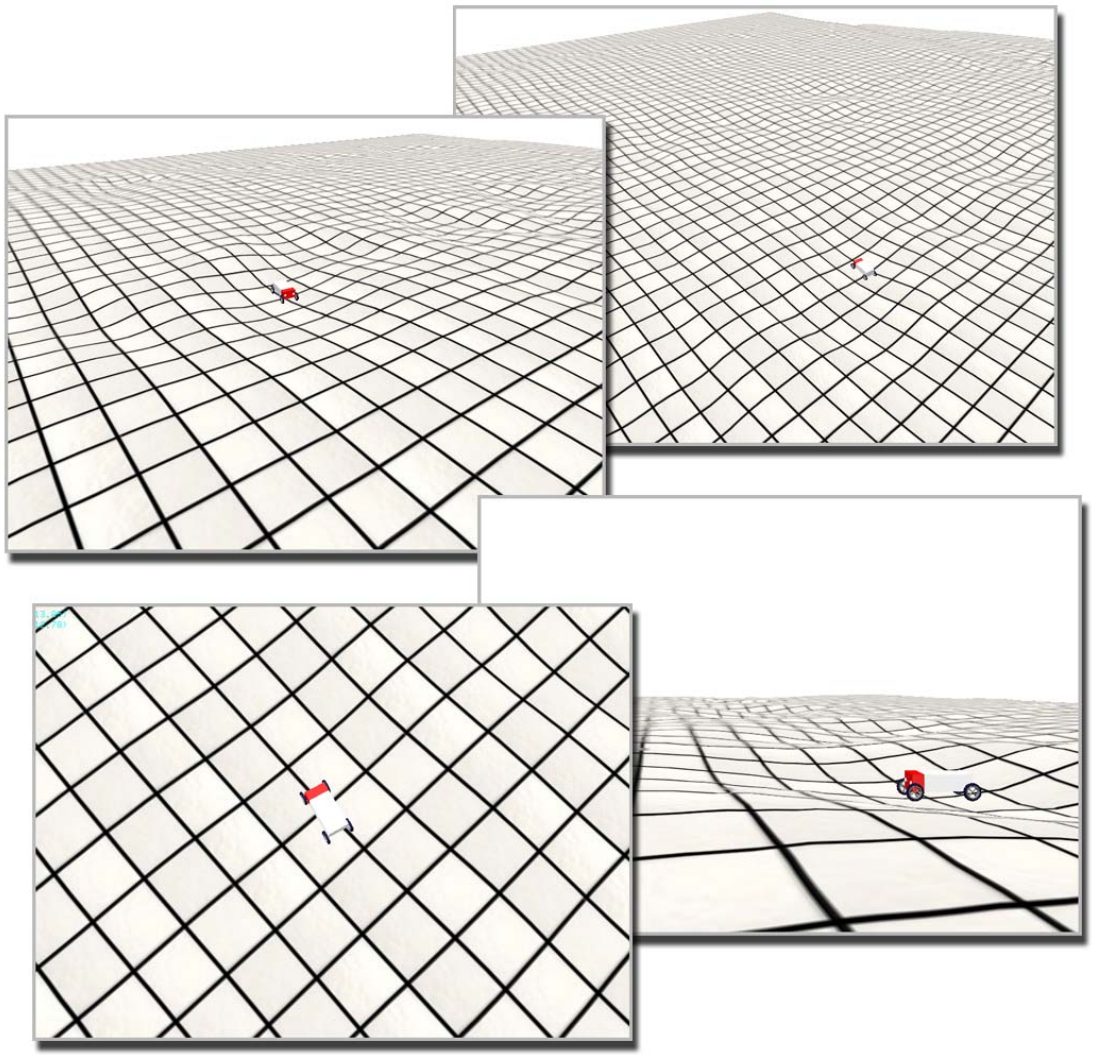


Figura 67 – Veículo robótico observado pela câmera virtual 1, permitindo a movimentação do mundo virtual de forma independente do veículo robótico

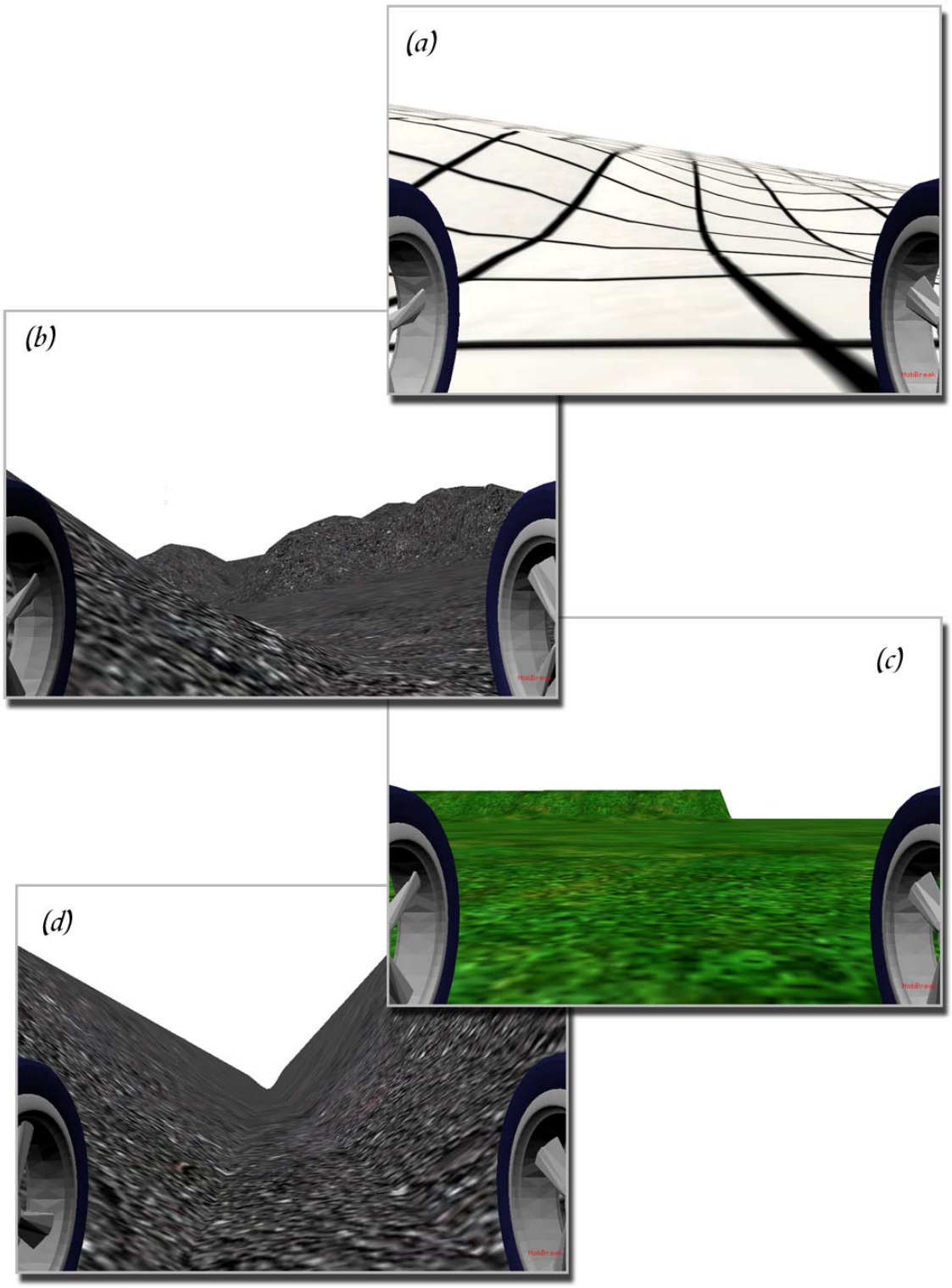


Figura 68 – Visão pela câmera virtual interna do veículo robótico durante várias simulações com diferentes tipos de terrenos

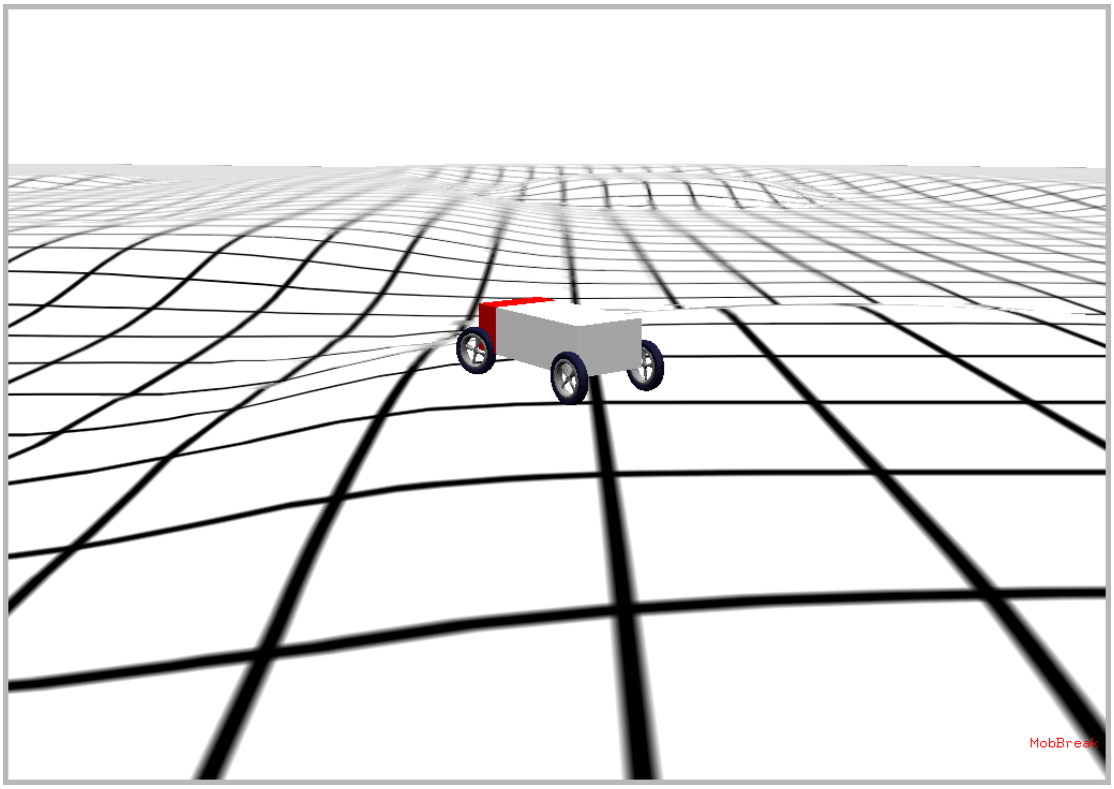


Figura 69 – Imagem obtida pela câmera virtual externa que acompanha o veículo robótico

D.3. Visualizar legendas

A tecla "**L**" (ou "**l**") torna as legendas visíveis ou invisíveis.

D.4. Encerrar (parar) simulação corrente

A tecla "**P**" (ou "**p**") interrompe a execução da simulação corrente e cessa toda a dinâmica e controle do veículo. Para iniciar uma nova simulação, pressione a tecla "**R**" (ou "**r**").

D.5. Iniciar nova simulação

A tecla "**R**" (ou "**r**") inicia uma nova simulação levando em conta os dados iniciais definidos no script de simulação. Essa tecla pode ser pressionada a qualquer momento durante a execução do aplicativo.

D.6. Salvar simulação corrente em vídeo

A tecla "**S**" (ou "**s**") salva o histórico de frames da simulação em um arquivo no formato AVI. Há a opção de escolher entre vários tipos de formatos de acordo com os codificadores e decodificadores instalados no computador (Figura 70). O simulador pergunta qual é o CoDec do vídeo a ser utilizado antes de gerar o arquivo AVI (Figuras 71 a 73).

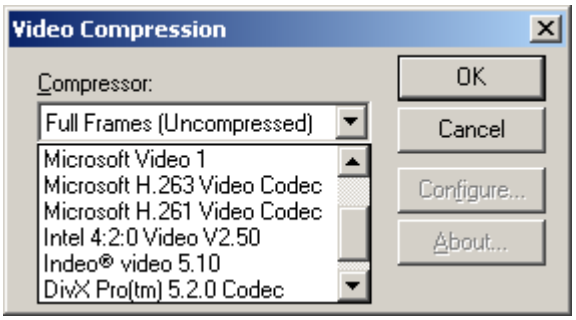


Figura 70 – Janela de seleção do CoDec para geração do vídeo AVI da simulação

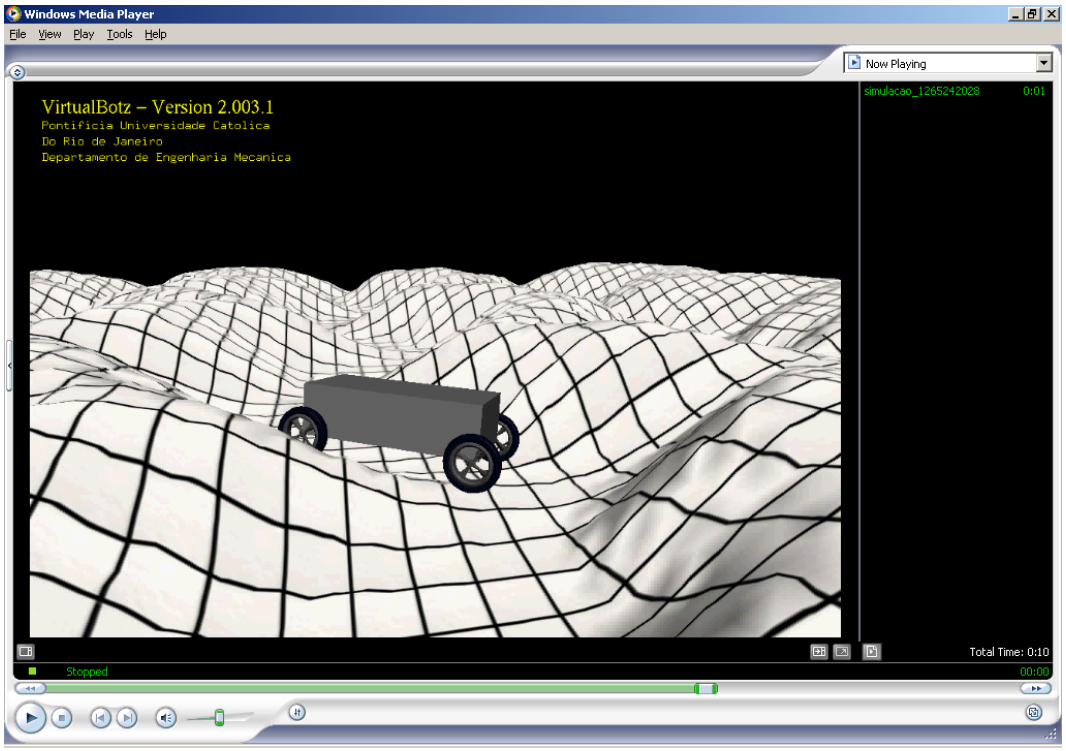


Figura 71 – Imagem de um vídeo AVI gerado com a visão do observador, externa ao veículo

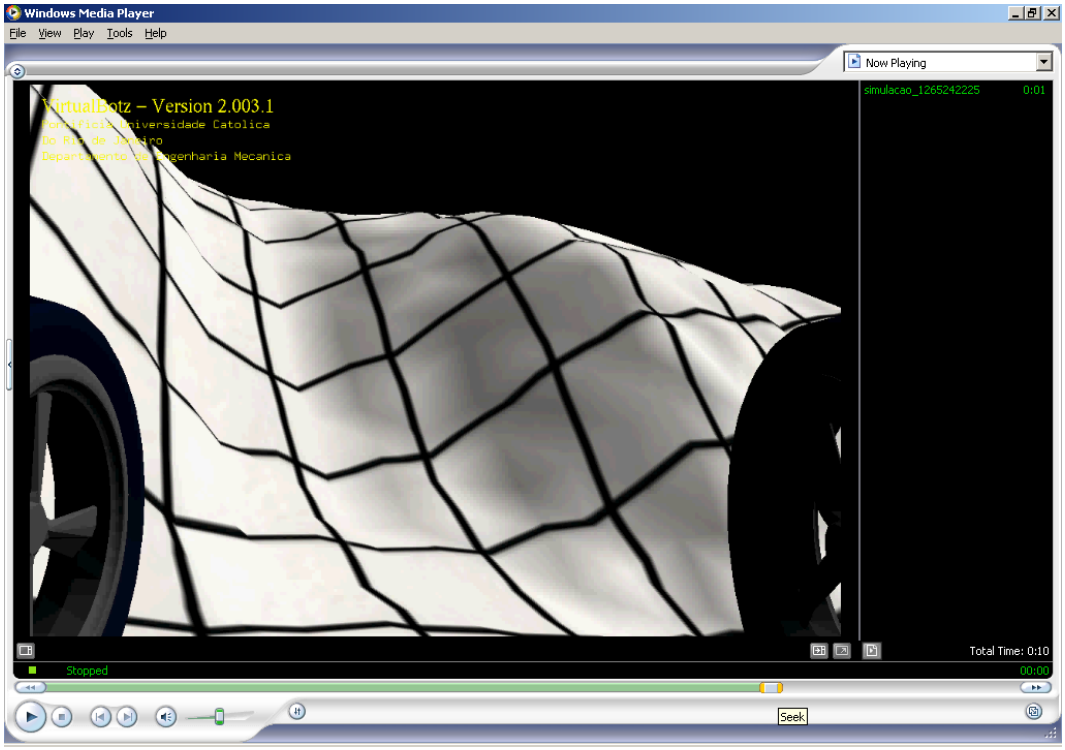


Figura 72 – Imagem de um vídeo AVI gerado com a câmera virtual interna do veículo robótico

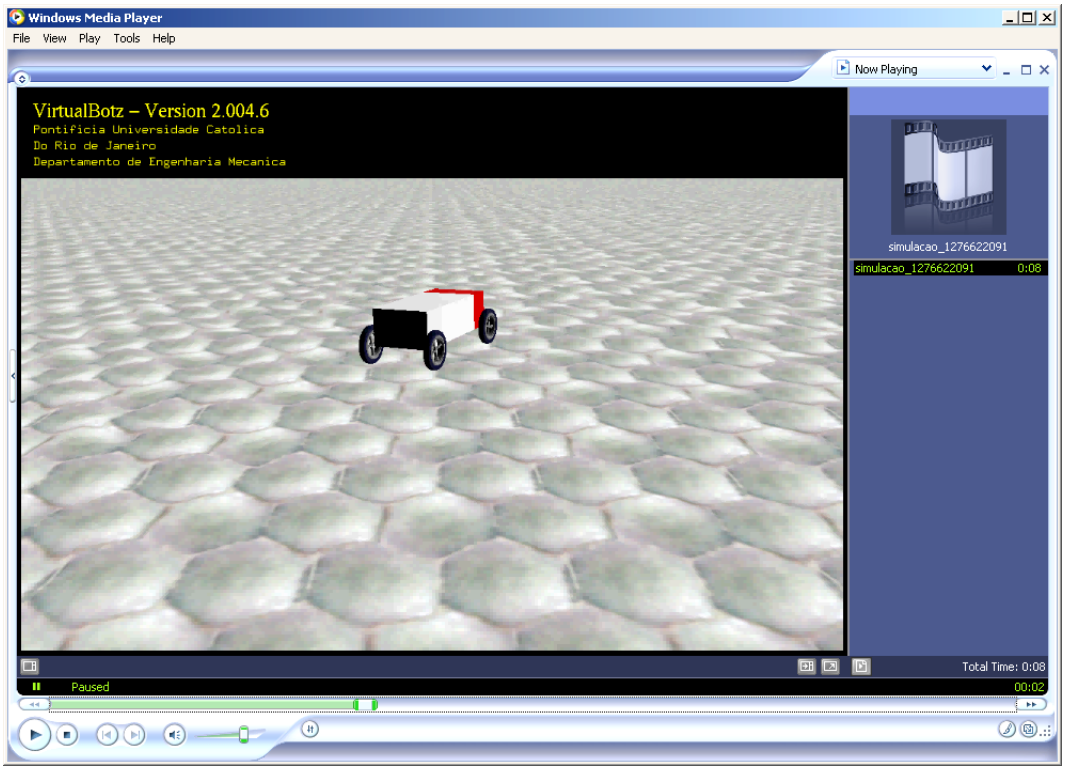


Figura 73 – Imagem de um vídeo AVI gerado com a visão da terceira câmera virtual do aplicativo VirtualBotz 3D, em uma simulação com o tipo de piso similar ao dos pilotis da PUC-Rio

D.7. Visualizar trajetória do centro de massa do veículo robótico após a simulação

A tecla "**T**" (ou "**t**") visualiza as últimas posições do centro de massa do veículo robótico após a interrupção da simulação. Pressionar sucessivamente essa tecla alterna o estilo de apresentação para uma das três opções: vetor ***Z***, vetores ***X***, ***Y*** e ***Z*** ou linha. A quantidade de posições é decorrente da definição da quantidade máxima de *frames* no histórico, predefinido com 2.000 *frames*. Para guardar mais *frames*, basta alterar o atributo "histórico máximo de frames" do script de simulação (**Anexo E**). A seguir estão alguns exemplos dos estilos de apresentação obtidos ao pressionar a tecla "**T**" (Figuras 74 a 76):

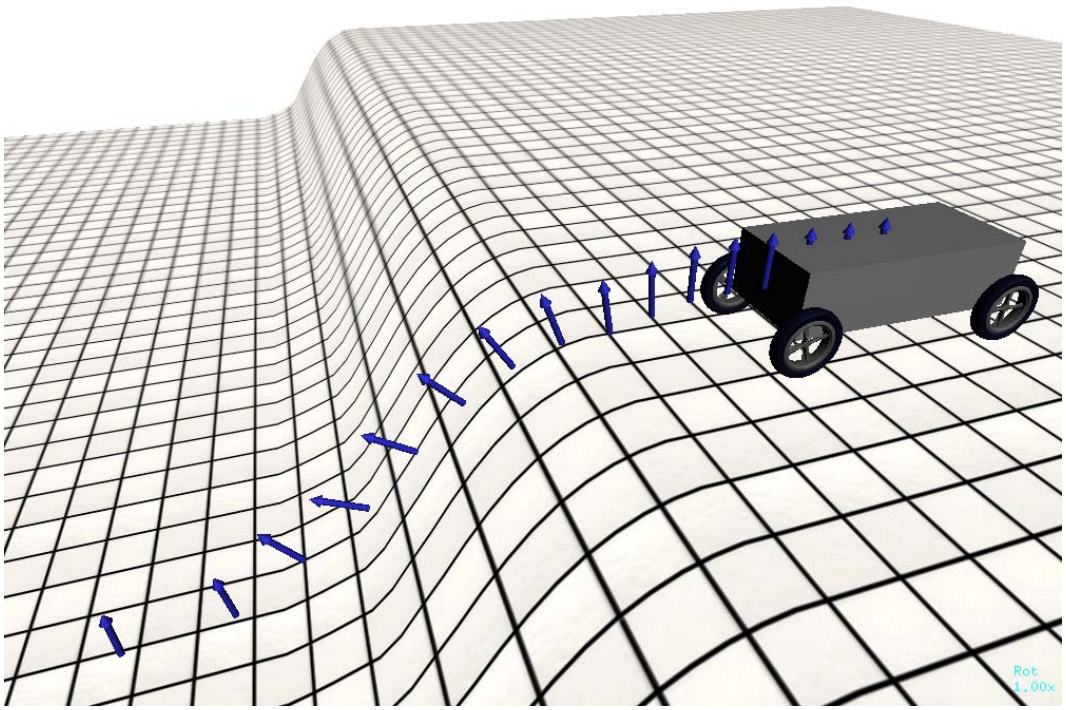


Figura 74 – Veículo robótico com os vetores ***Z*** do centro de massa após o término da simulação

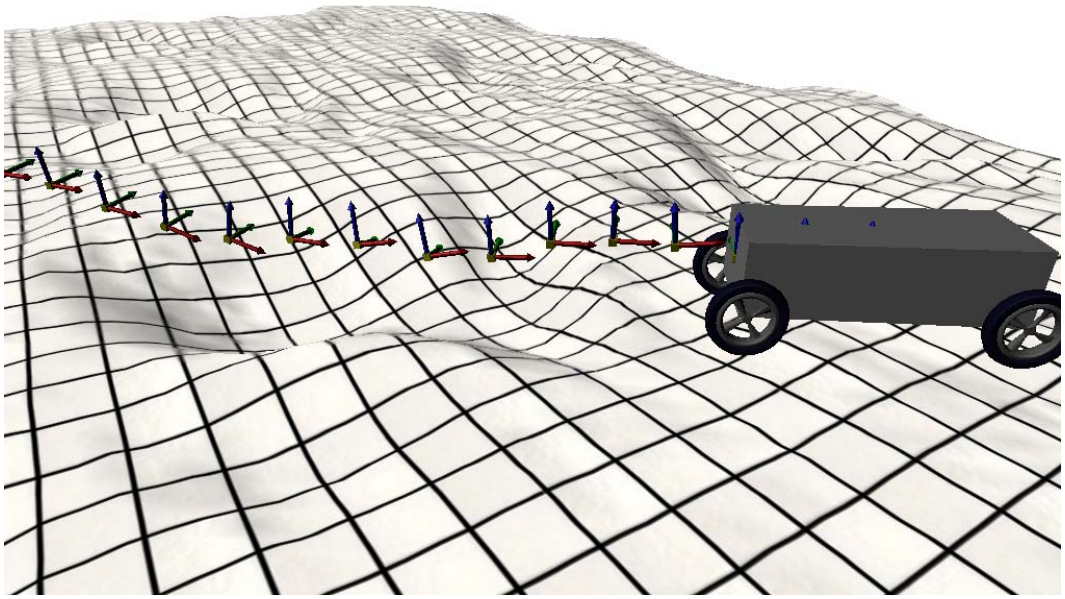


Figura 75 – Veículo robótico com os vetores ***X***, ***Y*** e ***Z*** do centro de massa após o término da simulação

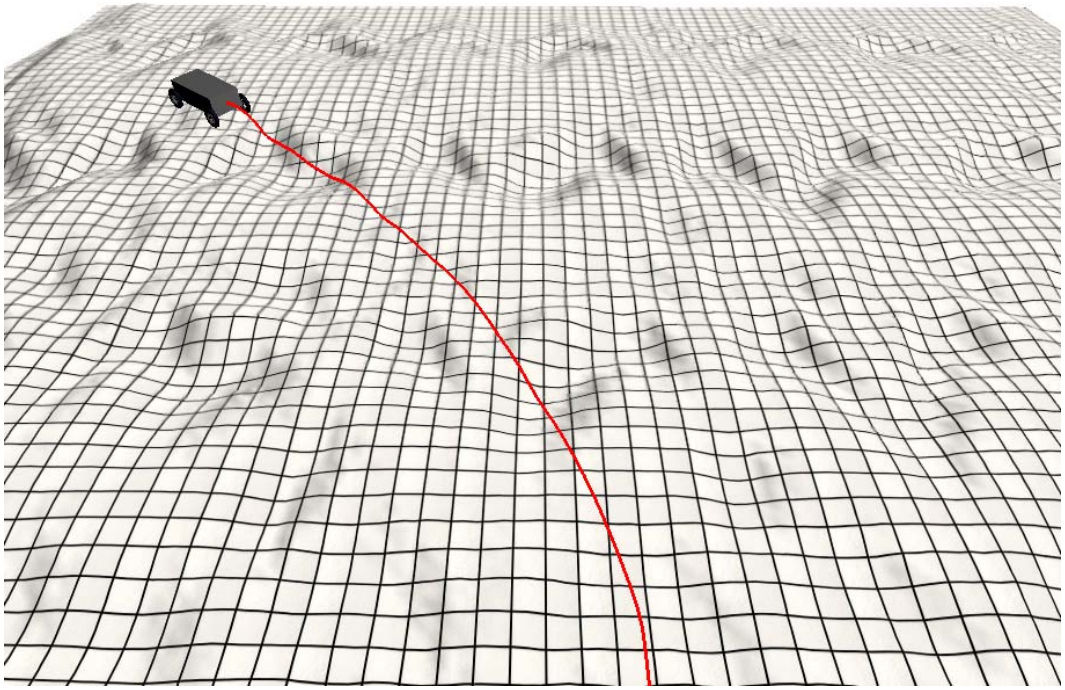
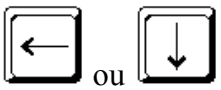


Figura 76 – Veículo robótico com o centro de massa representado por uma linha contínua, após o término da simulação

D.8. Posicionar o veículo robótico pelo centro de massa ao longo da trajetória após a simulação

 ou

A tecla seta "**para esquerda**" ou "**para baixo**" permite o retorno do veículo robótico às posições do centro de massa guardado no histórico ao término da simulação (Figura 77).

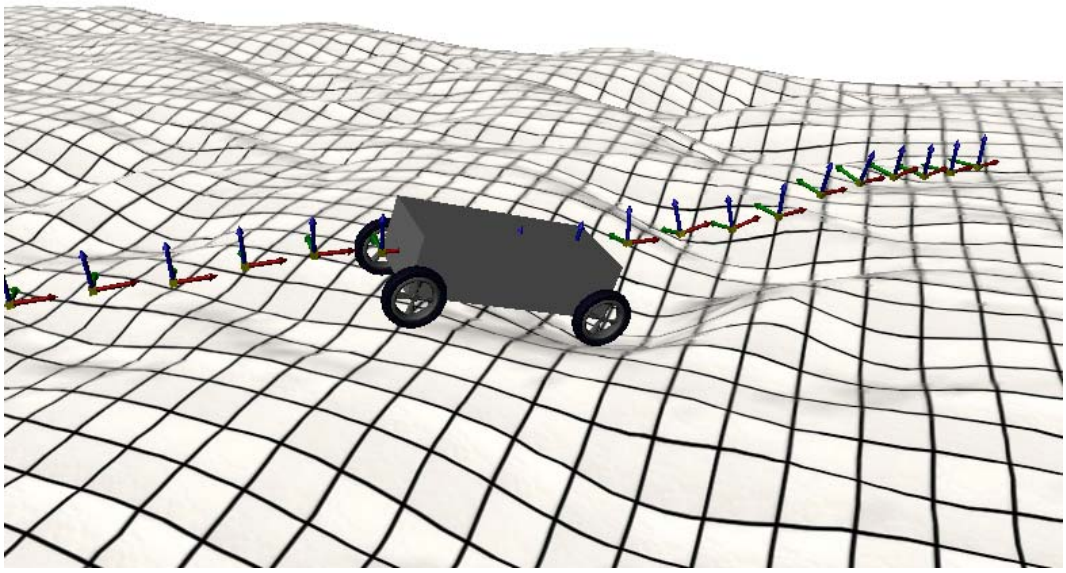
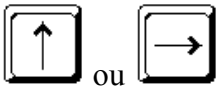


Figura 77 – Veículo robótico retornando aos centros de massa anteriores ao do término da simulação

 ou

A tecla seta "**para cima**" ou "**para a direita**" permite avançar o veículo robótico às posições do centro de massa guardadas no histórico ao término da simulação (Figura 78).

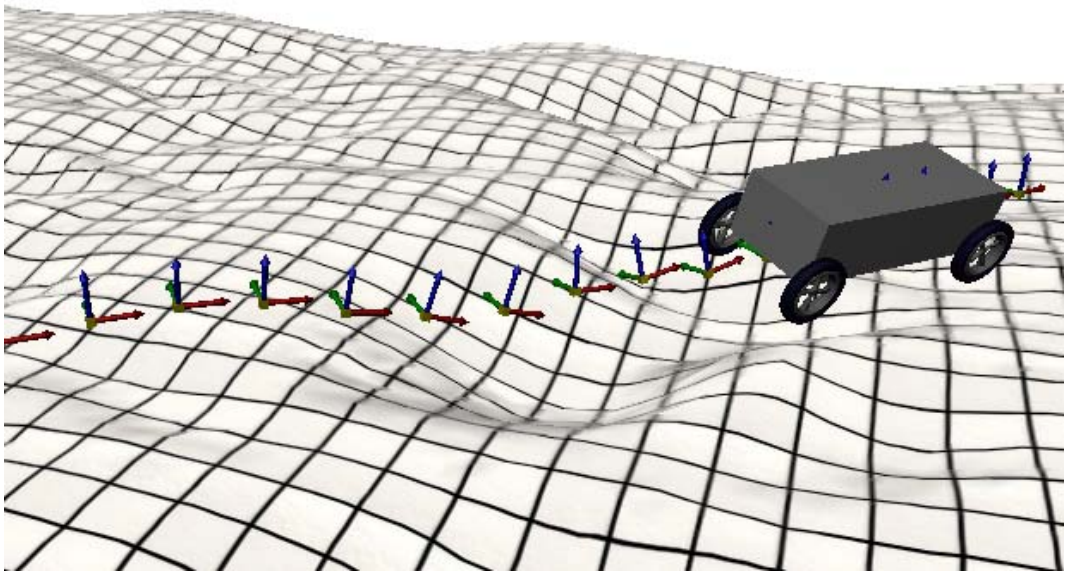


Figura 78 – Veículo robótico avançando às posições dos centros de massa até o término da simulação

D.9. Alterar escala de rotação e translação do mundo virtual

A tecla "**sinal de menor**" permite diminuir a escala de rotação e translação do observador, de 0,05 a 4,00.

A tecla "**sinal de maior**" permite aumentar a escala de rotação e translação do observador, de 0,05 a 4,00.

Anexo E:
Scripts de Configuração do Simulador

Como mencionado na parte “Descrição Geral” do simulador, o software é composto de dois arquivos scripts em formato textual simples, localizados na pasta “Scripts”. É importante salientar que os nomes de atributos não usam nenhum caractere especial, tampouco recursos gráficos linguísticos, como acentuação gráfica ou outros quaisquer. Portanto, o nome de atributo deve conter apenas caracteres textuais simples.

No primeiro arquivo, o principal, de nome “main_script.txt”, indica-se qual arquivo de configuração será utilizado para a simulação. A seguir são mostrados o arquivo principal e a chamada do arquivo de simulação (ou configuração) na linha em negrito. Observe que é possível conservar várias linhas de arquivos de simulação, desde que estejam comentadas com o símbolo “#” (denominado “tralha” ou “jogo da velha”) na primeira coluna de cada linha.

O simulador permite a alteração de qualquer dado dos arquivos scripts e posterior execução sem que seja necessário terminar a execução do programa. Para isso basta fazer a alteração dos dados, salvar o arquivo e teclar “**R**” (ou “**r**”) no simulador.


```
#
## main_script.txt
##
##   DESENVOLVIDO POR : Ricardo Morrot Lima
##              e-MAIL : morrot@gmail.com
##
##               CURSO : Mestrado em Engenharia Mecânica
##             MATÉRIA : Dissertação de Mestrado
##          ORIENTADOR : Prof. Marco Antonio Meggiolaro
##
##                 ANO : 2008-2009
##
## Este arquivo de script é o primeiro a ser lido a cada execução
```

```
## do aplicativo ou a cada restart (tecla "R") durante a
## execução do aplicativo.
##
## Aqui dentro apenas está definido o nome do próximo arquivo
## script que contém todas as definições do terreno, veículo
## robótico, controle e método numérico. Dessa forma, poderão ser
## executadas ao mesmo tempo várias instâncias do aplicativo com
## definições diferentes de terrenos, etc.
##
## Evite deixar qualquer tipo de caractere, inclusive o espaço em
## branco, no final das linhas.
##
## O nome do arquivo script de simulação não deve conter
## caracteres em branco. Devem-se evitar também caracteres ditos
## especiais como cedilhas, acentos, etc.
##
## A última linha deste arquivo não deve conter nada,
## obrigatoriamente.
#
########################################################################
## Configuração da Placa de vídeo ##############################
# Sincronismo Vertical
#     desligado - quando a desempenho é mais importante do que a
#                 qualidade da imagem
#     ligado    - prioridade na qualidade da imagem
sincronismo vertical = desligado
#
########################################################################
#########################
script de simulacao = mobilerobot_test.txt
#script de simulacao = mobilerobot_test_-_rough_terrain.txt
#
## Scripts de comparação do MATLAB com o C++
#
#script de simulacao = mr_test_-_flat_terrain_MATLABVersusC++.txt
#script de simulacao = mr_test_-_rough_terrain_MATLABVersusC++.txt
#script de simulacao = mr_test_-_flat_terrain.txt
#script de simulacao = mr_test_-_heightmap_rough_terrain.txt
#script de simulacao = mr_test_-_heightmap_1_rough_terrain.txt
```

No arquivo de simulação/configuração, não se deve alterar nada além do necessário. É preferível salvar o arquivo com um novo nome e indicá-lo no arquivo principal a alterar e salvar e/ou inserir vários comentários, pois polui o arquivo.

Em nenhum momento é permitida a troca na ordem dos atributos do arquivo de simulação. Isso pode gerar inconsistência de dados e consequente erro de leitura de dados ou erro na execução da simulação.

A seguir é apresentado um exemplo de arquivo (script) de simulação.

```
## mobile_robot_test.txt : Arquivo de configuração para simulação
##
##   DESENVOLVIDO POR : Ricardo Morrot Lima
##              e-MAIL : morrot@gmail.com
##
##               CURSO : Mestrado em Engenharia Mecânica
##             MATÉRIA : Dissertação de Mestrado
##          ORIENTADOR : Prof. Marco Antonio Meggiolaro
##
##                 ANO : 2008-2009
##
## Este arquivo de script deve permanecer na ordem em que se
## encontra.
## Ao definir novos valores para os atributos, preste atenção
## para não gerar inconsistência de dados com esses valores.
##
## Evite deixar qualquer tipo de caractere, inclusive espaço em
## branco, no final das linhas.
##
## Importante!
##
##   - Os nomes de quaisquer arquivos, como por exemplo os
##     de malha, modelos OBJ, texturas, não devem conter espaço em
##     branco ou qualquer outro caractere gráfico linguístico,
##     como acentuação, entre outros.
```

```
#
#####################################################################
## Título do script (entre aspas)
titulo = "Teste de subida na rampa terrain_mesh2.msh"
#
## Dados do Terreno ################################################
#
# Tipos possíveis para o terreno
# 1 - Rubber Asphalt Dry ( borracha em asfalto seco )
# 2 - Rubber Asphalt Wet ( borracha em asfalto molhado )
# 3 - Rubber Concrete Dry ( borracha em concreto seco )
# 4 - Rubber Concrete Wet ( borracha em concreto molhado )
# 5 - Dry Sand ( areia seca )
# 6 - Sandy Loam ( terreno arenoso, mais pra areia)
# 7 - Clayey Soil ( terreno argiloso, de barro)
# 8 - Snow ( neve )
# 9 - Washed Sand ( "areia lavada", sem materiais orgânicos e sal )
# 10 - Dried Bentonite Clay ( um determinado terreno argiloso seco )
# 11 - Compacted Topsoil ( terra batida, compactada )
#
tipo do terreno = 3
#
# Coordenadas extremas do terreno
vmin = { -40.0, -40.0, 0.0 }
vmax = { 40.0, 40.0, 0.0 }
#
# Discretização do terreno
pontos = 200.0
#
# Textura do terreno
# Importante!
# - O arquivo de textura do terreno deve estar na pasta
#   (diretório) "Textures\".
# - Caso não utilize nenhuma textura digite "none", sem aspas,
#   para o atributo "texture file".
# - Anti-aliasing do terreno deve ser "yes" ou "no", sem aspas.
anti-aliasing = yes
#texture file = none
texture file = solo2.bmp
#
# Tipos possíveis de terreno: senos, flat, mesh, image
```

```
tipo do terreno = image
#
# Apenas para mesh ----------------------------------
# Se o terreno não for lido de um arquivo de malha
# comente as linhas a seguir.
# Importante!
# - O arquivo do mapa de altura deve estar na pasta
#   (diretório) "Obj_files\".
#mesh file = terrain1.msh
#
# Apenas para image (mapa de altura) -----------------
# Se o terreno não for lido de um arquivo de imagem
# comente as linhas a seguir.
# Importante!
# - O arquivo do mapa de altura deve estar na pasta
#   (diretório) "Heightmaps\".
image file = heightmap52.bmp
altura da escala de cinza = 1.0
##
######################################################################
## Dados dos Motores #################################################
##
## Cada roda pode ter um atuador motor com características
## específicas ou todas podem ter o mesmo tipo de atuador motor
## com as mesmas características. Entretanto, o número de
## atuadores motor não pode ultrapassar a quantidade de rodas
## estipuladas logo acima.
##
## - Para definir atuadores motor de mesma marca com diferentes
##   especificações, basta colocar nomes diferentes, por exemplo:
##       motor = 0
##       nome, modelo e versao = Magmotor S28-150 ver1.0
##       ...
##       motor = n
##       nome, modelo e versao = Magmotor S28-150 ver1.2
##
## - Os atributos abaixo serão os mesmos para todos os atuadores
##   motor definidos nesta seção.
##
## Regras:
##
```

```
##   1) O primeiro atuador motor deve começar em zero.
##   2) Não pode ser utilizado o caractere espaço em branco
##      no nome do modelo.
#
# ------------------------------------------------
motor = 0
nome do modelo e versao = Magmotor_S28-150
torque constante Kt[Nm/A] = 0.03757
velocidade constante Kv[(rad/s)/V] = 26,61698
resistencia total do motor Rmotor[Ohm] = 0.148
corrente Inoload[A] = 3.4
# Para limitar a potencia da bateria PN3600,
# fazer Imax = 80 e Vmax = 24
corrente maxima[A] = 80
tensao maxima[V] = 36
caixa de reducao (adimensional) = 7.14
#
#####################################################################
## Dados do Veículo Robótico ########################################
robot opengl file name = chassi_2.obj
## Orientação inicial do veículo robótico
vetor n = { 1.0, 0.0, 0.0 }
vetor t = { 0.0, 1.0, 0.0 }
vetor b = { 0.0, 0.0, 1.0 }
## Posição inicial do veículo robótico
centro de massa = { -46.0, 0.0, 2.3 }
## Velocidade linear inicial
velocidade do centro de massa = { 0.0, 0.0, 0.0 }
## Velocidade angular
vetor w = { 0.0, 0.0, 0.0 }
## Centro de massa do veículo robótico
## Vetor centro de massa = { right, left, front, back, top, bottom }
right = 1.0
left = 1.0
front = 2.0
back = 2.0
top = 0.7
bottom = 0.5;
## Massa em quilos
massa = 120.0
quantidade de rodas = 4
```

```
quantidade de divisoes da roda = 18
rigidez do chassi = 1e4
amortecimento = 5e2
## Definição dos corners (quinas)
quantidade de corners = 8
## Posição dos corners na coordenada n-t-b local ----------------
corner = 0
ntb = { -back, -right, -bottom }
corner = 1
ntb = { front, -right, -bottom }
corner = 2
ntb = { front, left, -bottom }
corner = 3
ntb = { -back, left, -bottom }
corner = 4
ntb = { -back, -right, top }
corner = 5
ntb = { front, -right, top }
corner = 6
ntb = { front, left, top }
corner = 7
ntb = { -back, left, top }
#
## Rodas ---------------------------------------------------------
## Cada roda tem a sua configuração independente. Isso quer dizer
## que se no atributo "quantidade de rodas" em "Dados do Veículo
## Robótico" forem definidas seis rodas, as seis rodas deverão
## ser definidas uma a uma nesta seção.
##
## Regras:
##
##   1) A primeira roda deve começar em zero.
##   2) As coordenadas de cada roda devem seguir uma sequência
##      anti-horária. Essa sequência é fundamental para o controle
##      de estabilidade do veículo.
#
# ------------------------------------------------
roda = 0
nome do motor = Magmotor_S28-150
modelo da roda nome do arquivo = supertire_43.obj
rigidez = 1e4
```

```
amortecimento = 5e2
largura = 0.2
raio = 0.6
deslocamento h = 0.0
saturacao de h = 0.2
velocidade relativa da suspensao = 0.0
torque de saturacao = 30.0
# Torque constante: no ou yes (valendo o que está em torque,
# logo abaixo)
torque constante = no
torque = 0.0
# Deslizamento da roda: no ou yes
desliza = no
# Pneu
rigidez lateral do pneu = 3e4
rigidez longitudinal do pneu = 3e4
coordenadas local do centro de massa (x,y,z) = { -2.0, -1.1, -0.5 }
#
# ------------------------------------------------
roda = 1
nome do motor = Magmotor_S28-150
modelo da roda nome do arquivo = supertire_43.obj
rigidez = 1e4
amortecimento = 5e2
largura = 0.2
raio = 0.6
deslocamento h = 0.0
saturacao de h = 0.2
velocidade relativa da suspensao = 0.0
torque de saturacao = 30.0
# Torque constante: no ou yes (valendo o que está em torque,
# logo abaixo)
torque constante = no
torque = 0.0
# Deslizamento da roda: no ou yes
desliza = no
# Pneu
rigidez lateral do pneu = 3e4
rigidez longitudinal do pneu = 3e4
coordenadas local do centro de massa (x,y,z) = { 2.0, -1.1, -0.5 }
#
```

```
# ------------------------------------------------
roda = 2
nome do motor = Magmotor_S28-150
modelo da roda nome do arquivo = supertire_43.obj
rigidez = 1e4
amortecimento = 5e2
largura = 0.2
raio = 0.6
deslocamento h = 0.0
saturacao de h = 0.2
velocidade relativa da suspensao = 0.0
torque de saturacao = 30.0
# Torque constante: no ou yes (valendo o que está em torque,
# logo abaixo)
torque constante = no
torque = 0.0
# Deslizamento da roda: no ou yes
desliza = no
# Pneu
rigidez lateral do pneu = 3e4
rigidez longitudinal do pneu = 3e4
coordenadas local do centro de massa (x,y,z) = { 2.0, 1.1, -0.5 }
#
# ------------------------------------------------
roda = 3
nome do motor = Magmotor_S28-150
modelo da roda nome do arquivo = supertire_43.obj
rigidez = 1e4
amortecimento = 5e2
largura = 0.2
raio = 0.6
deslocamento h = 0.0
saturacao de h = 0.2
velocidade relativa da suspensao = 0.0
torque de saturacao = 30.0
# Torque constante: no ou yes (valendo o que está em torque,
# logo abaixo)
torque constante = no
torque = 0.0
# Deslizamento da roda: no ou yes
desliza = no
```

```
# Pneu
rigidez lateral do pneu = 3e4
rigidez longitudinal do pneu = 3e4
coordenadas local do centro de massa (x,y,z) = { -2.0, 1.1, -0.5 }
#
#####################################################################
## Controle ########################################################
# Timer sempre em milissegundos
timer = 50
torque Tal = 30.0      # torque Tal constante de torque das rodas
#
#####################################################################
# Métodos numéricos (Euler) ########################################
delta T = 0.001
#
# A próxima opção de configuração guarda uma grande quantidade
# de dados durante a simulação. Não utilize valores muito altos
# para o máximo de frames. Isso pode gerar trocas (swap) de
# memória com o disco rígido, tornando a simulação lenta e com
# efeito quebradiço, ou seja, com pausas.
# Quando o máximo de frames é igual a zero, o aplicativo é
# informado de que a simulação otimiza a memória e não guarda
# um histórico com muitos registros de dados, apenas o
# necessário para a simulação em tempo real.
#
#historico maximo de frames = 0
#historico maximo de frames = 630
#historico maximo de frames = 830
#historico maximo de frames = 1000
historico maximo de frames = 2000
#historico maximo de frames = 4000
#historico maximo de frames = 6150
#historico maximo de frames = 10000
## o limite abaixo é o máximo permitido(evite!)
#historico maximo de frames = 2147483647
#
#####################################################################
# Geração de vídeo, do tempo no histórico da simulação, em
# formato AVI ###################
#
# - O nome do arquivo NÃO DEVE conter ponto, extensão, espaço em
```

```
#   branco ou qualquer outro caractere gráfico linguístico,
#   como acentuação, entre outros.
# - O arquivo gerado será gravado na pasta "Videos".
#
nome do arquivo AVI = simulacao
#
# - Caso o atributo "timestamp", logo abaixo, esteja "yes", todos
#   os arquivos gerados conterão em seu nome a data e a hora
#   (timestamp), facilitando o armazenamento de várias visões para
#   a mesma simulação. Se o atributo estiver "no", então o arquivo
#   será reescrito.
timestamp = yes
#
# - O frame rate ideal para arquivos AVI é na faixa de 15 a 25
#   frames por segundo, inclusive.
frame rate = 25
#
######################################################################
# Graph ##############################################################
# Você pode especificar um arquivo com coordenadas XYZ em
# sequência. Esse arquivo será mostrado como uma linha gráfica.
# Do contrário, basta colocar "none" para que nenhum gráfico seja
# mostrado.
# É imprescindível que esse arquivo esteja na pasta (diretório)
# "testes\".
XYZ file stream = "none"      ## nenhum arquivo de gráfico será mostrado
#XYZ file stream = "MATLAB_xCM_mobile_robot.txt"
#XYZ file stream = "MATLAB_xCM_mobile_robot_flat.txt"
cor da linha = { 1.0, 0.0, 0.0 }          ## cor da linha em RGB
##
##
######################################################################
# Geração de arquivos de dados #######################################
#
# - O nome do arquivo NÃO DEVE conter ponto, extensão, espaço em
#   branco ou qualquer outro caractere gráfico linguístico, como
#   acentuação, entre outros.
# - Os arquivos gerados serão gravados na pasta "Report".
# - Para desligar a saída dos dados, basta digitar a palavra "off"
#   no início da linha referente ao dados a ser desligado. Para
#   ligar, basta digitar "on".
```

```
#
on,centro de massa do veiculo = mobile_xCM.txt
on,centro de massa das rodas do veiculo = wheels_xCM.txt
off,torque das rodas = wheels_torques.txt
off,sinal de controle das rodas = wheels_ControlSign.txt
##
## FIM
```

Anexo F:
Opções de Terrenos

Há quatro formas de configurar um terreno para simulação:

- Senos

Senos (Figura 79) é um formato predefinido a partir de equações senoidais para gerar um terreno ondulado. Para mudar a característica desse terreno, deve-se acessar o arquivo “CVBTerrain_.cpp” e mudar a equação, como visto a seguir:

```
void CVRTerrainSines::createMesh(void)
{
   unsigned int ix, jy;

   for (jy = 0; jy < getDimY(); jy++)
      for (ix = 0; ix < getDimX(); ix++)
      {
         setHeight(jy, ix, (0.3 * (1.0+sin(jy*PI/10.0)*1.0)*(1.0+sin(ix*PI/10.0)*1.0)) );
//       setHeight(jy, ix, (0.3 * (1.0+sin(jy*PI/20.0)*1.0)) );
//       setHeight(jy, ix, (0.3 * (1.0+sin(jy*PI/20.0)*1.0)*(1.0+sin(ix*PI/20.0)*1.0)) );
      }
}
```

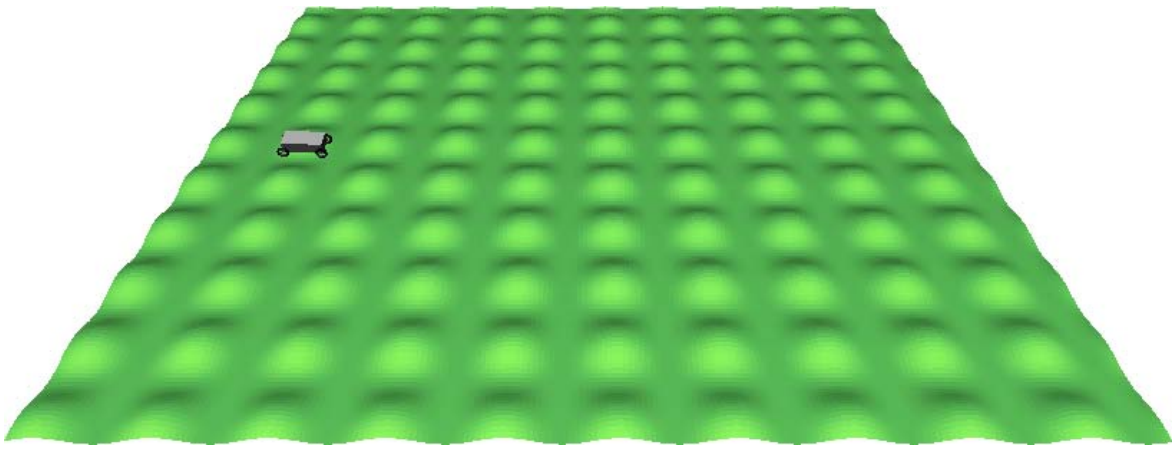


Figura 79 – Exemplo de opção de terreno senos

- *Flat*

Flat é um formato predefinido que permite a seleção de um terreno plano horizontal, sem nenhuma ondulação (Figura 80).

Figura 80 – Exemplo de opção de terreno *flat*

- *Mesh*

Mesh é um formato de malha que, especificamente para o terreno, deve ser aberta. O importante dessa malha para o sistema é que as coordenadas de cada vértice estejam numa grade regular retangular, ou seja, formem polígonos regulares retangulares (Figura 81). Dessa forma, o sistema recupera o arquivo de malhas com todos os vértices. As dimensões *dX* e *dY* entre os vértices devem ser iguais para esse tipo de malha de terreno (Figura 81).

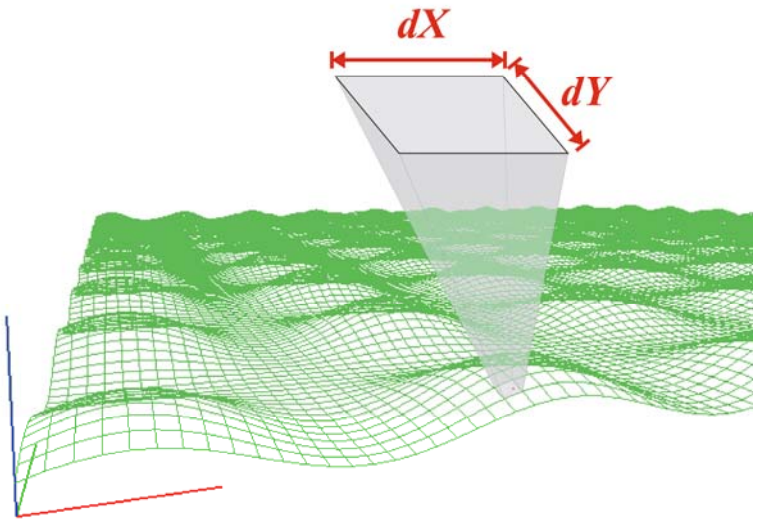


Figura 81 – Ilustração do terreno de grade regular retangular

O arquivo contendo a malha deve ser do tipo texto, contendo na primeira linha o número total de vértices e, a partir da segunda linha, começando em zero, o número do vértice e as coordenadas *x*, *y* e *z*, todos separados por espaço em branco. Veja o exemplo abaixo:

Arquivo "terrain_mesh.msh".

```
40401
0 -40.000000 -40.000000 0.000000
1 -40.000000 -39.600000 0.000000
2 -40.000000 -39.200000 0.000000
3 -40.000000 -38.800000 0.000000
4 -40.000000 -38.400000 0.000000
5 -40.000000 -38.000000 0.000000
6 -40.000000 -37.600000 0.000000
7 -40.000000 -37.200000 0.000000
8 -40.000000 -36.800000 0.000000
9 -40.000000 -36.400000 0.000000
10 -40.000000 -36.000000 0.000000
11 -40.000000 -35.600000 0.000000
12 -40.000000 -35.200000 0.000000
13 -40.000000 -34.800000 0.000000
14 -40.000000 -34.400000 0.000000
15 -40.000000 -34.000000 0.000000
16 -40.000000 -33.600000 0.000000
17 -40.000000 -33.200000 0.000000
18 -40.000000 -32.800000 0.000000
19 -40.000000 -32.400000 0.000000
...
40390 40.000000 36.000000 0.000000
40391 40.000000 36.400000 0.000000
40392 40.000000 36.800000 0.000000
40393 40.000000 37.200000 0.000000
40394 40.000000 37.600000 0.000000
40395 40.000000 38.000000 0.000000
40396 40.000000 38.400000 0.000000
40397 40.000000 38.800000 0.000000
40398 40.000000 39.200000 0.000000
40399 40.000000 39.600000 0.000000
40400 40.000000 40.000000 0.000000
```

A malha dentro do arquivo deve seguir uma ordenação de linha por colunas (Figuras 82 e 83) de modo a evitar uma sobrecarga de processamento desnecessário para reordenamento toda vez que for aberto um arquivo para leitura e renderização. O reordenamento do arquivo, caso ocorra, será sempre pelas coordenadas X_{min}/Y_{min} e X_{max}/Y_{max}.

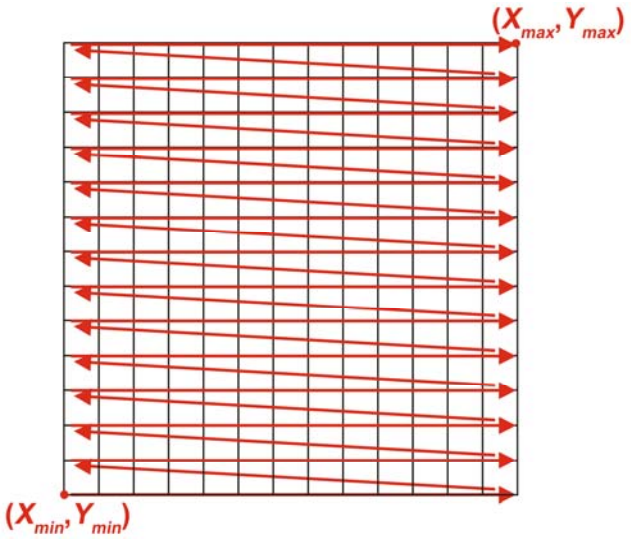


Figura 82 – Exemplo da ordenação da malha do terreno

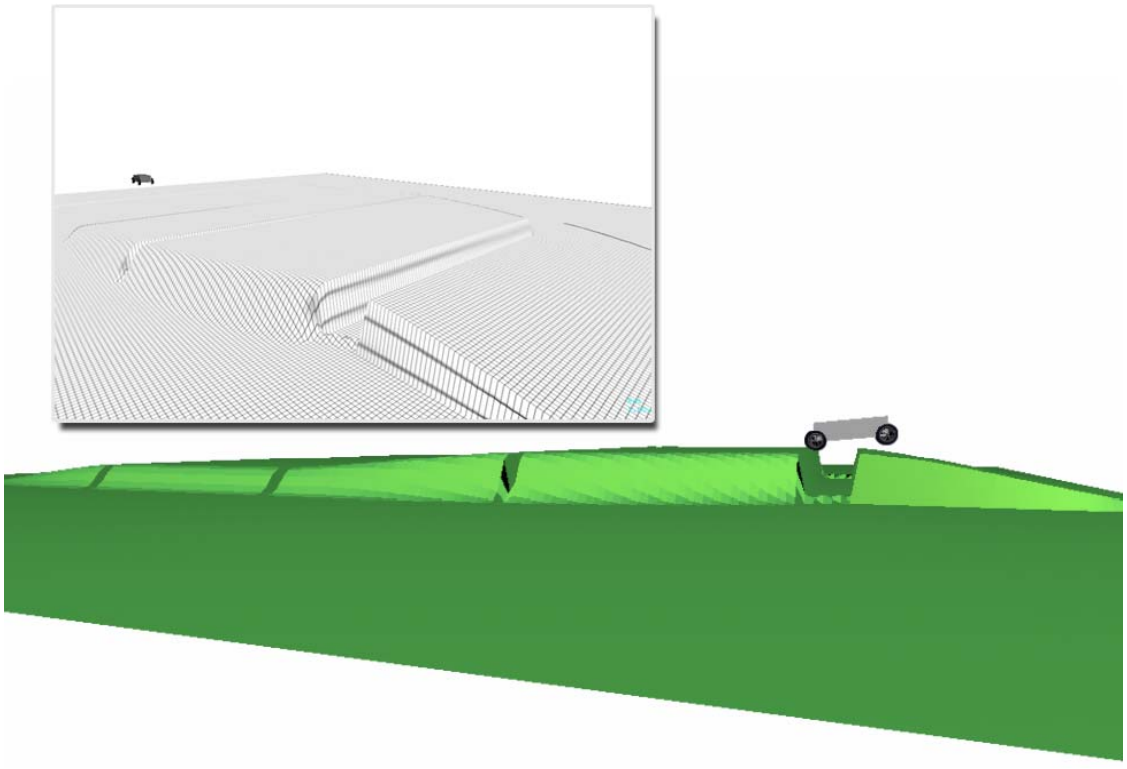


Figura 83 – Simulação utilizando a leitura de terreno do tipo *mesh* (arquivo de malha "terrain_mesh2.msh")

- *Image*

Image é uma opção de leitura de terreno a partir de uma imagem gráfica em escala de cinza, ou seja, um mapa de altura. Nessa opção há a possibilidade de definir a altura máxima da escala. Exemplos são mostrados nas Figuras 84, 85 e 86.

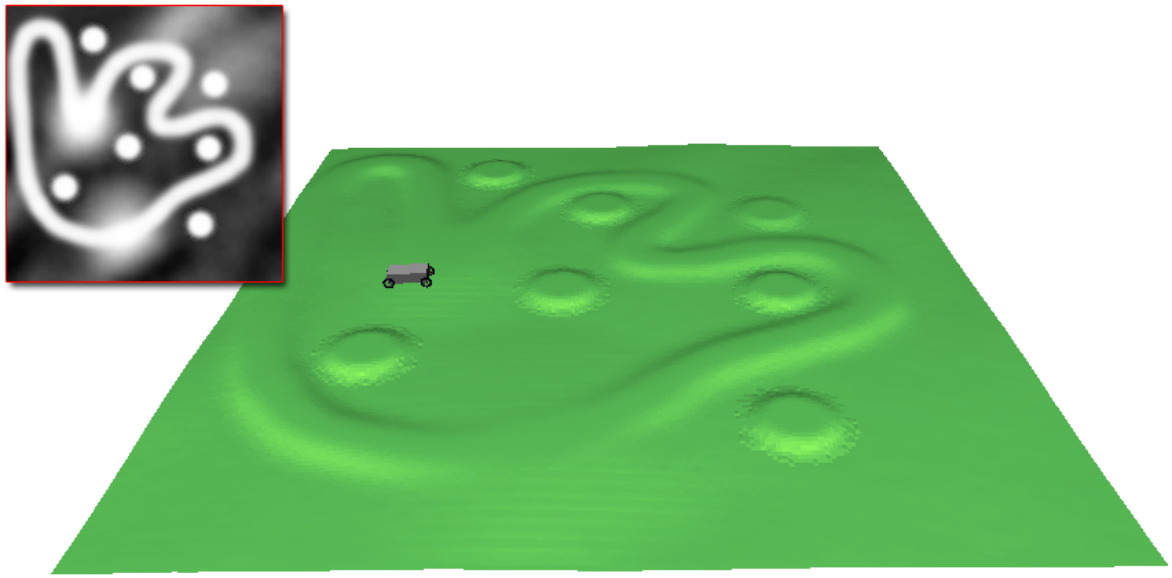


Figura 84 – Exemplo de opção de terreno *image* (mapa de altura) com o arquivo "heightmap3.bmp" definido para altura máxima "1.0"

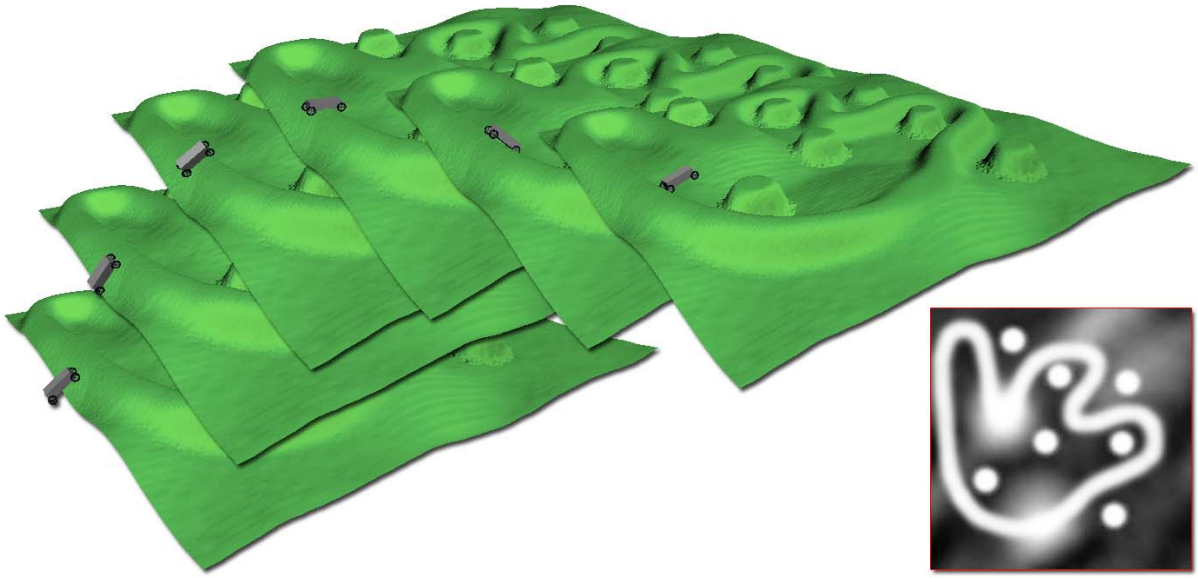


Figura 85 – Exemplo de opção de terreno *image* (mapa de altura) com o arquivo "heightmap3.bmp" definido para altura máxima "5.0", com o veículo robótico superando o aclive do terreno em diferentes instantes

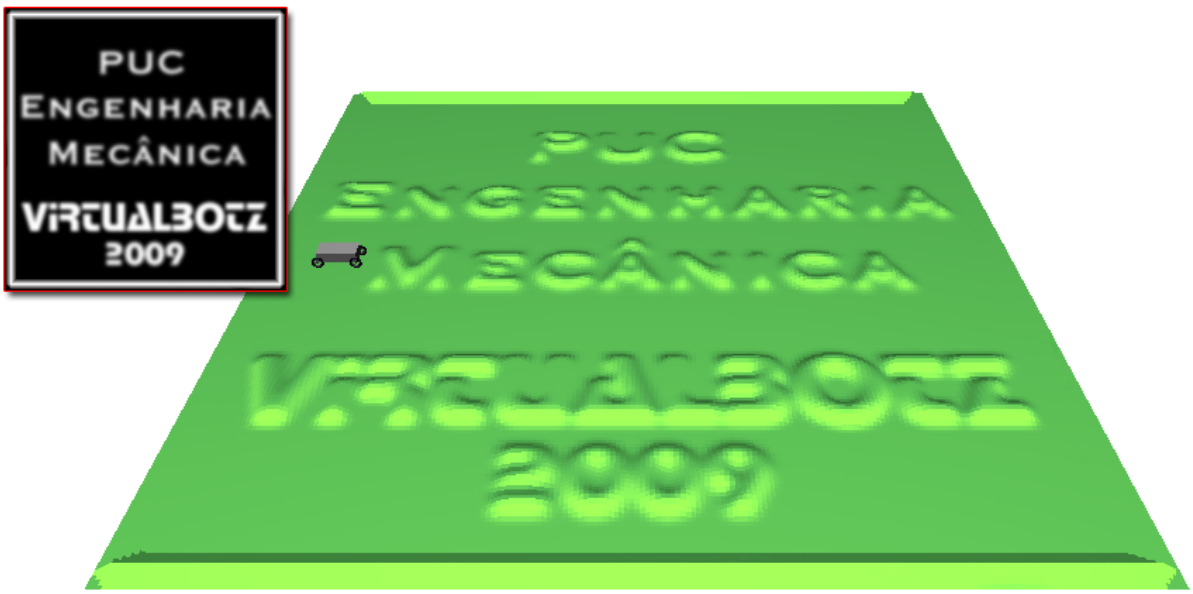


Figura 86 – Exemplo de opção de terreno *image* (mapa de altura) com o arquivo “heightmap52.bmp” definido para altura máxima “1.0”

Anexo G:
Configurando Opção de Terreno no Script da Simulação

No script corrente, logo a seguir, o terreno está selecionado como *image* (mapa de altura) e, na área de opções do mapa de altura, as linhas com o nome do arquivo e a escala para altura máxima não estão comentadas (sem o símbolo "#"). O mesmo procedimento serve para a opção *mesh*. Já no caso de uma das opções de terreno senos ou *flat*, devem-se comentar as linhas correspondentes às opções de *mesh* e *image*, com o símbolo "#" posicionado na primeira coluna de cada linha.

```
...
#
######################################################################
## Dados do Terreno ##################################################
#
# Coordenadas extremas do terreno
vmin = { -40.0, -40.0, 0.0 }
vmax = { 40.0, 40.0, 0.0 }
#
# Discretização do terreno
pontos = 200.0
#
# Textura do terreno
# Importante!
# - O arquivo de textura do terreno deve estar na pasta
#   (diretório) "Textures\".
# - Caso não utilize textura alguma, digite "none", sem aspas,
#   para o atributo "texture file".
# - Anti-aliasing do terreno deve ser "yes" ou "no", sem aspas.
anti-aliasing = yes
#texture file = none
texture file = solo2.bmp
#
# Tipos possíveis de terreno : senos, flat, mesh, image
```

```
tipo do terreno = image
#
# Apenas para mesh ----------------------------------
# Se o terreno não for lido de um arquivo de malha,
# comente as linhas a seguir.
# Importante!
# - O arquivo do mapa de altura deve estar na pasta
#   (diretório) "Obj_files\".
#mesh file = terrain_mesh.msh
#
# Apenas para image (mapa de altura) -----------------
# Se o terreno não for lido de um arquivo de imagem,
# comente as linhas a seguir.
# Importante!
# - O arquivo do mapa de altura deve estar na pasta
#   (diretório) "Heightmaps\".
image file = heightmap52.bmp
altura da escala de cinza = 1.0
#
#########################################################################
...
```

Anexo H:
Renderização do Terreno

A renderização de terrenos é um assunto muito discutido na comunidade acadêmica. O objetivo desta dissertação não é definir um algoritmo novo de renderização de terrenos, muito menos entrar em detalhes sobre as diferentes técnicas de implementação existentes, e sim, em vista da quantidade de dados a ser administrada durante a execução do simulador, usar a estrutura que melhor se adéque [42].

O simulador leva em conta a discretização do terreno em uma série de pequenos intervalos para *x* e *y* de forma que a parte do algoritmo do simulador responsável pelo cálculo do ponto de contato das rodas do veículo robótico com o terreno calcule com exatidão a posição de contato. Quanto menor o intervalo, maior será a quantidade de pontos, ou vértices, e maior será o custo operacional do cálculo. Os exemplos expostos nos **Anexos F** e **G** apresentam terrenos de dimensão 80x80 com uma discretização de 200 pontos tanto para *x* quanto para *y*.

```
...
# Coordenadas extremas do terreno
vmin = { -40.0, -40.0, 0.0 }
vmax = { 40.0, 40.0, 0.0 }
#
# Discretização do terreno
pontos = 200.0...
...
```

Logo, tem-se uma matriz de 200x200 com 40.000 vértices com intervalos regulares de 0.40m. Para manter a fidelidade do terreno, evitando que haja perda na qualidade visual (Figura 87) e principalmente não venha a prejudicar a velocidade de cálculo do programa, foi utilizada a sequência de QUADS [42].

Se um vetor de índice para os vértices (*vertexes array*) fosse usado para primitivas GL_TRIANGLES, seriam gerados 237.606 elementos do tipo Glfloat.

Com o GL_QUADS, passa-se a ter 158.404 elementos. Esse cálculo começa a tomar proporções bem grandes quando levam-se em conta os vetores das normais, as coordenadas de textura e os vértices, com 40.000 elementos para cada vetor. Isso sem contar os vetores de vértices e normais com (x, y, e z) e o vetor de coordenadas de textura com (s, t e r).

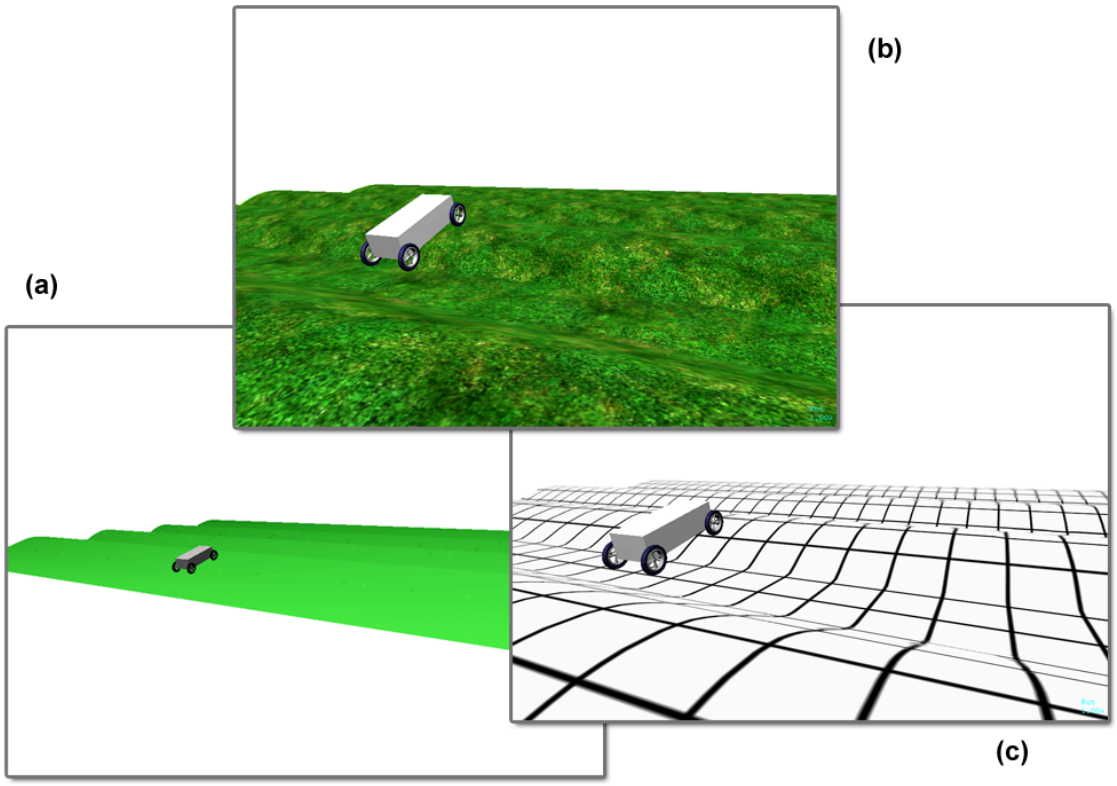


Figura 87 – As três simulações utilizam o mesmo terreno criado a partir de um mapa de altura (arquivo “heightmap63.bmp”), sendo a ilustração (a) sem nenhuma textura, com (b) textura “grass3.bmp” e (c) com “tile31.bmp”

Anexo I:
Configurando o Veículo Robótico no Script da Simulação

A configuração do veículo robótico é feita logo após a configuração do terreno (**Anexos F** e **G**). Devem-se levar em consideração vários fatores antes de começar a definir os valores dos atributos do veículo:

- orientação inicial
- posição inicial
- velocidade linear inicial
- velocidade angular
- centro de massa
- massa (kg)
- número de rodas
- número de divisões de cada roda (parte inferior)
- rigidez do chassi
- amortecimento
- número de quinas do chassi do veículo
- posição das quinas na coordenada ntb local

Imediatamente após as definições acima, é preciso definir as rodas, uma após a outra, iniciando por zero, até totalizar o valor definido para o atributo "quantidade de rodas" menos um. Seguem as definições para cada roda:

- número da roda (iniciando em zero)
- modelo da roda (nome do arquivo)
- rigidez
- amortecimento
- largura
- raio
- deslocamento da roda
- saturação do deslocamento

- velocidade relativa da suspensão
- torque de saturação
- torque constante (se a roda terá ou não um torque constante)
- torque (para o caso de a roda ter torque constante)
- deslizamento da roda
- coordenada local do centro de massa

Na apresentação visual do simulador VirtualBotz 3D, cada roda pode ter um modelo diferente ou apenas um único modelo para todas as rodas (desde que os seus atributos sejam iguais). Ao definir um modelo para uma roda, o programa verifica se ele já existe, evitando com isso a redundância de modelos. Todos os modelos devem permanecer na pasta "Obj_files" e devem ser do tipo wavefront (.OBJ) com malha triangular. Deve-se evitar o uso de modelos muito densos para não sobrecarregar o computador e, com isso, ocasionar perda de desempenho. Alguns fatores importantes que influenciam e melhoram consideravelmente o desempenho são grande quantidade de memória principal, boa placa gráfica com *chipset* NVidia ou ATI e uma quantidade razoável de memória da placa gráfica.

O modelo wavefront é muito usado em softwares de modelagem 3D, animação e efeitos visuais. É inclusive utilizado pelos softwares de modelagem 3D Alias|Wavefront Maya [44] e Blender [45], sendo este freeware. Os arquivos ".OBJ" são usados normalmente para guardar coordenadas 3D de objetos em três dimensões, mapas de texturas e outras informações úteis para padrão de imagens 3D. Dessa forma, os arquivos podem ser importados por vários programas de edição de imagem em três dimensões. Na Figura 88 tem-se um exemplo de roda desenhado no software Blender e salvo no formato wavefront, sempre com faces triangulares.

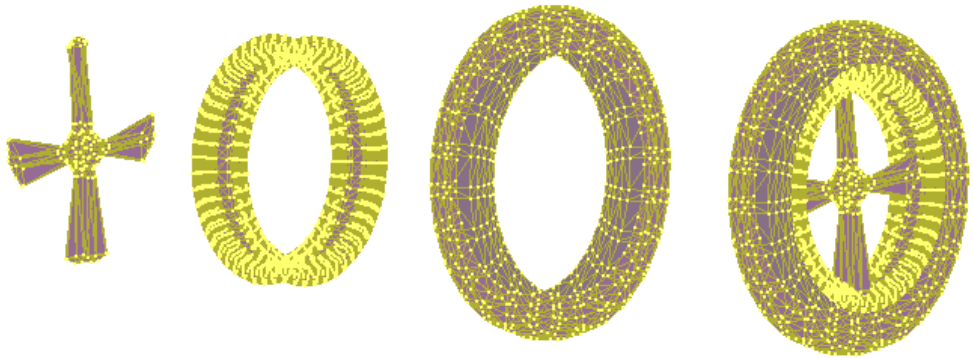


Figura 88 – Modelo de roda desenhado no software Blender com apresentação em vértices e arestas

Outro recurso implementado no simulador é a utilização de materiais coloridos para os modelos (Figura 89). Utilizando o mesmo formato de modelo (o wavefront), podem-se selecionar materiais de diferentes cores para cada grupamento de vértices/arestas e os salvar junto com os arquivos do wavefront. Os dados com materiais associados aos grupamentos de vértices/arestas ficarão no arquivo de extensão ".MTL".

Os arquivos ".MTL" são chamados biblioteca de materiais e contêm definições de um ou vários materiais, como por exemplo cor e textura. Esses arquivos são referenciados pelos arquivos wavefront e normalmente são salvos no formato ASCII. No caso de utilização de materiais texturizados para as rodas, como ilustrado no terreno, os arquivos de textura devem ficar na pasta "/*Textures*".

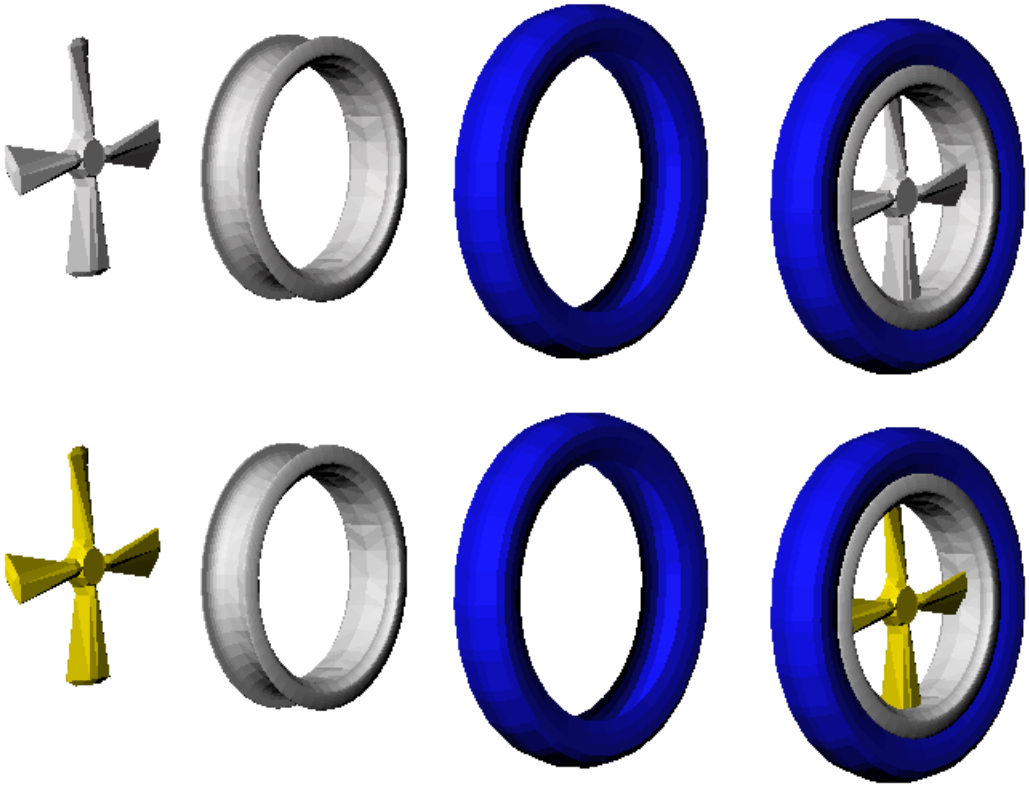


Figura 89 – Modelos de roda com definição dos materiais

O mesmo procedimento de modelagem visual da roda, descrito nos parágrafos anteriores, deve ser utilizado para o chassi do veículo robótico (Figuras 90 e 91). É importante salientar que o veículo utilizado neste estudo tem um formato de paralelepípedo, mas poderia ser tão ou mais elaborado que a roda mostrada acima. Na figura a seguir, há dois exemplos simples de chassi feitos no modelador geométrico Blender, ambos com faces triangulares.

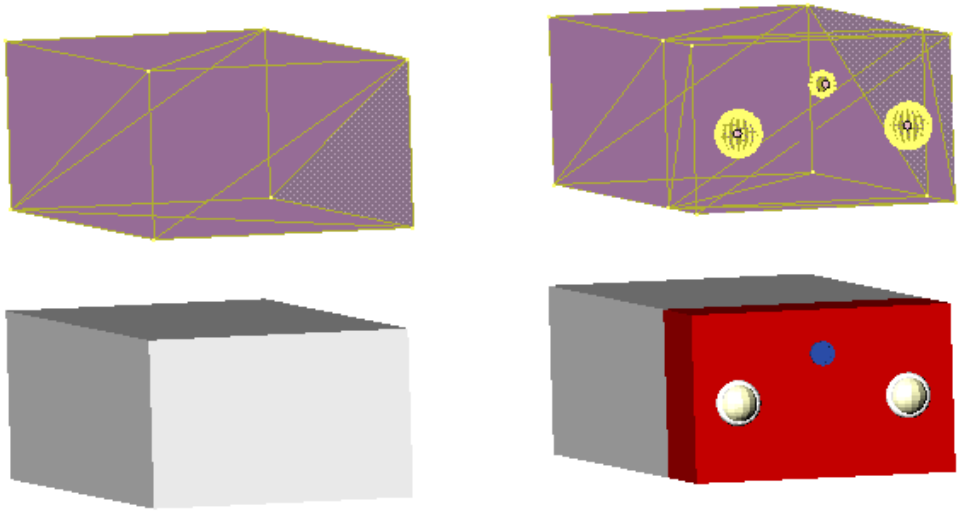


Figura 90 – Exemplos de chassi bem simples, ambos com poucos detalhes e malha triangular

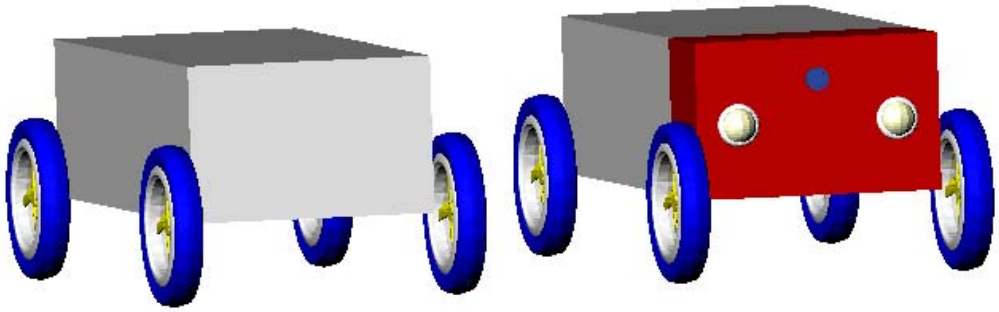


Figura 91 – Imagens da composição rodas mais chassi feita pelo simulador VirtualBotz 3D

Segue um trecho do script de simulação para configuração do veículo robótico e das rodas. O script integral encontra-se no **Anexo E**. Observe que, mesmo definindo rodas com materiais coloridos (linhas em negrito), basta informar o nome do arquivo modelo de formato wavefront (“.OBJ”), e o próprio programa se encarrega de procurar o arquivo de materiais (“.MTL”).

```
#
#####################################################################
## Dados do Veículo Robótico ########################################
robot opengl file name = chassi_2.obj
## Orientação inicial do veículo robótico
vetor n = { 1.0, 0.0, 0.0 }
vetor t = { 0.0, 1.0, 0.0 }
vetor b = { 0.0, 0.0, 1.0 }
## Posição inicial do veículo robótico
centro de massa = { -46.0, 0.0, 2.3 }
## Velocidade linear inicial
velocidade do centro de massa = { 0.0, 0.0, 0.0 }
## Velocidade angular
vetor w = { 0.0, 0.0, 0.0 }
## Centro de massa do veículo robótico
## Vetor centro de massa = { right, left, front, back, top, bottom }
right = 1.0
left = 1.0
front = 2.0
back = 2.0
top = 0.7
bottom = 0.5;
## Massa em quilos
massa = 120.0
quantidade de rodas = 4
quantidade de divisoes da roda = 18
rigidez do chassi = 1e4
amortecimento = 5e2
## Definição dos corners (quinas)
quantidade de corners = 8
## Posição dos corners na coordenada n-t-b local ----------------
corner = 0
ntb = { -back, -right, -bottom }
corner = 1
ntb = { front, -right, -bottom }
corner = 2
ntb = { front, left, -bottom }
corner = 3
ntb = { -back, left, -bottom }
corner = 4
```

```
ntb = { -back, -right, top }
corner = 5
ntb = { front, -right, top }
corner = 6
ntb = { front, left, top }
corner = 7
ntb = { -back, left, top }
#
## Rodas ----------------------------------------------------------
## Cada roda tem a sua configuração independente. Isso quer dizer
## que se no atributo "quantidade de rodas" em "Dados do Veículo
## Robótico" forem definidas seis rodas, as seis rodas deverão
## ser definidas uma a uma nesta seção.
##
## Regras:
##
##   1) A primeira roda deve começar em zero.
##   2) As coordenadas de cada roda devem seguir uma sequência
##      anti-horária. Essa sequência é fundamental para o controle
##      de estabilidade do veículo.
#
# -------------------------------------------------
roda = 0
nome do motor = Magmotor_S28-150
modelo da roda nome do arquivo = supertire_43.obj
rigidez = 1e4
amortecimento = 5e2
largura = 0.2
raio = 0.6
deslocamento h = 0.0
saturacao de h = 0.2
velocidade relativa da suspensao = 0.0
torque de saturacao = 30.0
# Torque constante: no ou yes (valendo o que está em torque,
# logo abaixo)
torque constante = no
torque = 0.0
# Deslizamento da roda: no ou yes
desliza = no
# Pneu
rigidez lateral do pneu = 3e4
```

```
rigidez longitudinal do pneu = 3e4
coordenadas local do centro de massa (x,y,z) = { -2.0, -1.1, -0.5 }
#
# ------------------------------------------------
roda = 1
nome do motor = Magmotor_S28-150
modelo da roda nome do arquivo = supertire_43.obj
rigidez = 1e4
amortecimento = 5e2
largura = 0.2
raio = 0.6
deslocamento h = 0.0
saturacao de h = 0.2
velocidade relativa da suspensao = 0.0
torque de saturacao = 30.0
# Torque constante: no ou yes (valendo o que está em torque,
# logo abaixo)
torque constante = no
torque = 0.0
# Deslizamento da roda: no ou yes
desliza = no
# Pneu
rigidez lateral do pneu = 3e4
rigidez longitudinal do pneu = 3e4
coordenadas local do centro de massa (x,y,z) = { 2.0, -1.1, -0.5 }
#
# ------------------------------------------------
roda = 2
nome do motor = Magmotor_S28-150
modelo da roda nome do arquivo = supertire_43.obj
rigidez = 1e4
amortecimento = 5e2
largura = 0.2
raio = 0.6
deslocamento h = 0.0
saturacao de h = 0.2
velocidade relativa da suspensao = 0.0
torque de saturacao = 30.0
# Torque constante: no ou yes (valendo o que está em torque,
# logo abaixo)
torque constante = no
torque = 0.0
```

```
# Deslizamento da roda: no ou yes
desliza = no
# Pneu
rigidez lateral do pneu = 3e4
rigidez longitudinal do pneu = 3e4
coordenadas local do centro de massa (x,y,z) = { 2.0, 1.1, -0.5 }
#
# ------------------------------------------------
roda = 3
nome do motor = Magmotor_S28-150
modelo da roda nome do arquivo = supertire_43.obj
rigidez = 1e4
amortecimento = 5e2
largura = 0.2
raio = 0.6
deslocamento h = 0.0
saturacao de h = 0.2
velocidade relativa da suspensao = 0.0
torque de saturacao = 30.0
# Torque constante: no ou yes (valendo o que está em torque,
# logo abaixo)
torque constante = no
torque = 0.0
# Deslizamento da roda: no ou yes
desliza = no
# Pneu
rigidez lateral do pneu = 3e4
rigidez longitudinal do pneu = 3e4
coordenadas local do centro de massa (x,y,z) = { -2.0, 1.1, -0.5 }
#
...
```

Anexo J:
Configurando os Motores das Rodas do Veículo Robótico no Script da Simulação

O aplicativo permite a configuração de *n* motores de corrente contínua (DC) no script da simulação, funcionando como uma folha de dados (*datasheet*) do fabricante do motor. Os dados reais inseridos nessa parte do aplicativo contribuem para a escolha do modelo ideal para o veículo robótico durante uma simulação. No entanto, é importante lembrar que, mesmo incluindo *n* tipos de motores, o simulador só permite a utilização de um motor por roda. Os atributos a serem configurados para cada motor são os seguintes:

- K_t – constante de torque do motor, em **Nm/A**
- K_v – constante de velocidade do motor, em **(rad/s)/V** (igual a $1/K_t$)
- R_{motor} – resistência elétrica do motor, em **Ohm**
- I_{noload} – corrente elétrica do motor, em ampéres (**A**)
- I_{max} – corrente máxima drenada pelo motor
- V_{max} – tensão elétrica máxima do motor
- caixa de redução (adimensional)
- potência máxima da bateria

Na simulação para este trabalho foi utilizado o modelo de motor **Magmotor S28-150**, mas o aplicativo permite a configuração de qualquer um dos motores mostrados na Tabela 11.

Tabela 11 – Exemplos de alguns dos dados dos fabricantes de motores [48]

				
Nome	**Etek**	**Magmotor S28-150**	**Magmotor S28-400**	**NPC T64 (w/gearbox)**
Tensão (V)	48	24	24	24
Potência (W)	11,185	2,183	3,367	834
Massa (kg)	9.4	1.7	3.1	5.9
Peso/Potência	1,190	1,284	1,086	141
I_{stall} / I_{no_load}	526	110	127	27
K_t (N·m/A)	0.13	0.03757	0.0464	0.86
K_v (RPM/V)	72	254	206	10
R_{motor} (Ω)	0.016	0.064	0.042	0.16
I_{no_load} (A)	5.7	3.4	4.5	5.5

Segue a parte do script de simulação que permite a configuração dos motores do veículo robótico. O script integral está no **Anexo E**.

```
...
#######################################################################
## Dados dos Motores ##################################################
##
## Cada roda pode ter um atuador motor com características
## específicas ou todas podem ter o mesmo tipo de atuador motor
## com as mesmas características. Entretanto, o número de atuadores
## motor não pode ultrapassar a quantidade de rodas estipuladas
## logo acima.
##
## - Para definir atuadores motor de mesma marca com diferentes
##   especificações, basta colocar nomes diferentes, por exemplo:
##       motor = 0
##       nome, modelo e versao = Magmotor S28-150 ver1.0
##       ...
##       motor = n
##       nome, modelo e versao = Magmotor S28-150 ver1.2
##
## - Os atributos abaixo serão os mesmos para todos os atuadores
##   motor definidos nesta seção.
##
## Regras:
##
```

```
##   1) O primeiro atuador motor deve começar em zero.
##   2) Não pode ser utilizado o caractere espaço em branco
##      no nome do modelo.
# ------------------------------------------------
motor = 0
nome do modelo e versao = Magmotor_S28-150
torque constante Kt[Nm/A] = 0.03757
velocidade constante Kv[(rad/s)/V] = 26,61698
resistencia total do motor Rmotor[Ohm] = 0.064
corrente Inoload[A] = 3.4
# Para limitar a potência da bateria PN3600,
# fazer Imax = 80 e Vmax = 24
corrente maxima[A] = 80
tensao maxima[V] = 36
caixa de reducao (adimensional) = 7.14
#
motor = 1
nome do modelo e versao = Magmotor_S28-400
torque constante Kt[Nm/A] = 0.0464
velocidade constante Kv[(rad/s)/V] = 21,5517
resistencia total do motor Rmotor[Ohm] = 0.042
corrente Inoload[A] = 4.5
# Para limitar a potência da bateria PN3600,
# fazer Imax = 80 e Vmax = 24
corrente maxima[A] = 80
tensao maxima[V] = 36
caixa de reducao (adimensional) = 7.14
...
#
## Rodas ----------------------------------------------------
...
roda = 0
nome do motor = Magmotor_S28-400
...
roda = 1
nome do motor = Magmotor_S28-150
...
roda = 2
nome do motor = Magmotor_S28-150
...
roda = 3
nome do motor = Magmotor_S28-400
```

Anexo K:
Habilitando no Script da Simulação a Geração de Dados da Simulação Corrente do Veículo Robótico em Arquivos

O aplicativo gera algumas saídas de dados em arquivos do tipo texto. Para a escolha dos tipos de dados, basta localizar no script a área “geração de arquivos de dados” e selecionar o necessário. Os nomes dos arquivos podem ser alterados, porém a gravação será sempre na pasta “Report”.

Tipos de saída de dados gerados pelo aplicativo VirtualBotz 3D:

- Posição do centro de massa do veículo robótico
- Posição do centro de massa das rodas do veículo robótico
- Torque das rodas
- Sinal de controle das rodas

```
...
#####################################################################
# Geração de arquivos de dados ######################################
#
# - O nome do arquivo NÃO DEVE conter ponto, extensão, espaço em
#   branco ou qualquer outro caractere gráfico linguístico, como
#   acentuação, entre outros.
# - Os arquivos gerados serão gravados na pasta "Report".
# - Para desligar a saída dos dados, basta digitar a palavra "off"
#   no início da linha referente ao dado a ser desligado. Para
#   ligar, basta digitar "on".
#
off,centro de massa do veiculo = mobile_xCM.txt
off,centro de massa das rodas do veiculo = wheels_xCM.txt
on,torque das rodas = wheels_torques.txt
off,sinal de controle das rodas = wheels_ControlSign.txt
##
...
```

Os arquivos seguem um padrão para serem facilmente aplicados a programas gráficos como Grapher e similares ou planilhas gráficas como Microsoft Excel e similares. As Figuras 92, 93, 94 e 95 foram geradas no Microsoft Excel a partir de dados salvos durante uma simulação de um veículo robótico.

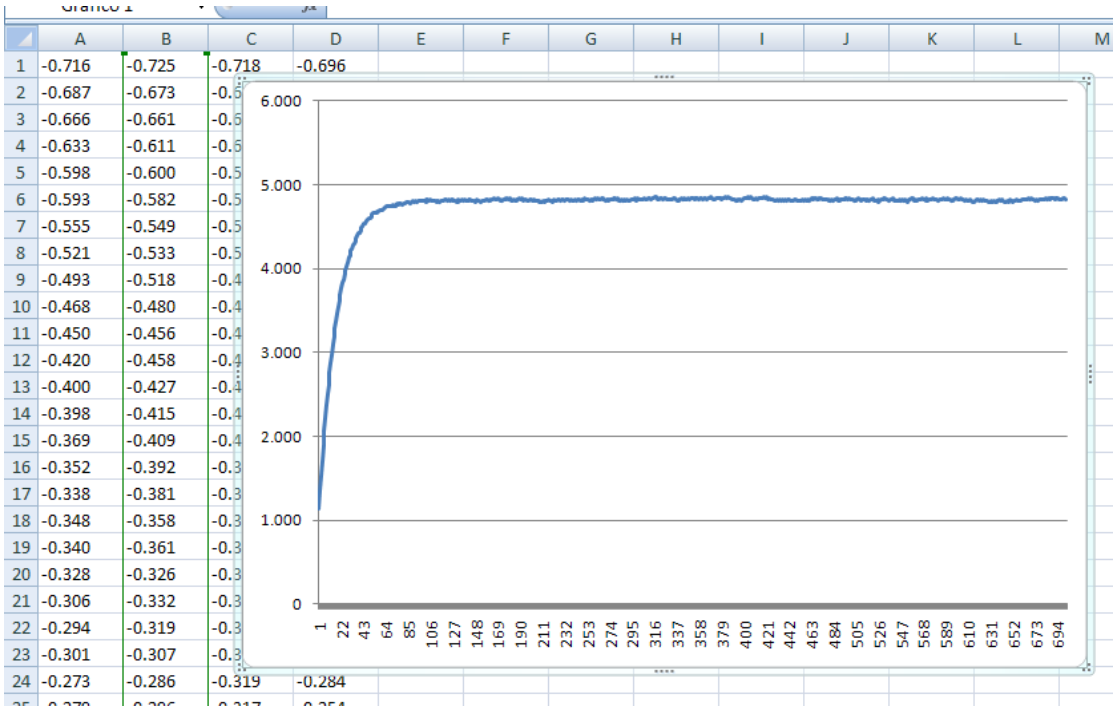


Figura 92 – Visualização em Microsoft Excel do gráfico gerado pelo torque de uma das rodas do veículo robótico em uma simulação

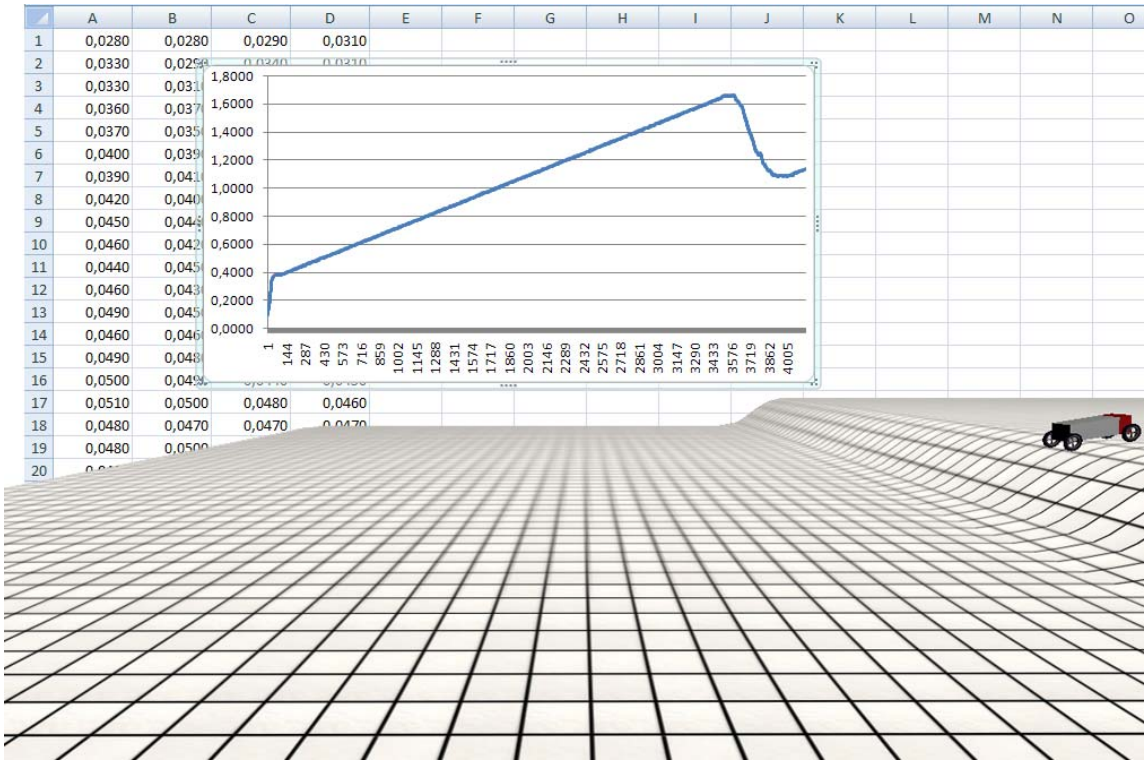


Figura 93 – Visualização em Microsoft Excel do gráfico do sinal de controle PID de uma das rodas do veículo robótico durante uma simulação

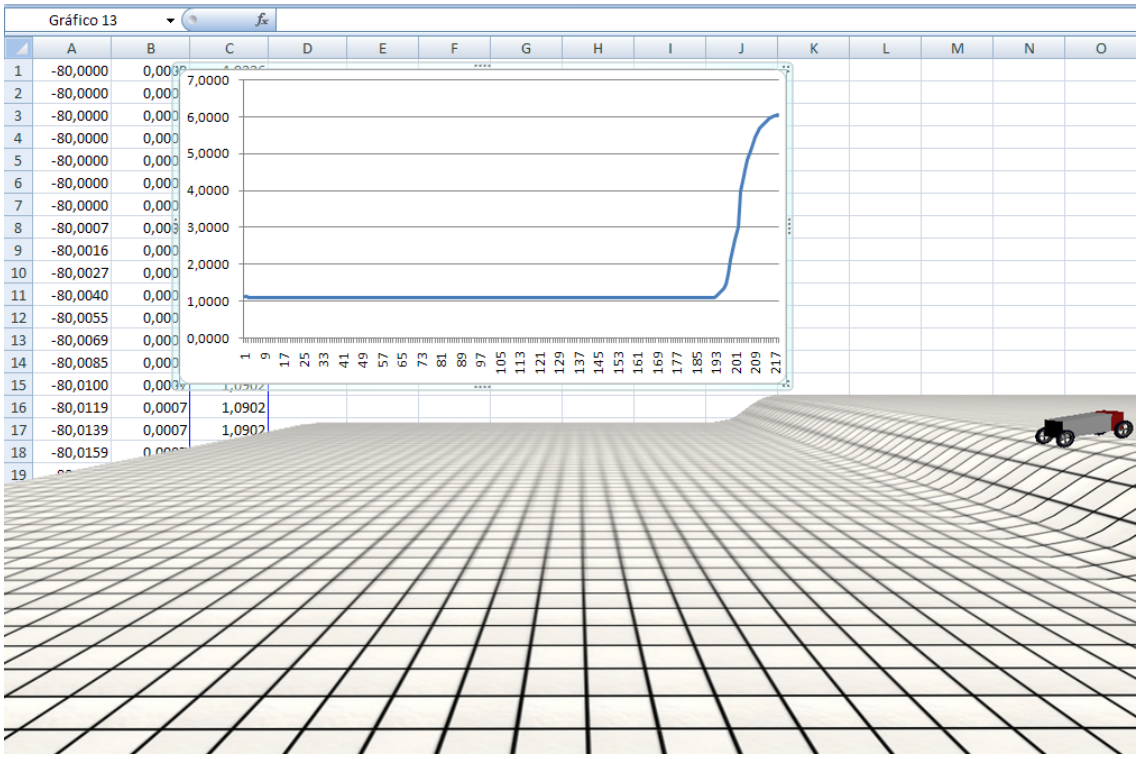


Figura 94 – Visualização em Microsoft Excel do gráfico do vetor ***z*** do centro de massa do veículo robótico em uma simulação

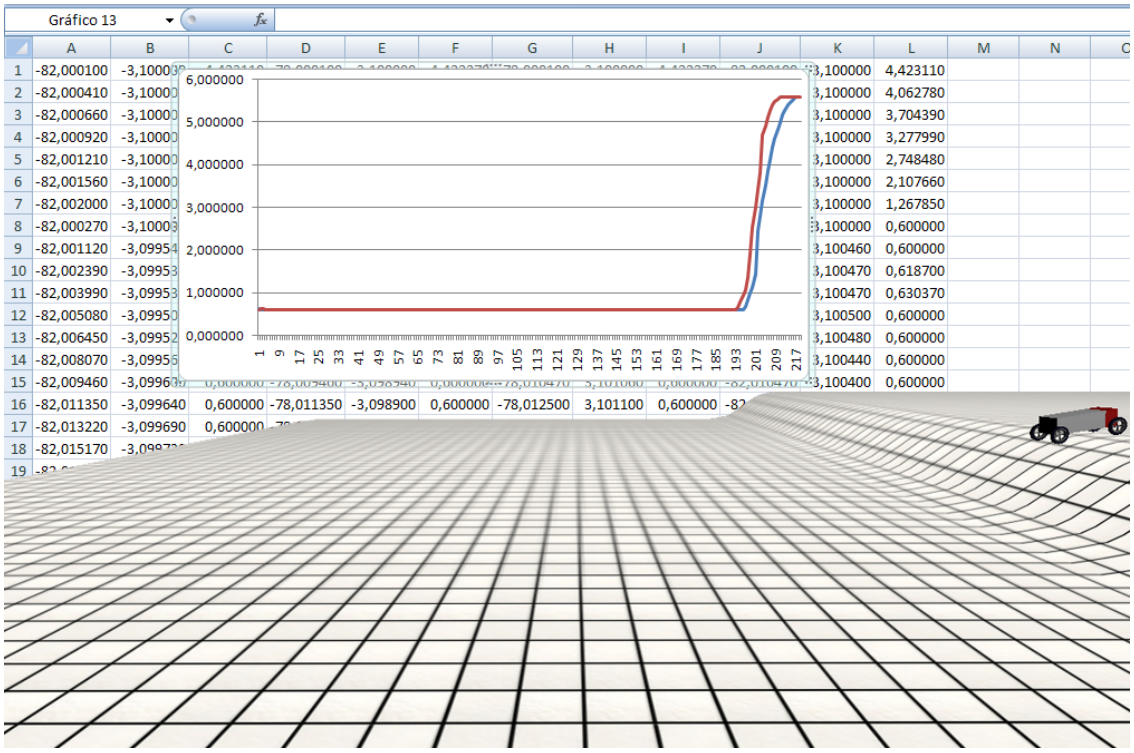


Figura 95 – Visualização em Microsoft Excel do gráfico do vetor ***z*** do centro de massa de uma das rodas dianteiras do veículo robótico, em vermelho, e uma das rodas traseiras, em azul, durante uma simulação