Capacitação RAS OnBoarding / Atividade 1

Universidade Federal de Campina Grande - UFCG

Departamento de Engenharia Elétrica - DEE Capítulo Estudantil IEEE RAS UFCG

Equipe 4:

Coordenador: Fábio Marçal

Leila Medeiros de Farias

Mércia Regina da Silva

Rogério Moreira Almeida

13

de junho de

2024

# SUMÁRIO

1.Instalar o CoppeliaSim

2.Elaborar um relatório de atividades que explique como funciona dentro do Simulador:

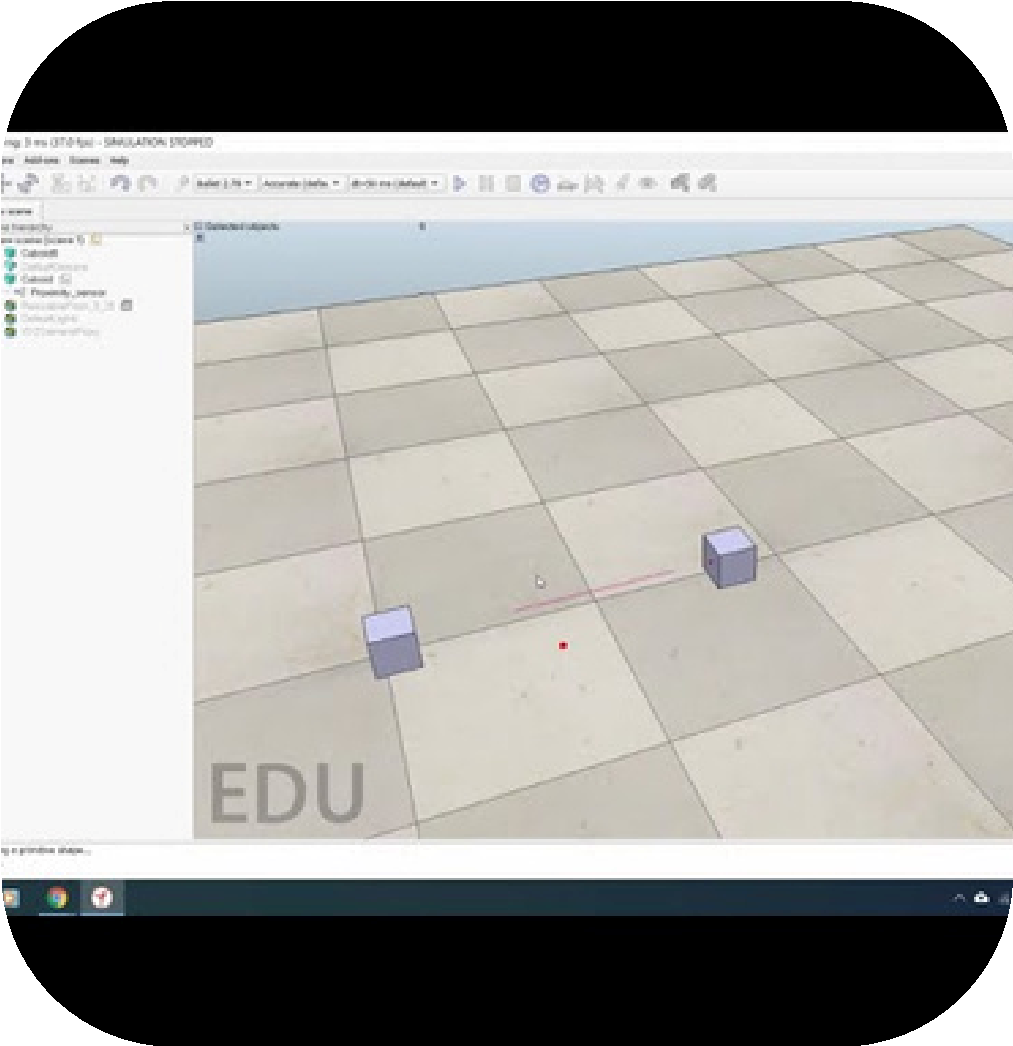
1. Criar uma nova Cena;
2. Como a simulação do Coppelia funciona;
3. Diferenças entre modo Síncrono e Assíncrono;
4. Relação Hierárquica dos objetos;
5. Configurações de propriedade dos objetos;
6. Acessar um modelo;
7. Configuração e utilização da API Remota em Python.

Capacitação Ras OnBoarding - Missão 01

A primeira atividade da Missão 01 se refere ao estudo do ambiente de simulação para a aplicação dos principais conceitos de robótica. Durante esta missão, temos como objetivo: criar um controlador GoToGoal, determinar visualmente ou por coordenadas, um ponto para que o robô se locomova.

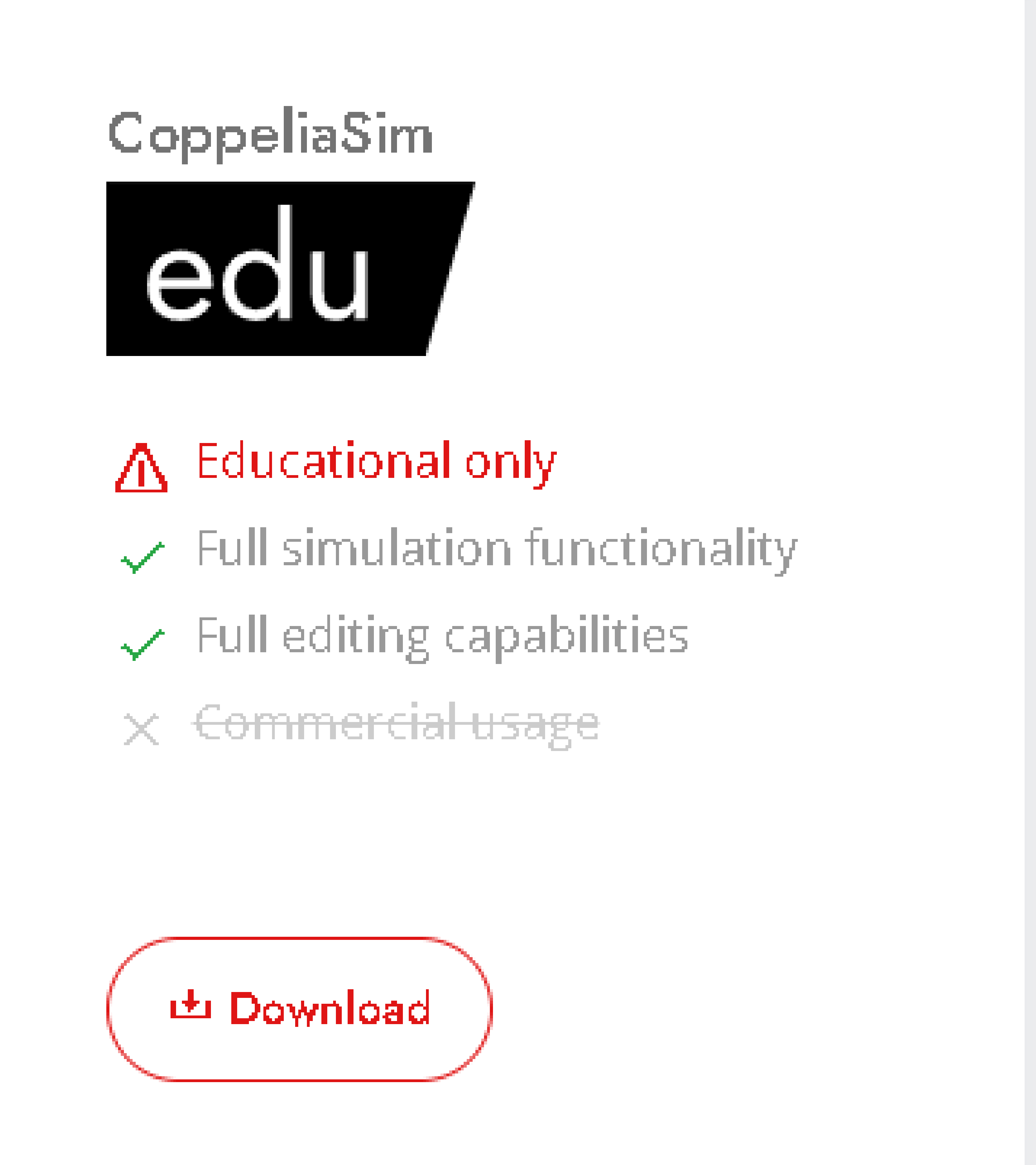
Descrição: Utilizar o software de simulação robótica CoppeliaSim para criar cenas e manipular objetos.

Atividade I: Conhecendo o CoppeliaSim



INSTALAR O COPPELIASIM

Inicialmente acessamos o site do CoppeliaSim por meio do link: https://www.coppeliarobotics.com/



Robot simulator CoppeliaSim: create, compose, simulate, any robot - Coppelia Robotics.

E realizamos o download da versão edu.

# CRIANDO UMA NOVA CENA

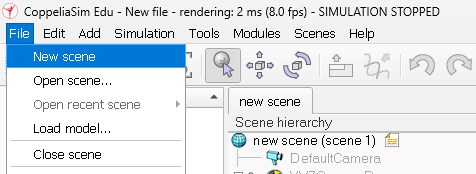
No canto superior esquerdo clicamos na opção:

File e em seguida na opção > New scene para criar uma nova cena.

As cenas podem ser abertas (carregadas) com:

[Arquivo > Abrir cena...] e salvas com [Arquivo > Salvar cena] ou [Arquivo > Salvar cena como]. Arquivos de cena (arquivos "\*.ttt").

Também suportam operações de arrastar e soltar entre a janela do explorer e a janela do aplicativo.



Uma cena ou conteúdo de imagem de cena pode ser visto por meio de um objeto visualizável associado a uma visualização, ele próprio contido em uma página. A cena padrão conterá os seguintes elementos:

 Vários objetos de câmera;

Vários objetos leves;

Várias visualizações;

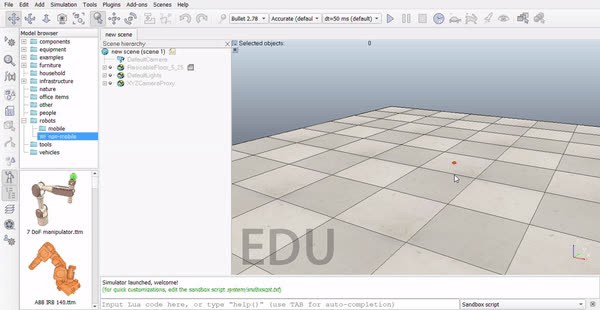
Várias páginas;

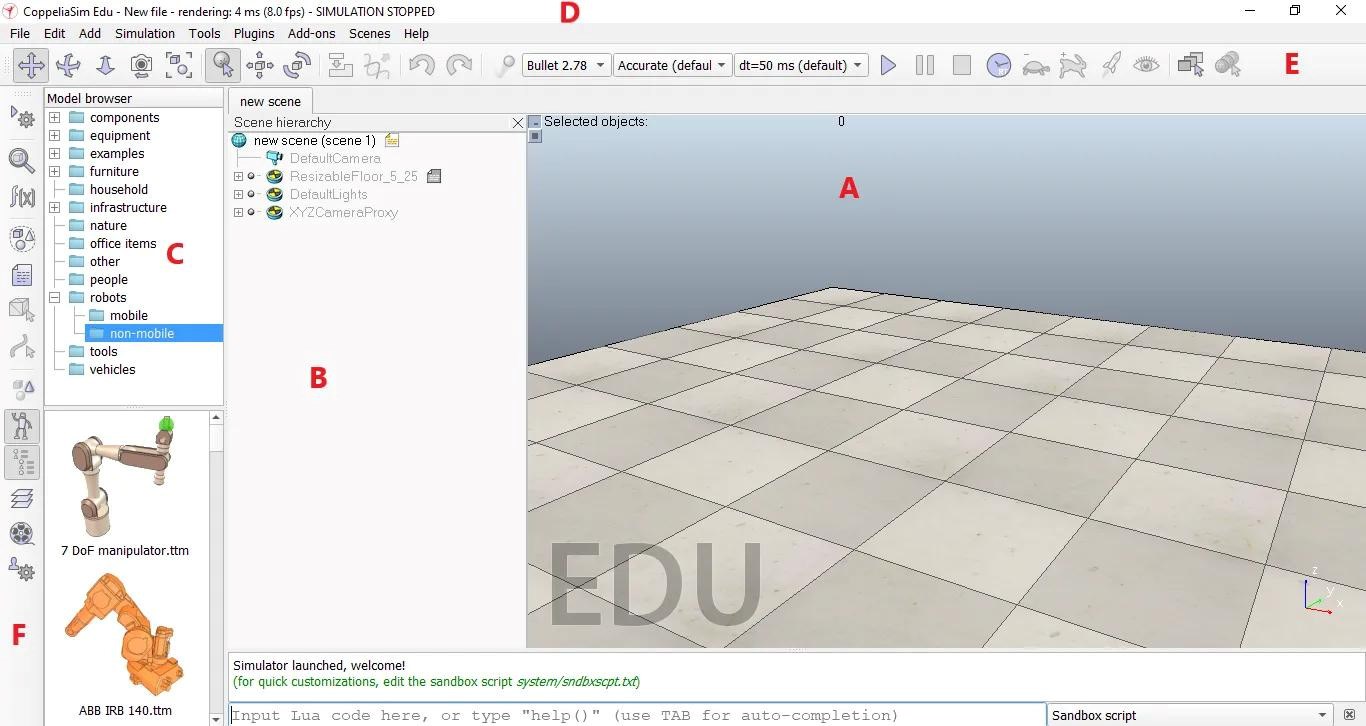
O ambiente: é composto por propriedades como luz ambiente, neblina, cor de fundo, etc;

O piso: é composto por objetos agrupados em um modelo;

O script principal padrão: o script principal padrão deve permitir a execução de simulações mínimas, sem a necessidade de scripts filhos.

ÁREA DE TRABALHO DO COPPELIASIM



LAYOUT PADRÃO

A é a área de Visualização da

Cena;

B é a Hierarquia da Cena;

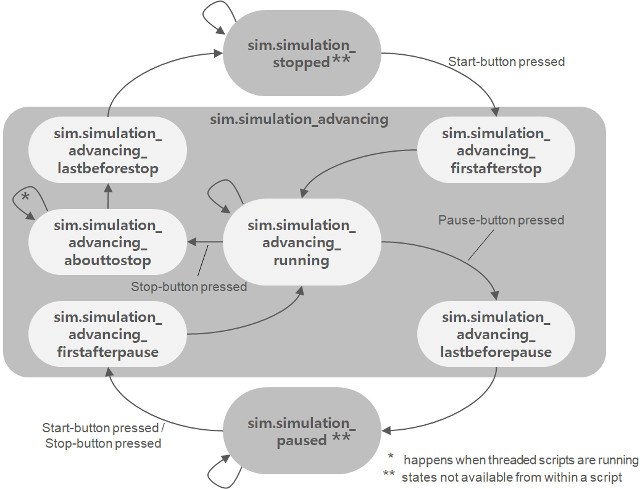
C é o Model Browser;

D mostra o status da cena;

E e F são as abas de acesso rápido;

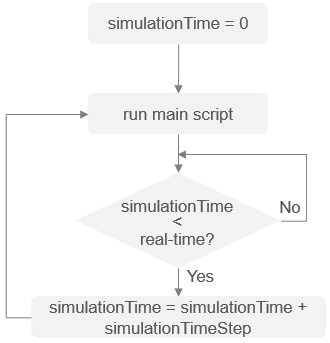
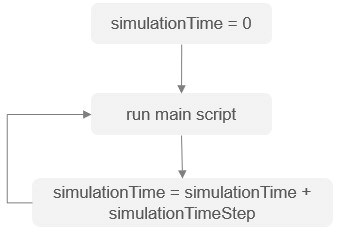
COMO A SIMULAÇÃO DO COPPELIA FUNCIONA

Uma simulação no CoppeliaSim pode ser iniciada, pausada e interrompida com [Simulação > Iniciar/Pausar/Parar simulação] ou através dos botões relacionados da barra de ferramentas:

o simulador utilizará estados intermediários adicionais para informar corretamente os scripts ou programas sobre o que acontecerá a seguir. O diagrama de estado a seguir ilustra os estados internos do simulador:

# CICLO DE SIMULAÇÃO/SIMULATION LOOP

O simulador opera avançando o A simulação em tempo real é suportada tempo de simulação em intervalos tentando manter o tempo de simulação de tempo constantes. sincronizado com o tempo real:

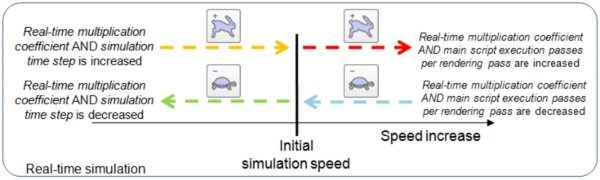
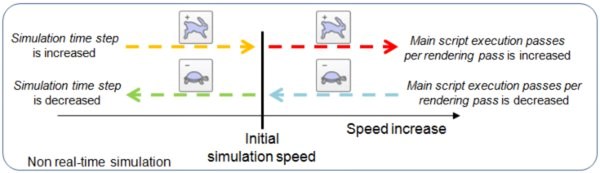


[LOOP DE SIMULAÇÃO PRINCIPAL] [LOOP DE SIMULAÇÃO EM TEMPO REAL]

# VELOCIDADE DE SIMULAÇÃO/SIMULATION SPEED

Em simulações não em tempo real, a velocidade da simulação depende principalmente de dois fatores: o intervalo de tempo da simulação e o número de passagens de simulação para uma passagem de renderização. No caso de uma simulação em tempo real, a velocidade da simulação depende principalmente do coeficiente de multiplicação em tempo real.

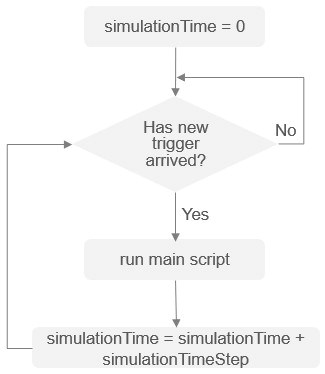




[Mecanismo de ajuste de velocidade de simulação para simulações não em tempo real] [Mecanismo de ajuste de velocidade de simulação para simulações em tempo real]

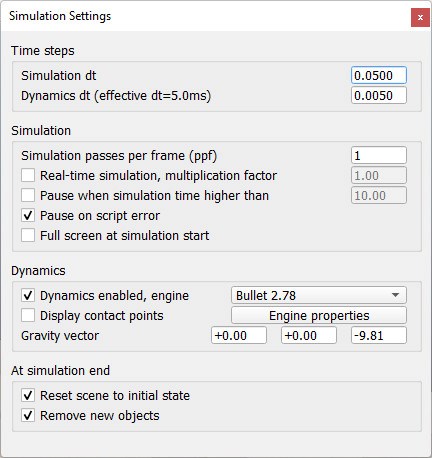
# MODO DE PASSO/STEPPING MODE

CoppeliaSim executa uma simulação executando uma etapa de simulação após a outra, automaticamente. Para sincronizar o CoppeliaSim com uma aplicação externa. Existe um mecanismo desse tipo, que é o modo stepping (ou modo síncrono):



## DIÁLOGO DE SIMULAÇÃO/SIMULATION DIALOG

Simulação dt;



### Dinâmica dt; (ppf);

Passagens de simulação por quadro

Simulação em tempo real, fator de multiplicação;

Pausa quando o tempo de simulação for superior a;

Pausa em erro de script;

Tela cheia no início da simulação;

### Dinâmica ativada;

Exibir pontos de contato;

Propriedades do motor;

### Vetor de gravidade; inicial;

Redefinir a cena para o estado

Remover novos objetos.

## DIÁLOGO DE SIMULAÇÃO/SIMULATION DIALOG

Uma simulação de etapas pode ser facilmente implementada por meio da função de retorno de chamada sysCall\_beforeMainScript. A partir de um aplicativo externo, geralmente há uma função dedicada para ativar o modo de etapas e, em seguida, acionar cada etapa individual. A seguir ilustra o modo de etapa de um cliente API remoto Python ZeroMQ:

import time

from coppeliasim\_zmqremoteapi\_client import RemoteAPIClient

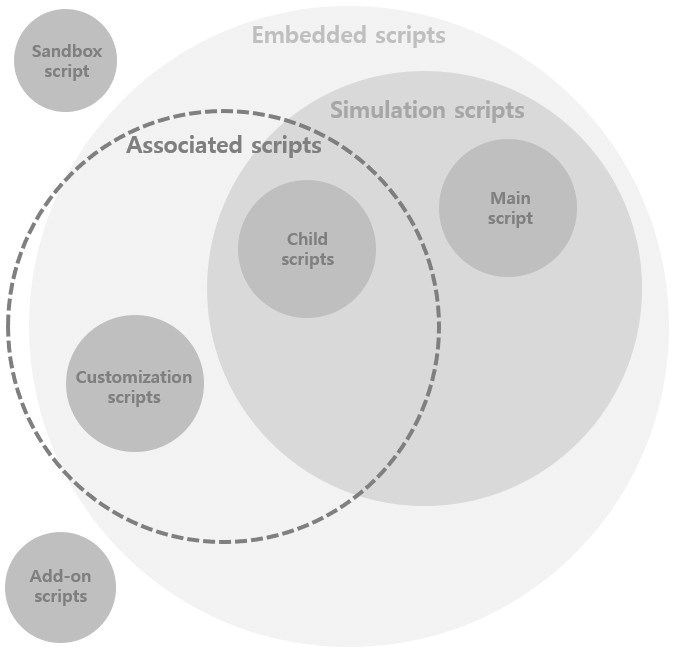
client = RemoteAPIClient() sim = client.getObject('sim') sim.setStepping(True)

sim.startSimulation() while (t := sim.getSimulationTime()) < 3:

s = f'Simulation time: {t:.2f} [s]' print(s) sim.step() sim.stopSimulation()

# ROTEIROS/SCRIPTS

CoppeliaSim é um simulador altamente personalizável: quase todas as etapas de uma simulação são definidas pelo usuário.

 Por outro lado, o script sandbox e os complementos não estão associados a nenhuma cena, modelo ou objeto específico e serão executados em todas as cenas de maneira semelhante.

## DIFERENÇAS ENTRE MODO SÍNCRONO E ASSÍNCRONO

Os modos síncrono e assíncrono são conceitos fundamentais em computação e comunicação, com diferenças significativas em como as operações são executadas e gerenciadas.

Quando usar cada modo:

1.Modo Síncrono:

Simulações científicas e de pesquisa onde a precisão temporal é crítica;

Desenvolvimento de algoritmos que dependem de passos exatos e sincronizados; Quando a ordem de operações deve ser garantida para evitar inconsistências.

2. Modo Assíncrono:

Simulações em tempo real que exigem alta performance;

Aplicações onde a flexibilidade e a velocidade são mais importantes que a precisão exata;

Cenários onde múltiplos processos ou agentes devem operar simultaneamente sem a necessidade de sincronização estrita.

|  |  |  |
| --- | --- | --- |
| Característica | Modo Síncrono | Modo Assíncrono |
| Execução | Bloqueante: uma tarefa deve ser concluída antes que outra possa começar. | Não Bloqueante: uma tarefa pode ser iniciada sem esperar a conclusão de outra. |
| Comunicação | Imediata: como chamadas telefônicas,  ambas as partes estão conectadas ao  mesmo tempo e interagem em tempo  real. | Não Imediata: as mensagens podem ser enviadas e recebidas em tempos diferentes, sem a necessidade de ambas as partes estarem conectadas simultaneamente. |
| Previsibilidade | Geralmente mais previsível, pois as  tarefas seguem uma ordem definida e  esperam uma resposta antes de  continuar. | Pode melhorar a eficiência e a responsividade do sistema, pois permite a execução de múltiplas tarefas simultaneamente, porém possui menor previsibilidade. |
| Simplicidade na Implementação | Pode ser mais simples, pois o fluxo de controle é linear e direto. | Pode ser mais complexo devido à necessidade de gerenciar  mecanismos para lidar com respostas que chegam em momentos  imprevisíveis. |
| Aplicações Comuns | Sistemas onde a ordem estrita de operações é crítica, como transações financeiras em um banco. Jogos multiplayer em tempo real onde a sincronia é essencial. | Aplicações web onde a responsividade do usuário é importante,  permitindo carregamento de dados em segundo plano. Sistemas  de notificação e mensageria, onde as respostas podem ser  tratadas conforme chegam sem bloquear outras operações. |

Principais diferença entre os modos Síncrono e Assíncrono

Autor: Rogerio

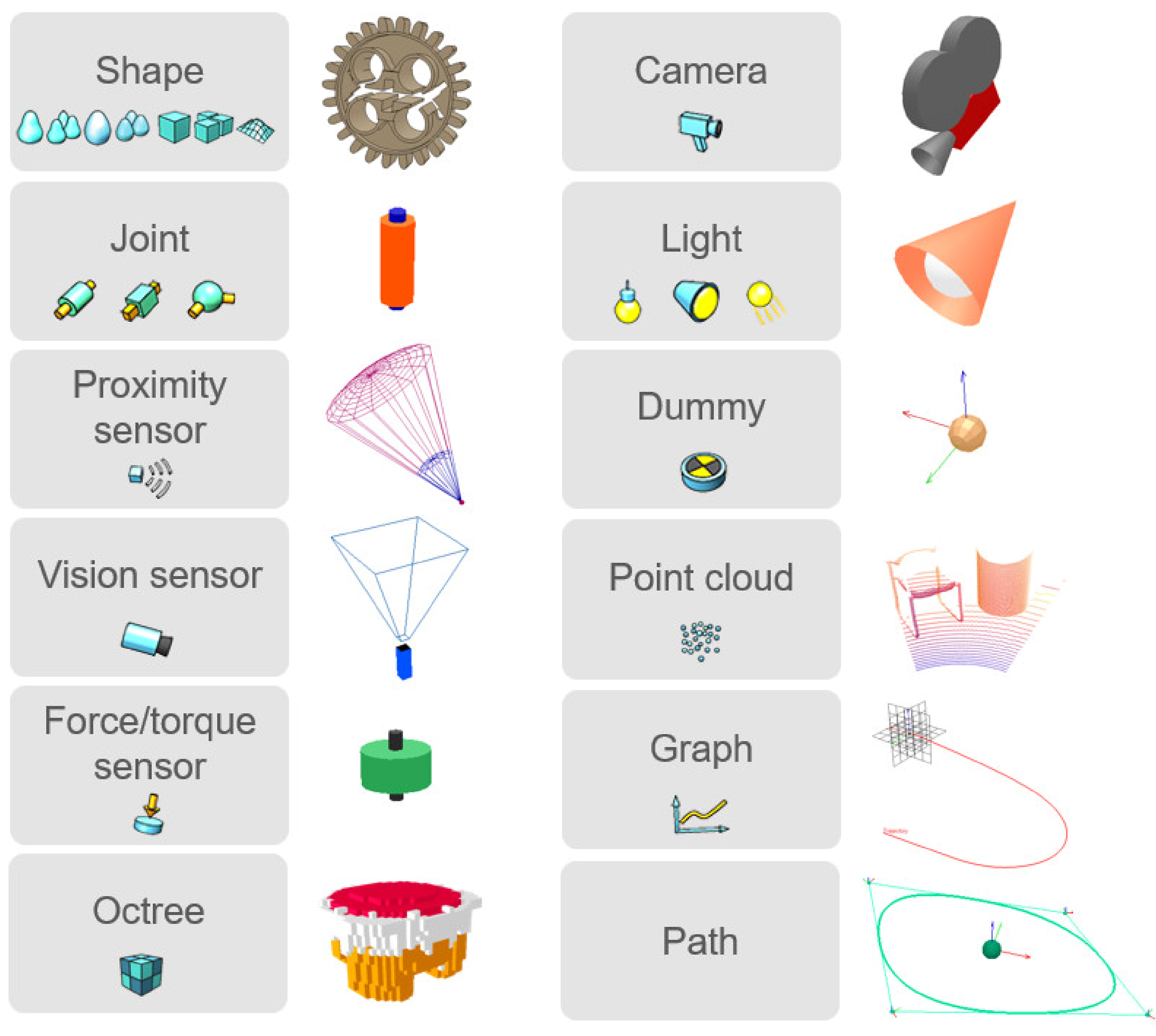
## RELAÇÃO HIERÁRQUICA DOS OBJETOS

A hierarquia de cenas é visível por padrão, mas pode ser alternada com o botão correspondente da barra de ferramentas. Ele exibe o conteúdo de uma cena (ou seja, todos os objetos de cena que compõem uma cena).

## RELAÇÃO HIERÁRQUICA DOS OBJETOS

1

. Estrutura Hierárquica

[Tipos de objetos no CoppeliaSim e sua representação tridimensional]

Root Object: ponto de partida da hierarquia e pode ser considerado o "pai" de todos os outros objetos na cena. Geralmente, é a própria cena ou um objeto que serve como referência central;

Parent Object (Objeto Pai): Qualquer objeto que tenha outros objetos "filhos" conectados a ele;

Child Object (Objeto Filho): Um objeto que está ligado a um objeto "pai" e que herda transformações e propriedades do seu pai.

Formas: uma forma é uma malha rígida composta por faces triangulares;

Articulações: um objeto conjunto é uma junta ou atuador;

Gráficos: um gráfico é usado para registrar e visualizar dados de simulação;

Manequins: um manequim é um ponto com orientação;

Dummies são objetos multifuncionais que podem ter muitas aplicações diferentes;

Sensores de proximidade: um sensor de proximidade detecta objetos de forma geometricamente exata dentro do seu volume de detecção;

Sensores de visão: um sensor de visão é um sensor do tipo câmera, que reage à luz, cores e imagens;

Sensores de força: um sensor de força é um objeto capaz de medir forças e torques que lhe são aplicados;

Câmeras: uma câmera é um objeto que permite ver a cena da simulação de vários pontos de vista;

Luzes: uma luz é um objeto que permite iluminar a cena da simulação;

Árvores OC: uma árvore OC é uma estrutura de dados de particionamento espacial composta por voxels;

Nuvens de pontos: uma nuvem de pontos é uma estrutura de árvore OC que contém pontos;

Caminhos: um caminho é uma sucessão de pontos com orientação no espaço;

As path é um pseudo objeto de cena, pois é construído usando apenas manequins e um script de customização que descreve sua funcionalidade e comportamento.

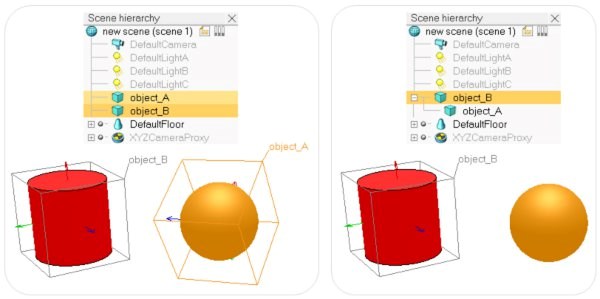
Os objetos podem ser:

Colidável: objetos colidíveis podem ser testados para colisão contra outros objetos colidíveis;

Mensuráveis: objetos mensuráveis podem ter a distância mínima entre eles e outros objetos mensuráveis calculada;

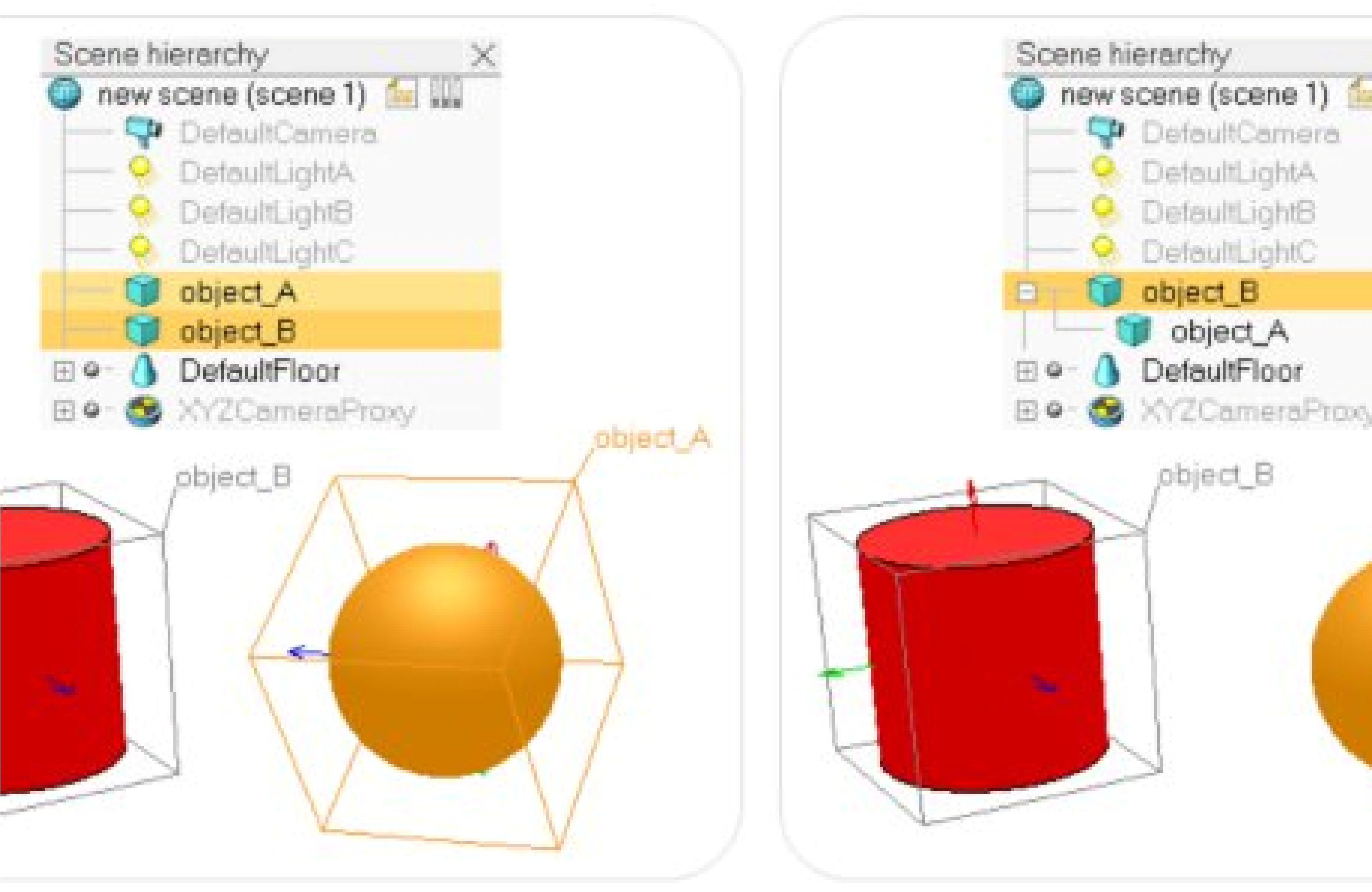
Detectável: objetos detectáveis podem ser detectados por sensores de proximidade;

Visível: objetos visíveis podem ser examinados, observados ou seu conteúdo de imagem pode ser visualizado em visualizações.

Se o objeto A for construído sobre o objeto B, então o objeto B será o pai e o objeto A será o filho. Para criar relacionamentos pai-filho entre o objeto B e o objeto A, selecione o objeto A e, em seguida, selecione o objeto B (a ordem de seleção é importante).

Em seguida, selecione [Editar > Definir pai, manter pose(s)].

## RELAÇÃO HIERÁRQUICA DOS OBJETOS

1. Estrutura Hierárquica

Root Object: ponto de partida da hierarquia e pode ser considerado o "pai" de todos os outros objetos na cena. Geralmente, é a própria cena ou um objeto que serve como referência central;

Parent Object (Objeto Pai): Qualquer objeto que tenha outros objetos "filhos" conectados a ele;

Child Object (Objeto Filho): Um objeto que está ligado a um objeto "pai" e que herda transformações e propriedades do seu pai.

## TRANSLAÇÃO E ROTAÇÃO DE OBJETOS

Na imagem, os botões 7 e 8 podem ser usados para translação e rotação do objeto nas direções X, Y e Z respectivamente. Quando você seleciona esses botões, uma caixa de diálogo aparece com 3 a 4 guias, então o que são?

Essas são as opções que CoppeliaSim nos oferece para brincar com os objetos.

## TRANSLAÇÃO E ROTAÇÃO DE OBJETOS

### Translação

Translação do mouse: podemos mover o objeto com o mouse com base no eixo selecionado;

Posição: permite colocar o objeto no local exato inserindo as coordenadas;

Translação: podemos mover o objeto em etapas, ou seja, podemos definir o tamanho do passo de acordo com o quanto queremos que o objeto seja deslocado iterativamente;

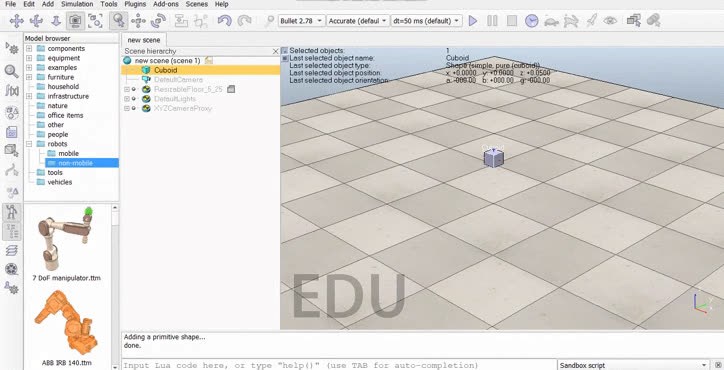
Pos Scaling: permite dimensionar a posição do objeto.

Rotação

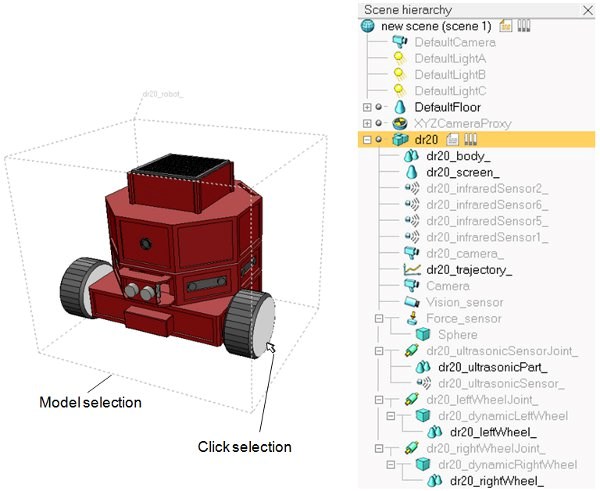
Rotação do mouse: Ao usar isto, o objeto pode ser girado usando o mouse com base no eixo selecionado;

Orientação: Ao utilizar este objeto pode ser colocado na posição de orientação exata inserindo os ângulos com o respectivo eixo;

Rotação: permite modificar a orientação em etapas com base no eixo selecionado.

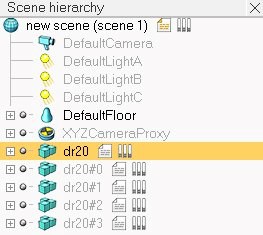
 TRANSLAÇÃO E ROTAÇÃO DE OBJETOS

## ACESSAR UM MODELO

 Para acessar um modelo dentro os diversos oferecidos pelo CoppeliaSim, clicamos em uma das opções da seção Model browser do lado esquerdo da tela, onde temos modelos humanoides, braço robóticos, entre outros.

Seleciona o modelo desejado e arrasta-lo com o curso do mouse para a nossa cena.

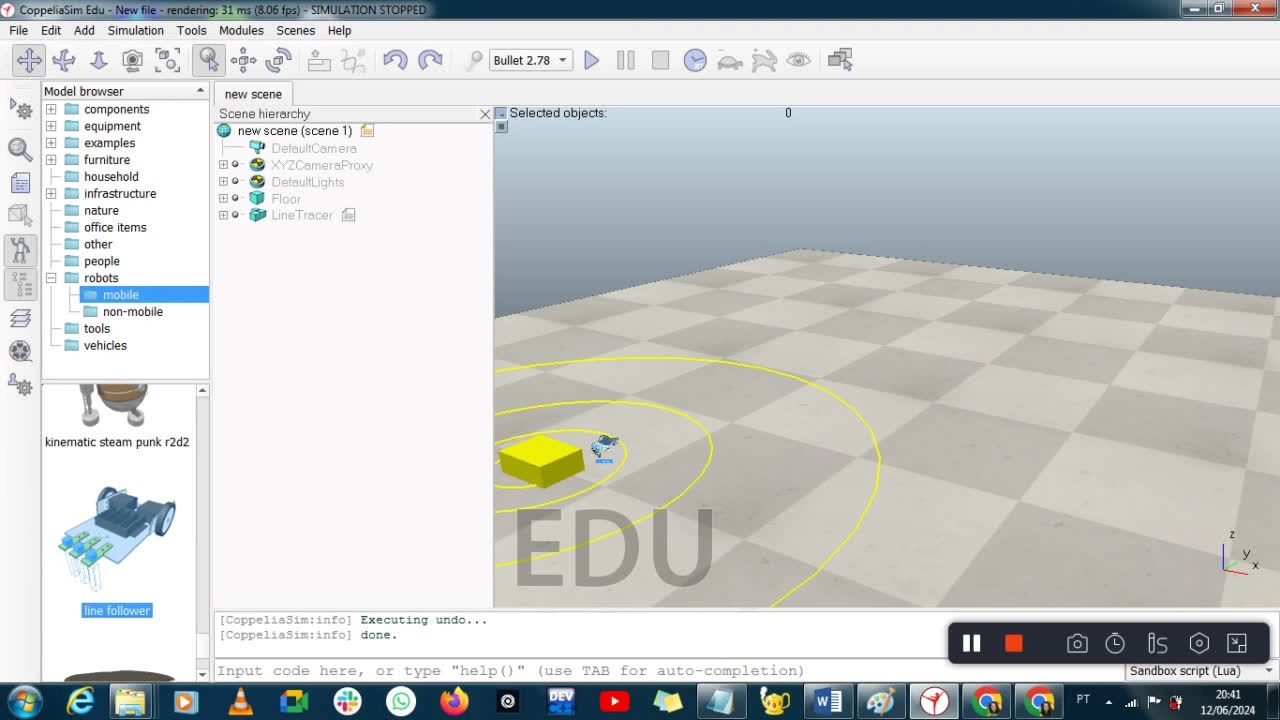
## ACESSAR UM MODELO

 Um clique duplo em uma tag de modelo abre a caixa de diálogo do modelo, onde as propriedades do modelo podem ser ajustadas.

Também é uma boa prática recolher a hierarquia de um modelo depois que o modelo foi editado para identificar facilmente o número de elementos/modelos agrupados logicamente:

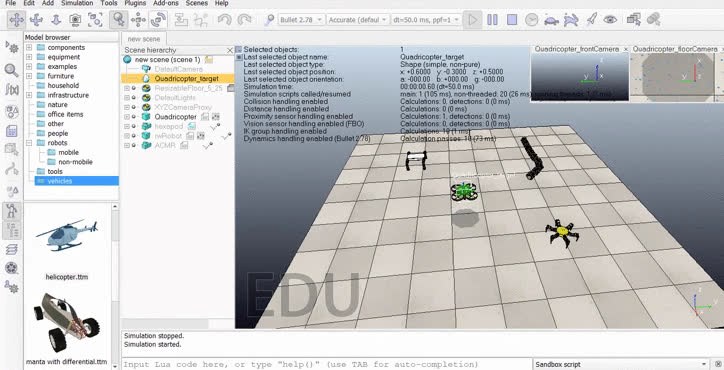
[Visualização da hierarquia de cenas exibindo 5 modelos recolhidos]

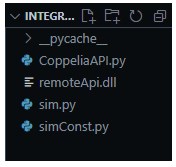
## ACESSANDO UM MODELO



[Acessando o modelo line follower] [modelo line follower anexado na cena]

## ACESSANDO UM MODELO



 Passo 1: Organização de diretório

Localize os arquivos “sim.py”, “simConst.py” e “remoteApi.ddl” localizados na pasta de instalação do coppelia, copie-os e cole-os em outra pasta.

Nesta mesma pasta, crie seu arquivo .py onde vamos escrever o código de conexão e

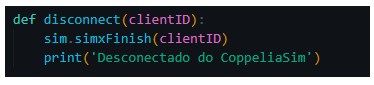
utilização da API

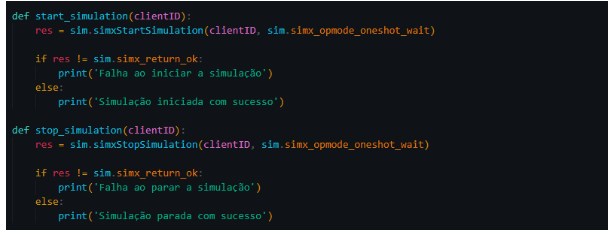
Passo 2: Métodos de conexão e controle de simulação

Abra o editor de texto da sua escolha, aqui estou usando o VsCode, importe os métodos do arquivo sim e a biblioteca time (será usada para fazer algumas manipulações em relação ao tempo).

Faça um método chamado connect(), para se conectar com o coppelia e obter o clientID referente ao seu computador e seu coppelia



 Também o método disconnect(), será utilizado quando quisermos encerrar a conexão:

 Após isso, para iniciar e interromper uma simulação, criaremos mais dois métodos:

start\_simulation() stop\_simulation() Passo 3: Métodos de controle do robô: próximo método será para pegar o handle do robô, um identificador único que usaremos para controlar o robô:

A partir daí, acessando a documentação temos acesso a vários métodos para controle do nosso robô, tais como:

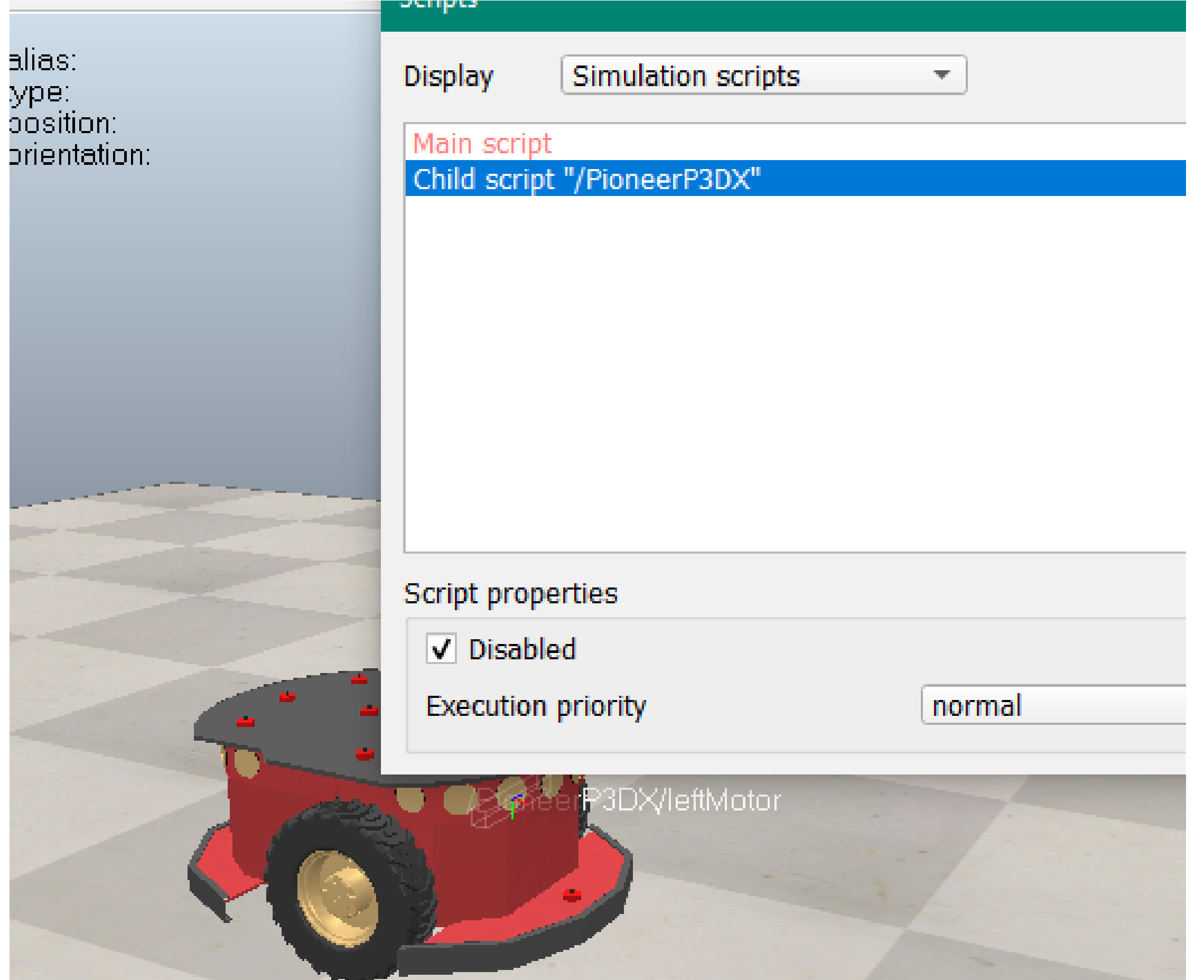
simxSetJointTargetPosition() e simxSetJointTargetVelocity() para controlar as posições e velocidades das articulações dos robôs.

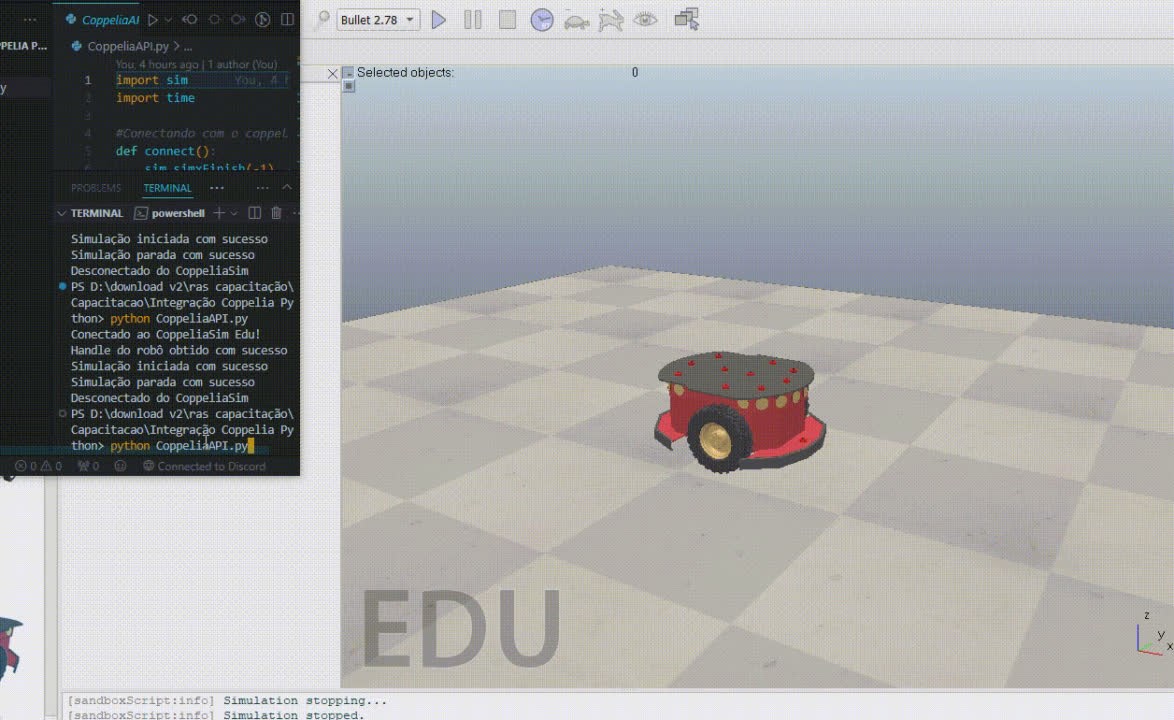
simxGetJointPosition() e simxGetForceSensorValue() para obter leituras de sensores, como posições articulares e valores de força.

Com isso, faremos um simples código para fazer o robô pioneer p3dx girar, dessa forma criamos o método move\_robot()

 E por fim, configuramos o nosso main(), utilizando os métodos que criamos da seguinte forma:

OBS: Para o seu código funcionar da maneira correta, você deve desabilitar o default script que já vem no robô

 Ao executar, podemos observar o resultado no próprio coppelia, se tudo estiver certo, a simulação será iniciada e o robô fará o que seu programa diz



## REPOSITÓRIO GITHUB - CAPACITAÇÃO



<https://github.com/rogeriomoral/Capacitacao>

## CONCLUSÃO

A aplicação de intervenções pedagógicas no campo da Robótica Educativa, através da utilização de simulações. A solução aqui proposta foi a utilização do software CoppeliaSim, escolhido por possuir uma versão educacional gratuita e por sua capacidade de simular realisticamente os sistemas robóticos.

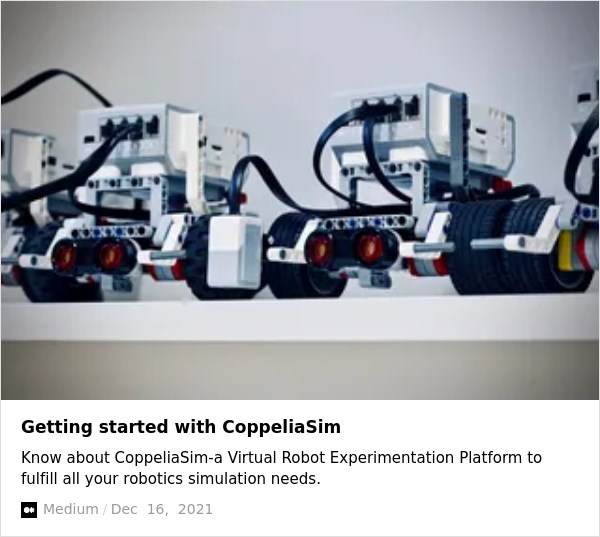
São apresentados conceitos teóricos para que os usuário crie seu próprio projeto e sua metodologia para a aplicação de intervenções com o uso dos conceitos de robótica.

Foi apresentado através de um tutorial de instalação, configuração e utilização de API em Python com objeitovo de criar um controlador GoToGoal, determinar visualmente ou por coordenadas, um ponto para que o robô se locomova.

Foi observado que a aplicação dos conceitos de robótica em diferentes contextos é capaz de aumentar a assimilação e a compreensão dos conteúdos pelos usuários, através de um ambiente de colaboração, construção e investigação

## REFERÊNCIAS

 [https://manual.coppeliarobotics.com](https://manual.coppeliarobotics.com/)



OBRIGADO(A)!

Universidade Federal de Campina Grande - UFCG

Departamento de Engenharia Elétrica - DEE

Capítulo Estudantil IEEE RAS UFCG 13 de junho de 2024 emails

leila.medeiros.farias@ccc.ufcg.edu.br

mercia.silva@ee.ufcg.edu.br

rogerio.almeida@ee.ufcg.edu.br