Capítulo

3

Introdução ao V-REP: Uma Plataforma Virtual para Simulação de Robôs

Luís B. P. Nascimento, Diego S. Pereira, Vitor G. Santos,
Daniel H. S. Fernandes, Pablo J. Alsina

Abstract

Practical experiments in robotics are usually difficult task to perform, since many robots are reasonably expensive devices. Thus, a plausible solution is to use simulator to test if the developed system works satisfactorily. In this sense, the Virtual Robot Experimentation Platform (V-REP) is an interesting tool that enables to model robots and perform simulations in a simple and intuitive way, focused in robotics areas such as path planning, mapping, controls, among others. Thus, this chapter presents an introductory course about V-REP, focusing on basic concepts of the plataform and some simulation examples.

Resumo

A realização de experimentos práticos na robótica nem sempre é trivial devido aos altos custos nessa área, porém, é comum a utilização de simuladores para possibilitar a realização de experimentos. A Plataforma Virtual para Experimentação Robótica (Virtual Robot Experimentation Platform - V-REP) é uma interessante ferramenta que viabiliza a modelagem de robôs, assim como a realização de experimentos simulados de maneira simples e didática, podendo atuar em áreas da robótica como planejamento de caminho, mapeamento, controle, dentre outras. Dessa forma, este capítulo apresenta uma visão introdutória sobre a plataforma V-REP apresentando conceitos básicos da ferramenta, assim como alguns exemplos de simulações.

3.1. Introdução

O aumento exponencial do poder de processamento dos computadores combinado com o vasto crescimento de pesquisas na área de robótica não só estimulou, como também tornou viável a utilização de simuladores capazes de reproduzir com qualidade cenários

2D/3D. Além disso, os simuladores tornam possível a realização de experimentos em tempo real permitindo ao pesquisador avançar mais rapidamente com seus resultados e, inicialmente, ter independência da utilização do robô real (*hardware*) que, em muitos casos, é um dos componentes mais onerosos do cenário em estudo.

Atualmente, existem diversas plataformas de simulação robótica disponíveis no mercado, tais como o Open HRP, Gazebo, Webots, V-REP, entre outras [Rohmer et al. 2013]. Contudo, a Plataforma Virtual de Experimentação Robótica (*Virtual Robot Experimentation Platform*), tratada na literatura apenas como V-REP [1], destaca-se por se tratar de uma ferramenta fácil e didática, o que atrai usuários com pouco ou sem conhecimentos na área de robótica.

A plataforma V-REP busca atender todos os requisitos de uma estrutura de simulação versátil e escalável que, além de oferecer abordagens tradicionais, já encontradas em outros simuladores, possui itens adicionais, pois com uma arquitetura de controle distribuída, cada objeto/modelo V-REP pode ser controlado individualmente através de um *script*, um plugin, uma API cliente remota, entre outras soluções [Rohmer et al. 2013].

É possível encontrar diversos trabalhos que fizeram uso desta plataforma, entre eles o trabalho de [Obdržálek 2017] apresenta uma estratégia para utilizar um sistema multi-agente formado por Veículos Aéreos Não Tripulados (VANTs), enquanto o trabalho de [Knoll et al. 2017] propõe um esquema de navegação para robôs terrestres usando um conjunto de sensores de rádio frequência. Em [Tanberk and Tükel 2017], o simulador é utilizado para efetuar testes com um robô capaz de interpretar dados oriundos de um sensor Kinect na tomada de decisão para movimentação de peças em um jogo de xadrez. Finalmente, o trabalho de [Lima et al. 2018] apresenta a utilização da V-REP para auxiliar no processo de aprendizagem de robótica como uma ferramenta educacional. Isso é possível graças à sua versão educacional (V-REP PRO EDU) disponibilizada gratuitamente.

Diante disso, percebe-se a versatilidade do simulador em atuar com diferentes plataformas robóticas e em diversas aplicações, isso permite aos pesquisadores, alunos e professores validarem suas ideias, seja para a obtenção de resultados iniciais em pesquisas utilizando ambientes simulados, ou como ferramenta educacional para o ensino-aprendizagem de conceitos dentro da robótica. Nesse sentido, esse capítulo apresenta uma visão introdutória da plataforma V-REP a fim de destacar algumas características e funcionalidades interessantes, além de descrever como prototipar um robô terrestres utilizando os próprios objetos disponíveis na plataforma.

3.2. Virtual Robot Experimentation Platform (V-REP)

Desenvolvido pela Coppelia Robotics, o simulador robótico V-REP possui um ambiente de desenvolvimento integrado e baseia-se em uma arquitetura distribuída onde cada objeto/modelo pode ser controlado individualmente através de um *script*, um *plugin*, um nó ROS (*Robot Operating System*), um cliente de uma API (*Application Programming Interface*) remota ou uma solução personalizada. Tais itens tornam este simulador versátil

[1] Site oficial do V-REP: http://www.coppeliarobotics.com/

e ideal para diversas aplicações. Os controladores podem ser escritos em C/C++, Python, Java, Lua, Matlab ou Octave. A versão atual da V-REP é a 3.6.2, disponibilizada em 25 de junho de 2019 [Robotics 2019].

3.2.1. Onde encontrar o simulador V-REP?

A plataforma de simulação V-REP está disponível para diversos sistemas operacionais. O software está disponível para download através do link (http://www.coppeliarobotics.com/downloads.html). A Figura 3.1 mostra as versões do software disponíveis para download.

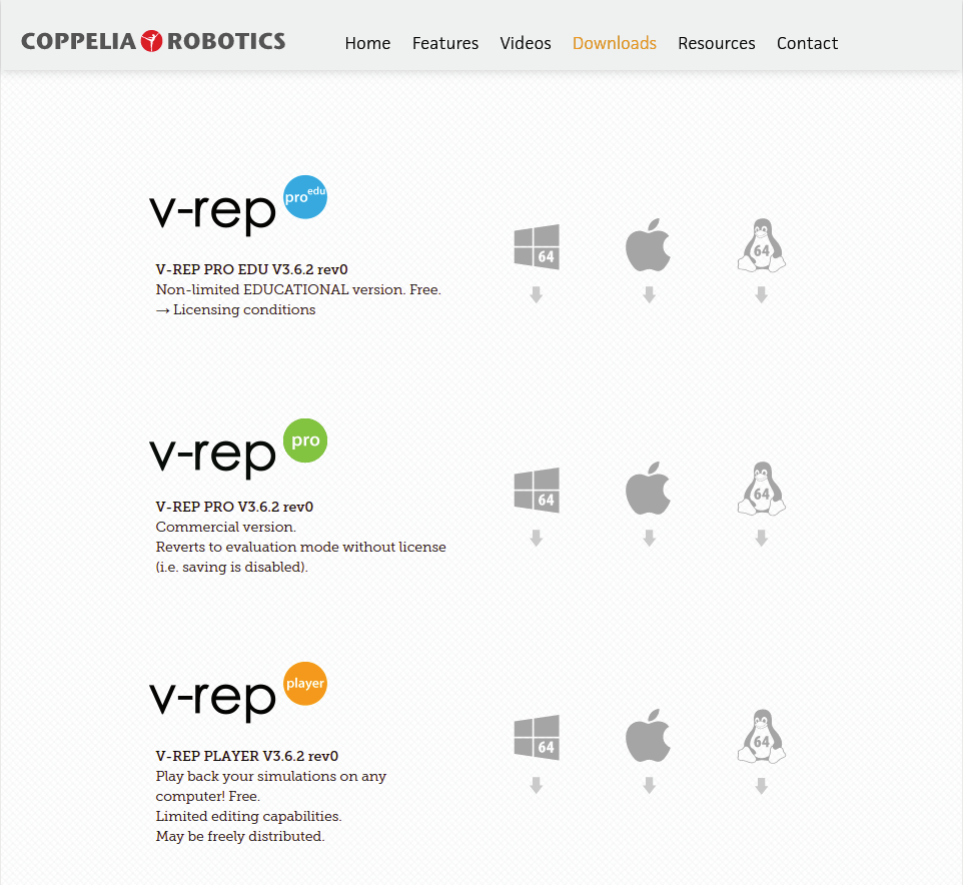


Figura 3.1. Página de download do V-REP.

No site do V-REP estão disponíveis três versões do software (Pro EDU, Pro e Player) para Windows, Mac e Sistemas Linux. A primeira (Pro EDU) é uma versão gratuita, disponível para fins educacionais (essa é a versão que utilizaremos em todos os exemplos). A segunda versão mostrada é a versão Pro, disponível para fins comerciais, sendo necessária a aquisição de uma licença de software. A terceira versão (V-REP Player) é uma ferramenta utilizada para reproduzir e interagir com simulações.

3.2.2. Interface do Usuário

A interface de usuário do simulador V-REP é composta por um conjunto de elementos. Os principais elementos são: o terminal, a janela de aplicação e *dialogs* (caixas de diálogo).

O primeiro deles, o terminal, é ocultado quando iniciado em ambiente Windows. Nas demais plataformas, Linux e Mac, é possível acompanhar a inicialização da plataforma de simulação, o carregamento de plugins, dentre outras informações, inclusive mostra se o sistema foi inicializado de forma correta. Este capítulo não irá apresentar maiores detalhes para iteração com o terminal.

A janela de aplicação ou janela principal (interface gráfica) do simulador V-REP agrupa a grande maioria das funcionalidades do simulador e, através dela, é possível exibir, editar, simular e interagir com uma cena. O usuário também pode editar e interagir com uma cena ajustando as configurações ou parâmetros presentes em uma caixa de diálogo. Cada caixa de diálogo agrupa um conjunto de funções relacionadas ou funções que se aplicam a um mesmo objeto de destino. O conteúdo de um diálogo pode ser sensível ao contexto (por exemplo, dependente do estado de seleção do objeto)[Robotics 2019]. A Figura 3.2 retrata o ambiente da janela principal do simulador V-REP.

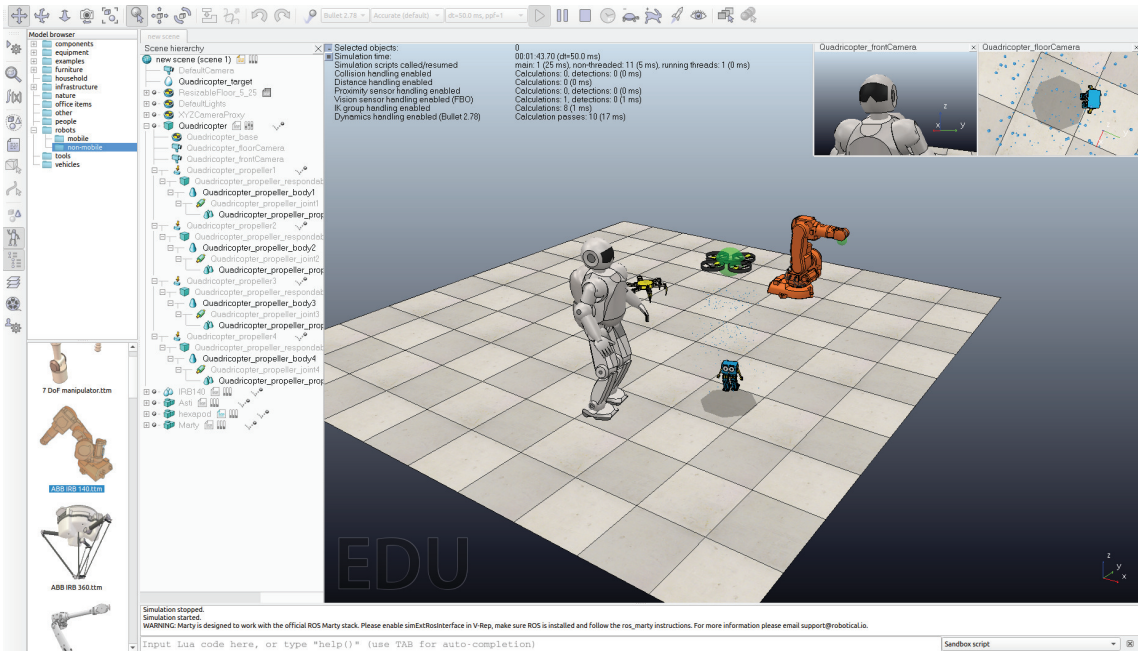


Figura 3.2. Exemplo da interface V-REP com diversos tipos de robôs.

De forma geral, o simulador V-REP possui uma interface gráfica agradável e intuitiva formada por menus, barras de ferramentas e *dialogs* que permitem ao usuário inserir, remover, movimentar, editar objetos de forma simples, fazendo uso apenas do *mouse*. No canto esquerdo da imagem é possível observar o navegador de modelos (*Model Browser*). A partir desse navegador é possível encontrar diversos modelos de robôs já prontos para ser utilizados em simulações. No V-REP, os robôs disponíveis estão divididos em móveis e não-móveis, sendo possível adicioná-los à cena arrastando e soltando (*drag-and-drop*). O navegador de modelos com alguns robôs pode ser observado na Figura 3.7.

Como comentado anteriormente, outros elementos importantes que facilitam a criação de simulações e que estão presentes na interface principal do V-REP são

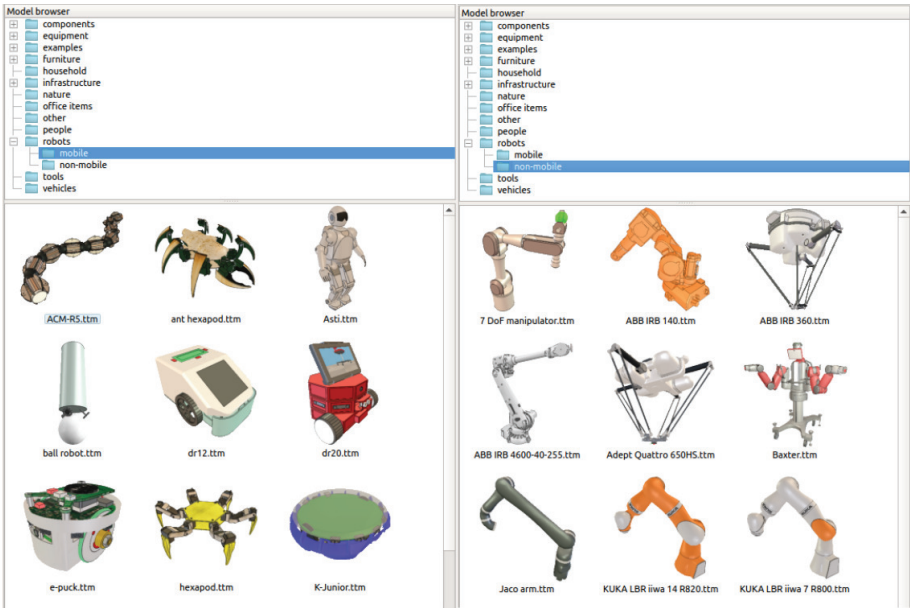
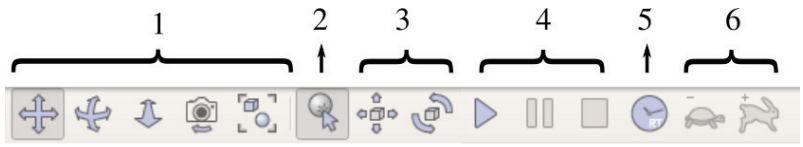


Figura 3.3. Exemplo da interface V-REP com diversos tipos de robôs.

as barras de ferramentas. Nelas é possível encontrar botões com funções úteis que tornam o manuseio do simulador mais simples, já que as funcionalidades presentes nas barras estão contidas dentro de menus internos, o que tornaria mais difícil tarefas básicas que necessitem ser realizadas, tais como, mudança na angulação da câmera ou reposicionamento de objetos dentro de cena. A Figura 3.4 mostra duas barras de ferramentas contendo alguns botões com as funcionalidades mais úteis.



Barra de ferramentas 1

Barra de ferramentas 2

Figura 3.4. Algumas funções encontradas nas barras de ferramentas do V-REP.

A Figura 3.4 mostra duas barras de ferramentas contendo algumas funções importantes para a criação de simulações no V-REP. Na barra de ferramentas 1 é possível encontrar algumas funções diretamente relacionadas às simulações, como seguem:

1. Navegação da câmera
2. Seleção por clique

3. Manipulação de objetos (translação e rotação)
4. Botões de simulação (iniciar, pausar e parar)
5. Simulação em tempo real
6. Controle de velocidade das simulações.

A barra de ferramentas 2 apresenta algumas funcionalidades mais avançadas, assim como algumas configurações, como podem ser listadas a seguir:

7. Configurações da simulação
8. Propriedade do objeto selecionado
9. Módulo de cálculos (ex: distâncias)
10. Coleções
11. Scripts
12. Editor de formas
13. Editor de caminhos selecionados
14. Objetos Selecionados
15. Navegador de Modelos
16. Hierarquia de cena
17. Camadas
18. Gravador de vídeos
19. Configurações do usuário

É possível ver com detalhes a função de cada item no Manual do Usuário disponível na página oficial do V-REP (http://www.coppeliarobotics.com/helpFiles/). Além disso, alguns robôs, tais como o IRB140 e o Asti, possuem uma interface própria que permite ao usuários enviar informações aos robôs e interagir com o cenário em tempo de execução durante uma simulação.

Conforme mostrado, uma simulação V-REP pode ser iniciada, pausada ou encerrada através de botões na barra de menu. Internamente, o simulador gerencia estados intermediários adicionais necessários para informar a *scripts* ou programas o que ocorrerá a seguir, contudo, isto é transparente para o usuário. Vale destacar que existe uma preocupação do simulador em manter o tempo de simulação sincronizado com o tempo real, entretanto, em alguns casos isto não é possível devido a complexidade da simulação, dos recursos computacionais disponíveis e das próprias configurações atribuídas pelo usuário. É possível observar na Figura 3.5 que são exibidas diversas informações durante o processo de simulação. Elas referem-se as próprias configurações atribuídas a cena, como também aos objetos presentes nela.

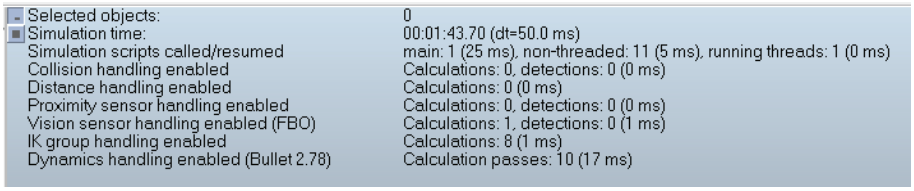


Figura 3.5. Informações da Cena em execução no V-REP.

3.2.3. Cenas e Modelos

Cenas e modelos são os principais elementos de simulação V-REP. Um modelo é um sub-elemento de uma cena e uma cena pode conter vários modelos. Um modelo só pode ser executado quando incluído em uma cena, o Asti, robô humanoide de cor branca apresentado anteriormente, é um exemplo de modelo. A plataforma V-REP permite que pesquisadores desenvolvam modelos e importem para seu ambiente de simulação.

Todos os itens que compõe uma cena estão organizados de forma hierárquica, compondo a hierarquia de cena (*scene hierarchy*). A hierarquia da cena, apresentada na Figura 3.6, exibe o conteúdo de uma cena, ou seja, todos os objetos presentes na respectiva cena. Essa hierarquia é feita em uma estrutura de árvore e representa a relação pai-filho entre todos os objetos. No exemplo a seguir, a cena com nome *new scene* possui um modelo Quadricóptero adicionado. É possível observar na Figura que ele está em um nível inferior ao objeto new scene, logo é chamado nó filho. De forma análoga, os objetos Quadricopter_floorCamera e Quadricopter_frontCamera são nós filhos do nó Quadricopter, que é seu respectivo nó pai.

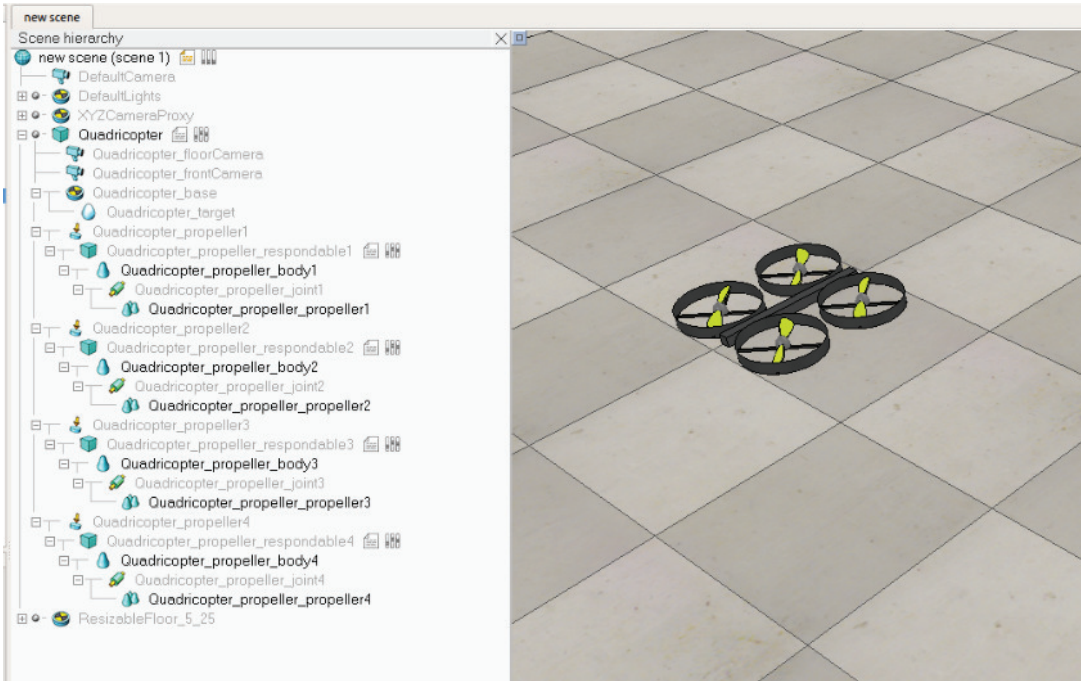


Figura 3.6. Exemplo de Hierarquia de Cena no V-REP.

3.2.4. Objetos de Cena

No simulador V-REP existem diversos elementos dos quais é possível construir uma cena de simulação, são os chamados Objetos de Cena (*scene objects*). Esses objetos são visíveis e possuem uma representação em três dimensões. Além disso, possuem papeis fundamentais para garantir a qualidade da simulação. Os principais objetos são apresentados a seguir:

- *Shapes*: São as formas básicas encontradas no V-REP, como por exemplo, cubo, cilindros, esferas, etc.
- *Joints*: São objetos utilizadas para a modelagem de robôs para prover movimento, ex: junta rotacional, junta prismática, etc.
- *Graphs*: São gráficos utilizados para visualizar e gravar dados durante as simulações.
- Sensores: Objetos utilizados para percepção do ambiente (Sensores de proximidade, Sensores de Visão e Sensores de Força)
- *Path*: Esse objeto define caminhos ou trajetórias no espaço.

Para a adição de objetos ao cenário, basta ir ao menu Add da barra de menus e escolher qual objeto, ou simplesmente, clicar com o botão esquerdo do mouse sob o cenário e selecionar Add. A Figura 3.7 mostra alguns objetos adicionados à cena: Três formas primitivas (Esfera, Cubo e um Cilindro) e um objeto *Graph*.

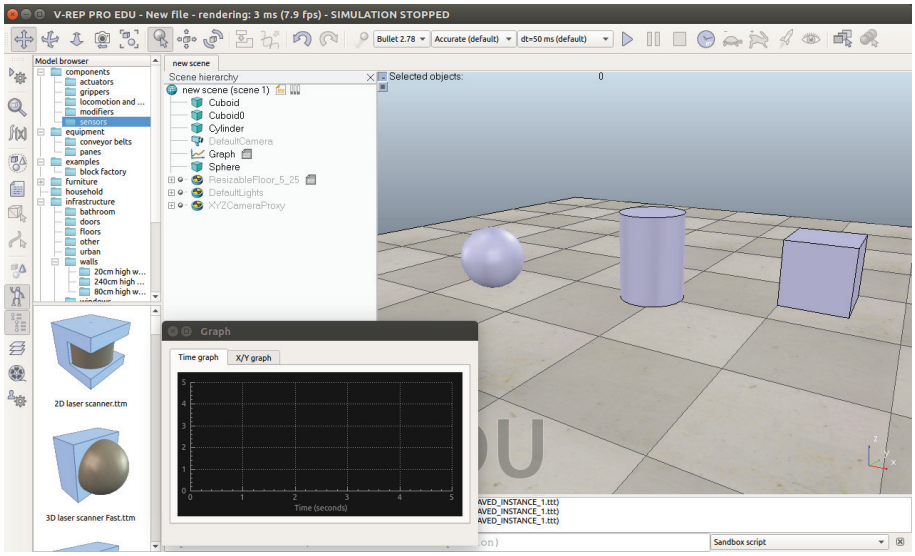


Figura 3.7. Objetos em cena. Algumas formas primitivas adicionadas ao cenário.

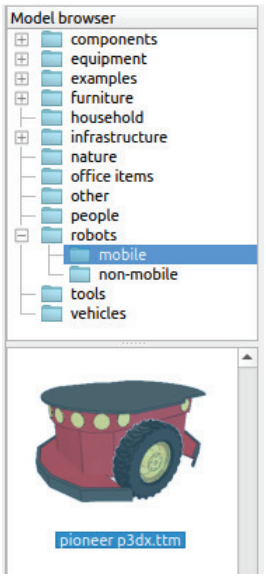
Os objetos do tipo *shapes* são formas que possibilitam a modelagem de outros elementos na cena, como por exemplo, paredes, obstáculos, partes de robôs, e etc. Na Seção 3.5 serão apresentados alguns passos para a construção de um simples robô utilizando alguns objetos de cena básicos.

3.2.5. Exercício 01

O cenário a seguir tem como objetivo apresentar uma primeira simulação com uso da plataforma V-REP. Para tal, será feito uso de um robô modelo Pionner P3DX, já disponibilizado pelo V-REP, sem a necessidade de nenhuma configuração adicional. Será construída uma barreira impedindo que o robô saia do cenário. Dessa maneira, os seguintes passos devem resultar no cenário descrito.

PASSO 01: Criar uma nova *cena*;

PASSO 02: Inserir um robô *Pioneer P3DX* ao cenário criado. ;



PASSO 03: Inserir quatro *resizable concret blocks* e redimensona-los por meio do costumizador do bloco, de tal maneira que o espaço de navegação que o robô está inserido seja limitado;

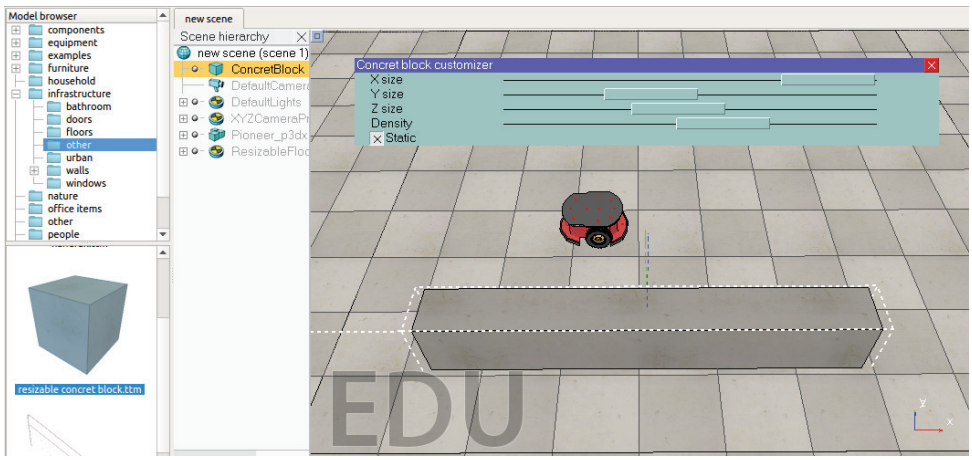


Figura 3.8. Selecionando o *resizable concret block*.

PASSO 04: Finalmente, execute a simulação e observe o comportamento do

robô. A Figura 3.9 retrata o robô P3DX já inserido no espaço de trabalho adequadamente limitado pelos blocos para restringir sua área de circulação.

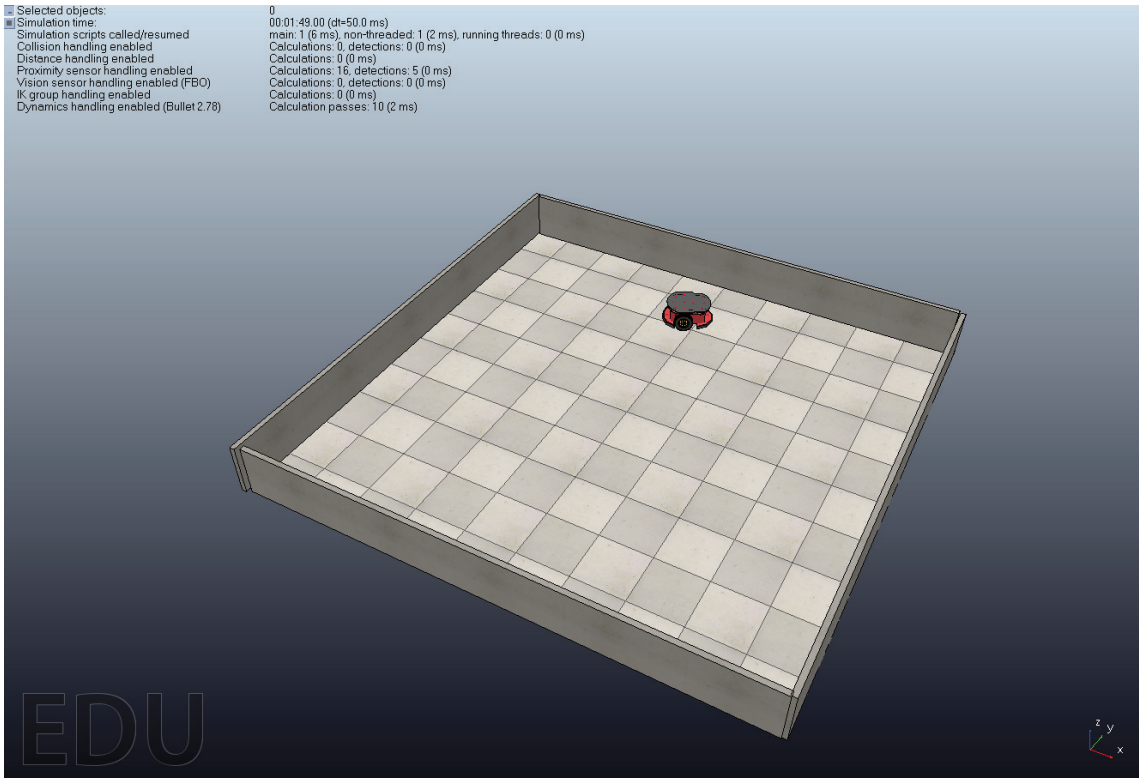


Figura 3.9. Exercício 01 em execução.

QUESTIONAMENTO 01: Qual a estratégia utilizada pelo robô para evitar colisões? Onde é possível encontrá-la?

3.3. Scripts

Um script é considerado uma entidade utilizada para controlar um modelo. O V-REP possui um interpretador de scripts integrado onde, por padrão, a linguagem de programação suportada é Lua[2]. A linguaguem Lua permite programação procedural, programação orientada a objetos, programação funcional, programação orientada a dados e descrição de dados. Ela é considerada uma linguagem ideal para configuração, automação e rápida prototipagem, amplamente utilizada em aplicações industriais, como o Adobe's Photoshop Lightroom, e jogos, por exemplo, World of Warcraft e Angry Birds.

3.3.1. Exercício 02 - Codificando um robô com linguagem Lua

O objetivo desse cenário é realizar um exemplo prático para demonstrar como os objetos do ambiente de simulação V-REP são controlados por um script escrito na linguagem de programação Lua. O ponto de partida desse exercício é a conclusão do Exercício 01.

PASSO 01: Abra o arquivo referente ao Exercício 01;

[2]https://www.lua.org/home.html

PASSO 02: Vá até a Hierarquia de Cena, selecione o robô Pioneer P3DX e dê dois cliques no ícone script, localizado ao lado direito do nome Pioneer_p3dx, conforme Figura 3.10;

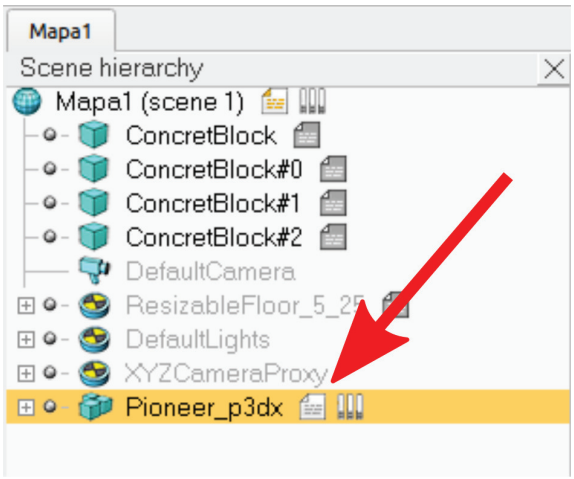


Figura 3.10. Ícone Script vinculado ao modelo Pioneer P3Dx.

PASSO 03: Apague todo o código fonte encontrado na janela e insira o código Lua apresentado seguir. Tenha atenção na identação e na escrita do script;

```
if (sim_call_type==sim.syscb_init) then
    motorLeft=sim.getObjectHandle("Pioneer_p3dx_leftMotor")
    motorRight=sim.getObjectHandle("Pioneer_p3dx_rightMotor")
end

if (sim_call_type==sim.syscb_actuation) then
    sim.setJointTargetVelocity(motorLeft,1)
    sim.setJointTargetVelocity(motorRight,2)
end
```

Script 3.1. Acionando os motores do robô Pioneer em Lua

PASSO 04: Confira se seu script está conforme a imagem 3.11. Em caso positivo, execute a simulação e observe o comportamento do robô.

QUESTIONAMENTO 02: Como é possível justificar o comportamento do robô tomando como base no código fonte utilizado para controla-lo? Qual a solução para o movimento que o robô executa ser no sentido horário?

3.4. Cliente de API Remota

A plataforma de simulação V-REP trata-se de uma ferramenta altamente flexível, sendo possível personalizar vários aspectos de uma simulação. Um dos itens que está diretamente relacionado a tal característica é a utilização de APIs. Por meio de determinadas APIs, é possível controlar e manipular aspectos da cena ou objetos a partir de diferentes maneiras, tais como, por meio de *scripts* ou de *plugins*. Uma interessante API que pode ser utilizada pelo simulador V-REP é a API Remota, que torna possível

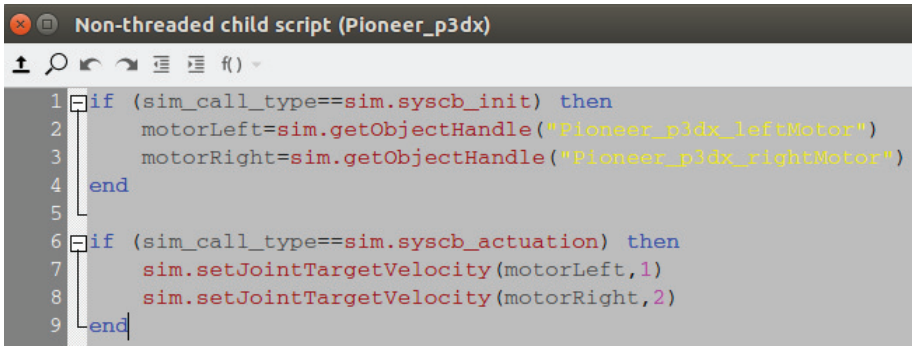


Figura 3.11. Script Lua concluído.

a comunicação do V-REP em tempo de execução, com diferentes tecnologias externas, além de tornar possível controle de robôs de maneira remota.

A API remota é composta por cerca de 100 funções específicas e uma função genérica que pode ser chamada a partir da aplicação cliente desenvolvida na linguagem escolhida pelo desenvolvedor. Essas funções remotas interagem com o V-REP através de soquetes de comunicação. Para cada cliente que realiza comunicação com o servidor V-REP, um identificador é atribuído. Todas as funções das APIs suportadas pelo V-REP estão disponíveis *online*, como por exemplo, as funções para utilização em conjunto com o Python estão disponíveis no link [3].

A Figura 3.12 ilustra um cenário típico de comunicação via API remota utilizando linguagens de programação distintas. Neste caso, a partir do momento que o servidor V-REP tem sua simulação iniciada um ponto de comunicação é criado através do endereço ip:A.A.A.A e porta 19999, enquanto os clientes, normalmente, fazem uso de configurações dinâmicas, das quais o ip é atribuído através de um serviço de rede a qual o cliente está conectado e a porta pelo sistema operacional nativo.

3.4.1. Exercício 03 - Utilizando uma API Remota

Com base no cenário dos exercícios anteriores, faremos uma pequena modificação com o objetivo de controlar o robô de maneira remota através de um script em linguagem Python. Para possibilitar essa comunicação, é necessário adicionar uma linha de código ao *script* principal para inicializar o servidor V-REP e permitir a chamada de clientes remotos. A linha de código que segue abaixo habilita o servidor no V-REP e deve ser adicionada na função sysCall_init(), como mostra a Figura 3.13.

```
1 simRemoteApi.start(19999)
```

Além de modificar o *script* principal da cena, uma modificação no *script* do robô deve ser realizada para impedir que o código original seja executado ao invés do código remoto. Dessa forma, sugere-se apagar todo o conteúdo do *script* atribuído ao robô. Uma outra forma é acessar o botão Scripts na barra de ferramentas e desabilitar o *script* do robô.

[3] http://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsPython.htm

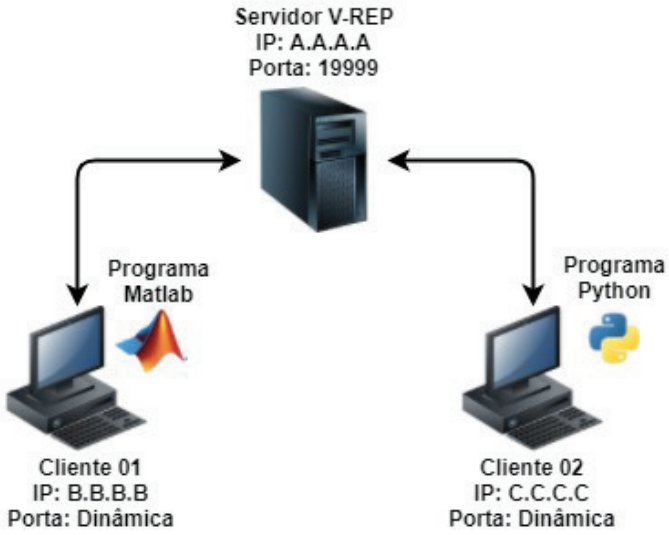


Figura 3.12. Comunicação típica utilizando API remota e V-REP.

Main script (customized)

```
10
11 function sysCall_init()
12     sim.handleSimulationStart()
13     sim.openModule(sim.handle_all)
14     sim.handleGraph(sim.handle_all_except_explicit,0)
15     simRemoteApi.start(19999)
16 end
17
```

Figura 3.13. Arquivo com o script em Python e arquivos da API Remota.

É importante ressaltar que cada cliente deve fazer da API para determinada tecnologia (Matlab, Python, Java, etc) e para seu sistema operacional (arquivos remoteAPI.so, remoteApi.dll e remoteApi.dylib para Linux, Windows e Mac, respectivamente). Isso é feito por meio da adição de arquivos de configuração dentro do mesmo diretório do programa responsável em fazer as chamadas as servidor V-REP. As APIs para cada tecnologia estão disponíveis dentro do diretório de instalação do V-REP, na pasta /programming/remoteApiBlidings/. Nosso exemplo foi realizado no Linux e o nome do arquivo do nosso script é testePython.py. A Figura 3.14 mostra os arquivos necessários para essa simulação.

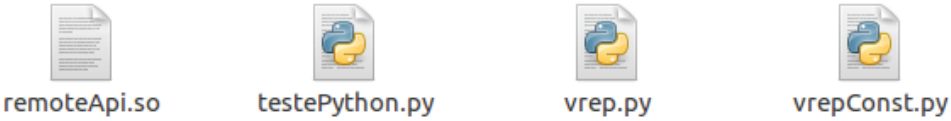


Figura 3.14. Arquivo com o script em Python e arquivos da API Remota.

O script abaixo deve ser adicionado ao arquivo *testePython.py* para inicializar a conexão com o servidor e acionar os motores. O valor da variável clientID indica se a comunicação com o servidor está de fato ocorrendo. A função simxGetObjectHandle()

obtém uma referência de um objeto dentro do simulador, por isso é importante que o segundo parâmetro represente o nome exato do objeto referente ao simulador. O terceiro parâmetro está relacionado ao modo como a troca de mensagens ente cliente e servidor ocorre, então é recomendado seguir estritamente o parâmetro indicado na documentação. Por fim, a função simxSetJointTargetVelocity() deve indicar a velocidade e a ser aplicada e roda que receberá o sinal.

```
1  import vrep
2  import time
3
4  clientID = vrep.simxStart('127.0.0.1',19999,True,True,5000,5)
5
6  if (clientID == 0):
7    print('Conectado!')
8
9  returnC, LwMotor = vrep.simxGetObjectHandle(clientID,'
       Pioneer_p3dx_leftMotor',vrep.simx_opmode_oneshot_wait)
10 returnC, RwMotor = vrep.simxGetObjectHandle(clientID,'
       Pioneer_p3dx_rightMotor',vrep.simx_opmode_oneshot_wait)
11 vrep.simxSetJointTargetVelocity(clientID,LwMotor,1,vrep.
       simx_opmode_streaming)
12 vrep.simxSetJointTargetVelocity(clientID,RwMotor,1,vrep.
       simx_opmode_streaming)
13
14 time.sleep(1)
```

Script 3.2. Acionando os motores do robô Pioneer em Python

No instante em que o código do cliente é executado o cenário passa fazer uso das chamadas remotas e atender as requisições que lhe são feitas, sendo assim, o robô deve acionar as rodas e seguir em linha reta.

3.5. Construindo meu primeiro robô

O último conteúdo a ser apresentado neste capítulo será um breve exemplo para a modelagem um robô móvel simples. Para isso, inicialmente deve-se criar uma nova cena. Alguns passos devem ser seguidos para a modegem desse robô como podem ser observados nos itens abaixo:

- **Passo 1 - Criação do Corpo do Robô**
  1. Add (botão direto do mouse) → Primitive Shape → Cuboid
  2. Dimensão: (x = 0.4, y = 0.2, z = 0.1)
  3. Nos botões localizados na barra de ferramentas (item 3 na Figura 3.4):
     - Posição: (x = 0, y = 0, z = 0.1)
     - Orientação: (Alpha = 0, Gamma = 0, Beta = 0)
  4. Renomeie o *Cuboid* como *Carro*
  5. Em Scene Object Properties (item 8 na Figura 3.4) → Common, Marcar as opções Collidable, Measurable, Detectable e Renderable.

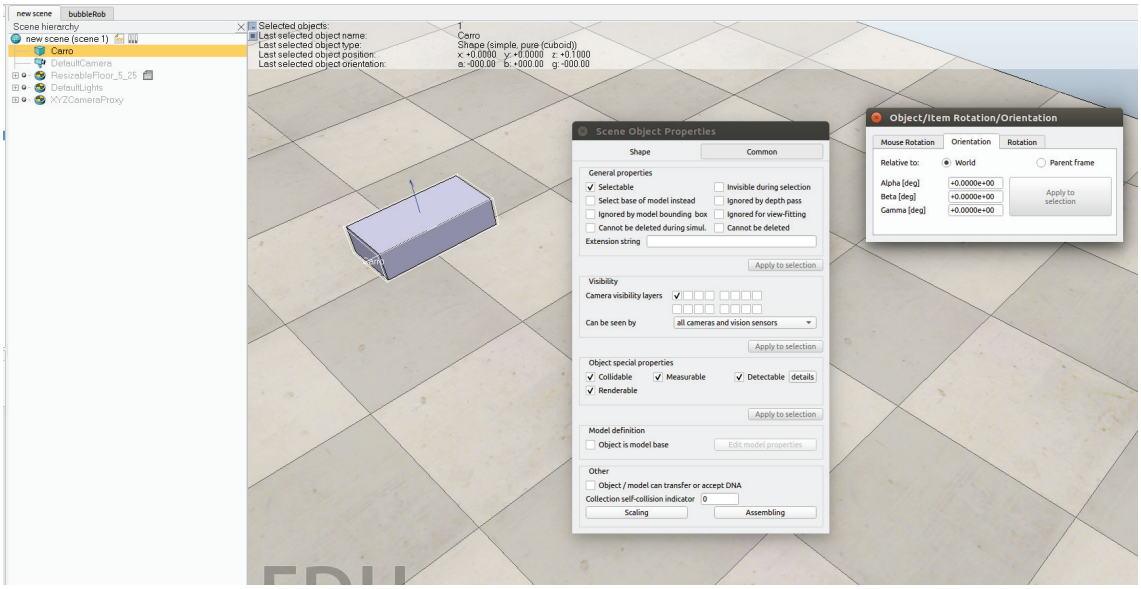


Figura 3.15. Resultado referente ao Passo 1.

- **Passo 2 - Criação da Roda Direita**

  1. Add → Primitive Shape → Cylinder
  2. Dimensão: (x = 0.125, z = 0.01)
  3. Posição: (x = -0.1, y = -0.105, z = 0.0625)
  4. Orientação: (Alpha = -90, Gamma = 0, Beta = 0)
  5. Renomeie Cylinder como Roda Direita
  6. Em Scene Object Properties → Common, Marcar as opções Collidable, Measurable, Detectable e Renderable

- **Passo 3 - Criação da Roda Esquerda**

  1. Add → Primitive Shape →Cylinder
  2. Dimensão: (x = 0.125, z = 0.01)
  3. Posição: (x = -0.1, y = 0.105, z = 0.0625)
  4. Orientação: (Alpha = -90, Gamma = 0, Beta = 0)
  5. Renomeie Cylinder como Roda Esquerda
  6. Em Scene Object Properties → Common, Marcar as opções Collidable, Measurable, Detectable e Renderable

- **Passo 4 - Criação da Roda Boba**

  1. Add → Primitive Shape → Sphere
  2. Dimensão: (x = 0.05)

3. Posição: (x = 0.15, y = 0, z = 0.025)
4. Orientação: (Alpha = 0, Gamma = 0, Beta = 0)
5. Renomeie Sphere como Roda Boba
6. Em Scene Object Properties → Common, Marcar as opções Collidable, Measurable, Detectable e Renderable

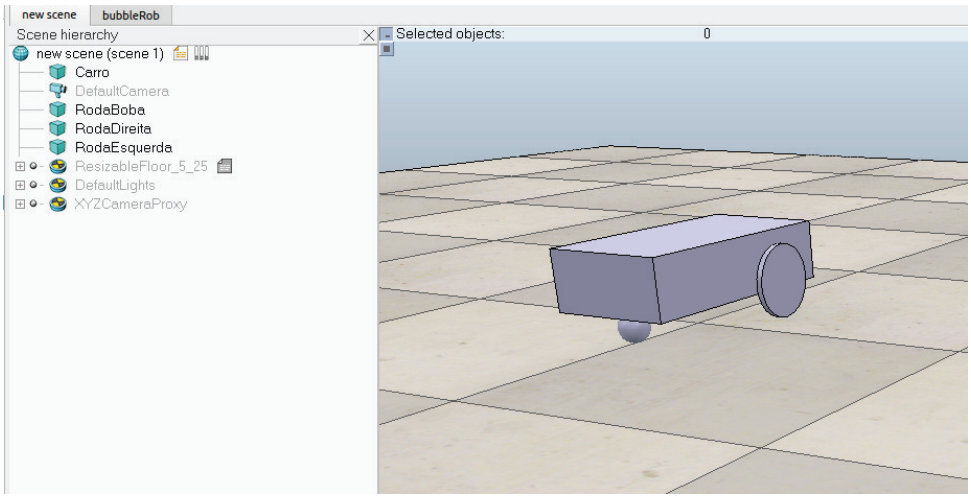


Figura 3.16. Resultado referente ao Passos 2 a 4.

- **Passo 5 - Adicionando o conector da roda boba**

1. Add → Force Sensor
2. Posição: (x = 0.15, y = 0, z = 0.05)
3. Renomeie para Conector

- **Passo 6 - Criação Motor Direito**

1. Add → Joint → Revolute
2. Posição: (x = -0.1, y = -0.105, z = 0.0625)
3. Orientação: (Alpha = -90, Gamma = 0, Beta = 0)
4. Renomeie Joint como MotorDireito

- **Passo 7 - Criação Motor Esquerdo**

1. Add → Joint → Revolute
2. Posição: (x = -0.1, y = 0.105, z = 0.0625)
3. Orientação: (Alpha = -90, Gamma = 0, Beta = 0)
4. Renomeie Joint como MotorEsquerdo

- **Passo 8 - Habilitar motores**

    1. Selecione os dois motores
    2. Em Scene Object Properties → *Show dynamic properties dialog*, selecione as opções *Motor Enabled* e *Lock motor when target velocity is zero*

- **Passo 9 - Hierarquizar os componentes**

    1. Ponha o Conector, MotorDireito e MotorEsquerdo como filhas do Carro
    2. Ponha RodaDireita como filha do MotorDireito
    3. Ponha RodaEsquerda como filha do MotorEsquerdo
    4. Ponha RodaBoba como filha do Conector

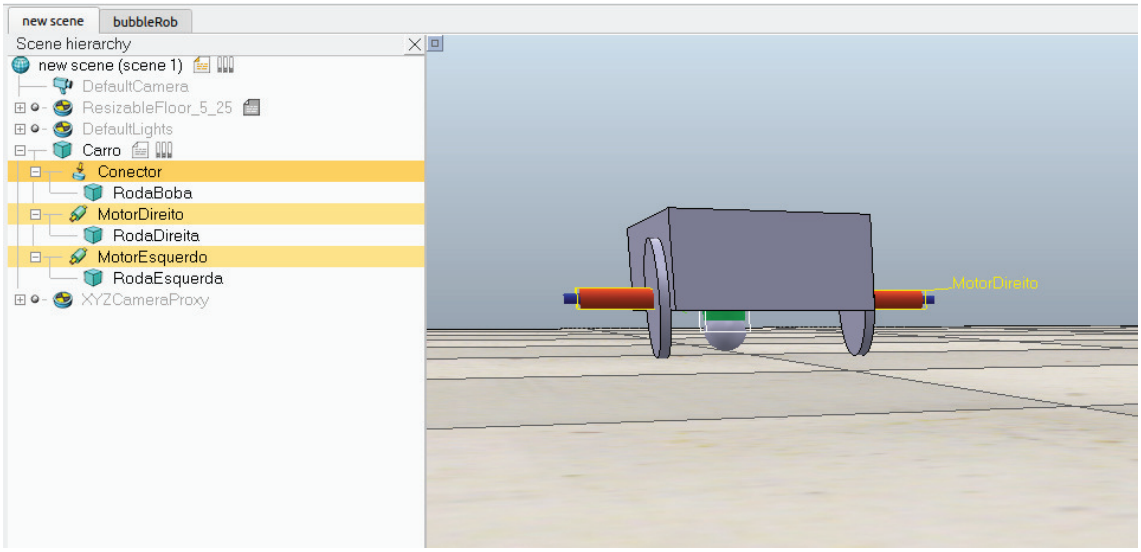


Figura 3.17. Resultado referente ao Passo 9.

- **Passo 10 - Escondendo as juntas**

    1. Selecione o Conector, o MotorEsquerdo e o MotorDireito.
    2. Scene Object Properties → Common
    3. Em *Camera visible layers*, desmarque todos as caixas
    4. Apply to selection

- **Passo 11 - Movimentando o carro**

    1. Com o mouse sobre o objeto Carro na hierarquia de cena, clique com o botão direito Add → Associated schild Script → Non threaded
    2. Adicione o código conforme explicado na Seção 3.3
    3. Execute a simulação

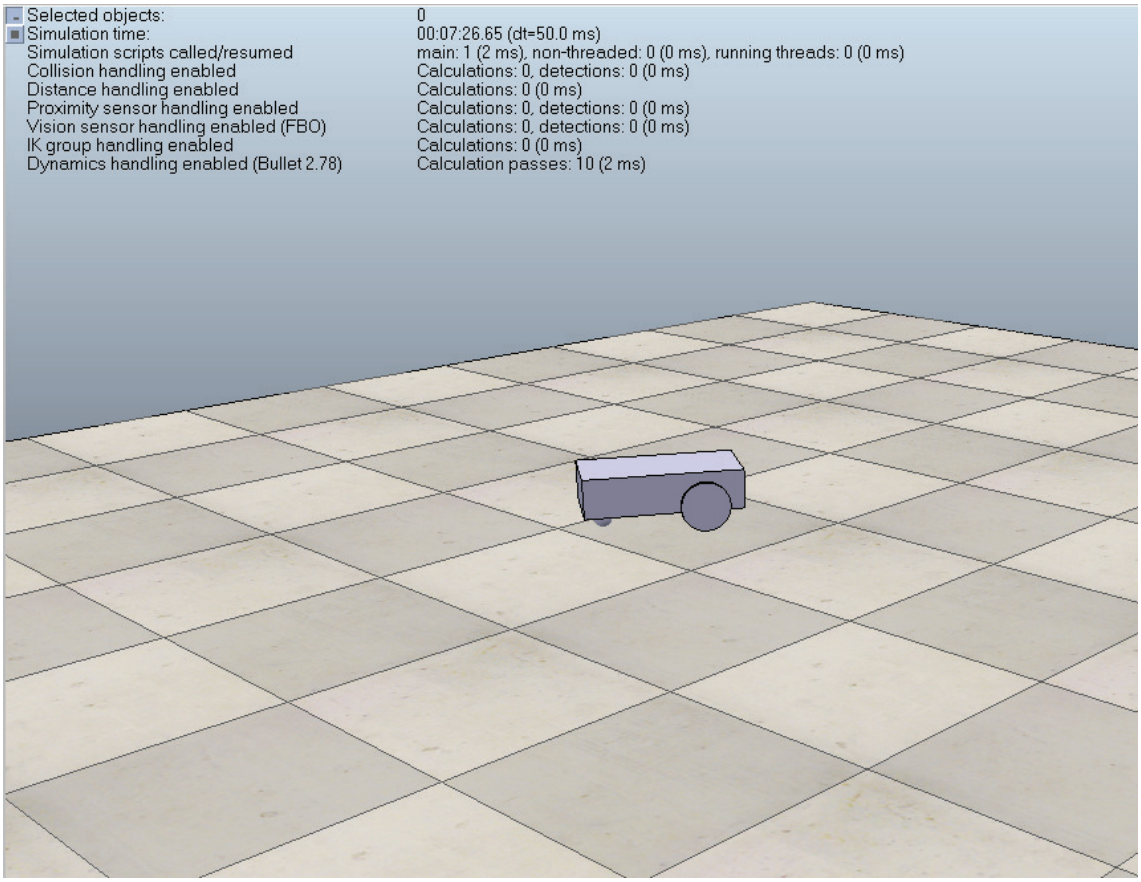


Figura 3.18. Robô montado executando.

3.6. Conclusão

Nesse capítulo foi apresentado um conteúdo introdutório sobre a plataforma de simulação de robôs V-REP. O software de simulação foi descrito, assim como suas principais ferramentas, demonstrando como realizar simulações mais simples e até mesmo criando seu ambiente próprio modelo de simulação.

A API remota apresentada é uma interessante característica dessa plataforma pois, mesmo sem conhecimento na linguagem Lua, o usuário poderá programar os robôs utilizando outras tecnologias que mais favorecer o seu conhecimento técnico.

Devido a facilidade do software, qualquer pessoa, mesmo sem um conhecimento aprofundado em robótica, poderá realizar simulações. Além disso, para alunos de robótica, profissionais e pesquisadores da área que necessitam realizar testes experimentais e não possuem uma bancada completa com robôs e sensores, a plataforma ajuda a realizar os experimentos simulados a fim de validar sua metodologia num estágio inicial.

Finalmente, V-REP é uma plataforma para simulação de robôs completa, podendo ser utilizada, inclusive, como ferramenta de base para disciplinas de Introdução à robótica ou Sistemas robóticos autônomos, por exemplo, por ser uma ferramenta ideal para colocar em prática os conceitos abordados sobre robótica.

3.7. Currículo dos autores

Luís Bruno Pereira do Nascimento possui Bacharelado em Ciência da Computação pela Universidade Estadual do Piauí (UESPI) e Mestrado em Engenharia Elétrica e de Computação pela Universidade Federal do Ceará (UFC). Atualmente é aluno de doutorado pelo Programa de Engenharia Elétrica e de Computação (PPgEEC) pela Universidade Federal do Rio Grande do Norte (UFRN). Suas áreas de interesse incluem robótica autônoma, otimização e aprendizado de máquina.

Diego da Silva Pereira possui graduação em Redes de Computadores pelo Instituto Federal do Rio Grande do Norte (IFRN), Mestrado em Ciência da Computação pela Universidade Estadual do Rio Grande do Norte (UERN) e está em doutoramento no Programa de Pós-graduação em Engenharia Elétrica e de Computação (PPgEEC) pela UFRN. Atualmente é professor no IFRN com pesquisas na área de redes de comunicação sem fio aplicadas à sistemas robóticos autônomos.

Vitor Gaboardi dos Santos possui graduação em Engenharia Elétrica na Universidade Federal do Rio Grande do Norte (UFRN) com período sanduíche na University of Kansas (USA). Atualmente é aluno de Mestrado no Programa de Pós-graduação em Engenharia Mecatrônica pela UFRN com pesquisas relacionadas a visão computacional aplicada à robótica assistiva.

Daniel Henrique Silva Fernandes possui Bacharelado em Engenharia Mecatrônica e Bacharelado em Ciências e Tecnologia pela Universidade Federal do Rio Grande do Norte (UFRN). Atualmente é aluno de Mestrado no Programa de Pós-graduação em Engenharia Mecatrônica pela UFRN com pesquisas relacionadas a robótica assistiva e visão computacional.

Pablo J. Alsina possui graduação em Engenharia Elétrica (1987), mestrado na área de controle de motores de indução (1991) e doutorado em Engenharia Elétrica com tema em controle de manipuladores robóticos (1996), pela Universidade Federal da Paraíba. Atualmente é professor titular do Departamento de Engenharia de Computação e Automação (DCA). É professor nos Programas de Pós-Graduação em Engenharia Mecatrônica (PPGEM) e em Engenharia Elétrica e de Computação (PPgEEC) da Universidade Federal do Rio Grande do Norte (UFRN), onde é chefe do Laboratório de Robótica. Desenvolve pesquisas em robótica, nas áreas de controle, planejamento e percepção robótica, com aplicações em robótica assistiva, robótica móvel e Veículos Aéreos Não Tripulados.

Referências

[Knoll et al. 2017] Knoll, J., Hevrdejs, K., Malinowski, A., and Miah, S. (2017). Virtual robot experiments for navigation in structured environments. In *Industrial Electronics (ISIE), 2017 IEEE 26th International Symposium on*, pages 1173–1178. IEEE.

[Lima et al. 2018] Lima, A. T., de Oliveira, I. F., Rodrigues, G. B., Domingues, J. D., Rocha, F. A. S., and Freitas, G. M. (2018). Utilização do robot operating system (ros) em conjunto com o simulador v-rep no ensino de robótica. In *Anais do XXII CBA, João Pessoa*.

[Obdržálek 2017] Obdržálek, Z. (2017). Mobile agents in multi-agent uav/ugv system. In *Military Technologies (ICMT), 2017 International Conference on*, pages 753–759. IEEE.

[Robotics 2019] Robotics, C. (2019). V-rep: Virtual robot experimentation platform.

[Rohmer et al. 2013] Rohmer, E., Singh, S. P., and Freese, M. (2013). V-rep: A versatile and scalable robot simulation framework. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1321–1326. IEEE.

[Tanberk and Tükel 2017] Tanberk, S. and Tükel, D. B. (2017). Kinect controlled chess playing robot. In *Smart Technologies, IEEE EUROCON 2017-17th International Conference on*, pages 594–598. IEEE.