Tarefas: 1. Instalar o CoppeliaSim

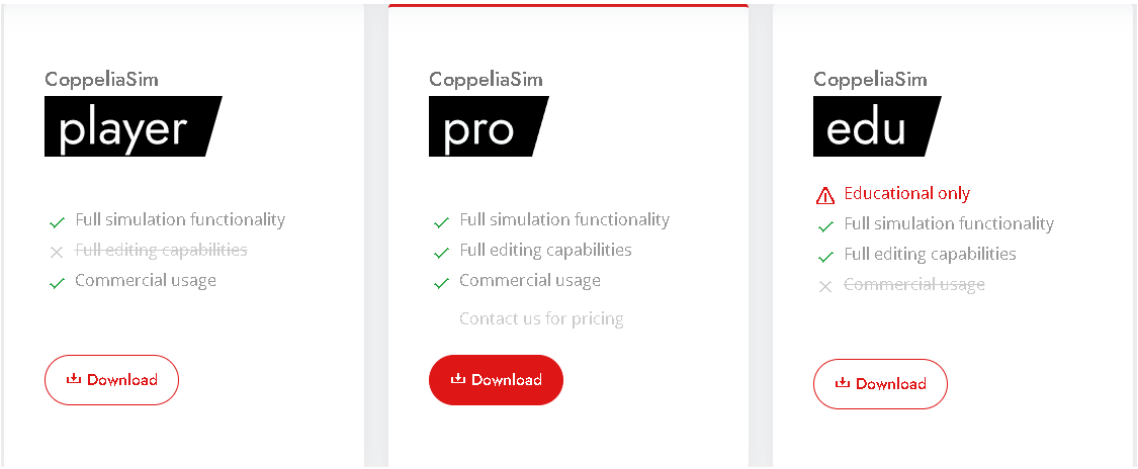
2. Elaborar um relatório de atividades que explique como funciona dentro do Simulador:

a) Criar uma nova Cena

Inicialmente acessamos o site do CoppeliaSim por meio do link: [Robot simulator CoppeliaSim: create, compose, simulate, any robot - Coppelia Robotics.]

E realizamos o download da versão edu.

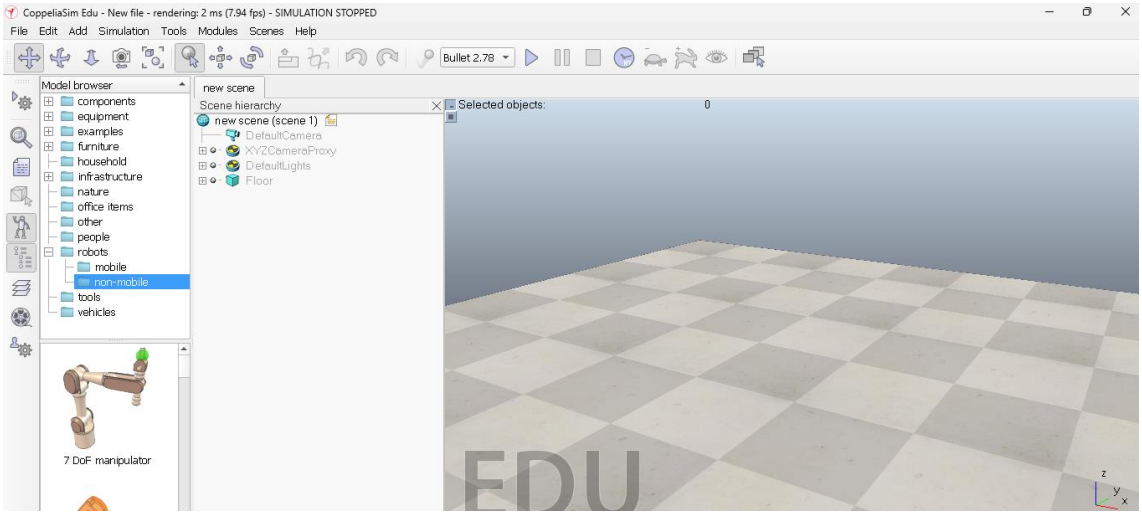
Figura1: Área de download, site CoppeliaSim



Autoria: Site CoppeliaSim

Posteriormente após a instalação inicializamos o CoppeliaSim.

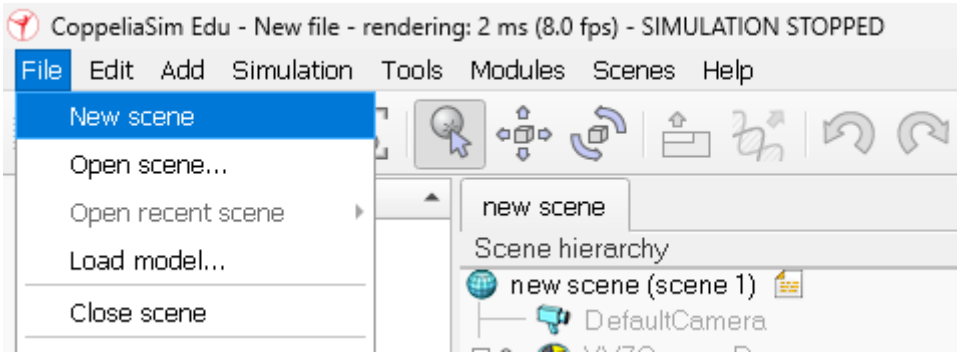
Figura2: Pagina, do CoppeliaSim



Autoria: Própria.

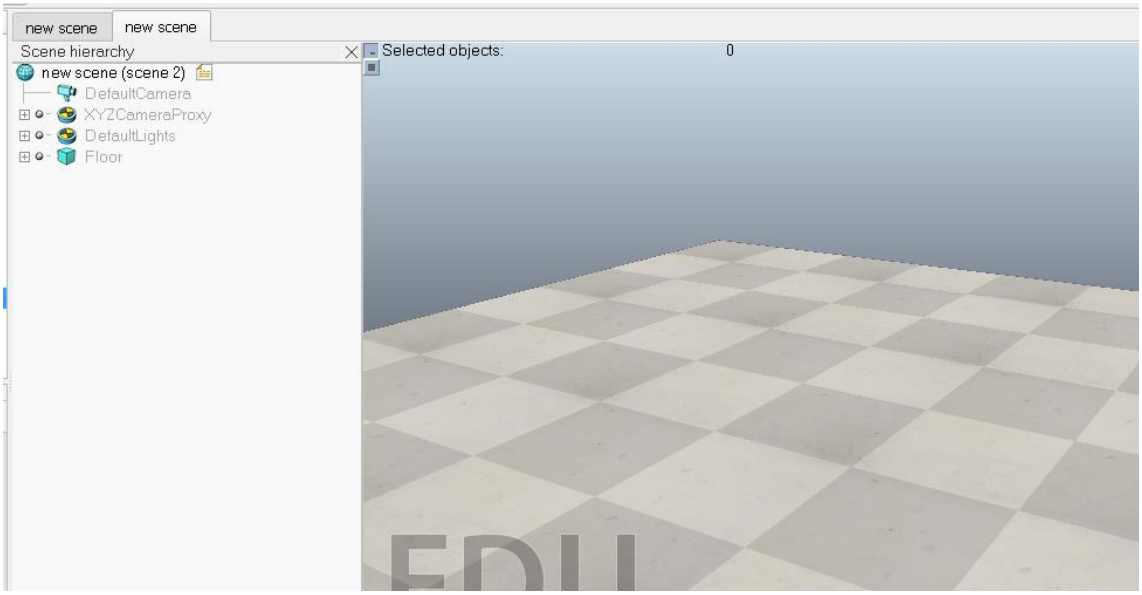
No canto superior esquerdo clicamos na opção File e em seguida na opção New scene para criar uma nova cena

Figura 3: Criando uma nova cena.



Autoria: Própria.

Figura 4: Nova cena criada.



Autoria: Própria.

b) Como a simulação do Coppelia funciona

O CoppeliaSim, é um software de simulação de robótica e automação que permite a criação e o controle de sistemas complexos de robótica. Ele é amplamente utilizado em pesquisa, educação e desenvolvimento de robótica.

Podemos dividir o CoppeliaSim em 5 áreas para melhor descrever suas funcionalidades

1. Modelagem de Ambientes e Robôs

Objeto 3D: O CoppeliaSim permite a importação e criação de objetos 3D, que podem ser componente de robôs, sensores, moveis, atuadores...

Construção Modular: Utilizando uma abordagem modular, os usuários podem construir robôs a partir de componentes básicos, como rodas, braços, sensores, e ligá-los entre si para formar um sistema completo.

2. Motores de Física

Simulação Física: O software utiliza motores de física como Bullet, ODE (*Open Dynamics Engine*), Vortex e Newton para simular interações físicas realistas, incluindo colisões, gravidade, fricção e dinâmica dos corpos rígidos e flexíveis.

Parâmetros de Física: Os usuários podem ajustar os parâmetros dos motores de física para obter o comportamento desejado das simulações.

3. Scripting e Automação

LUA *Scripting*: O CoppeliaSim usa a linguagem de script Lua para controlar a simulação. Scripts podem ser escritos para definir comportamentos de robôs, responder a eventos, controlar movimentos, processar dados de sensores e muito mais.

APIs Externas: Além do *scripting* interno, o CoppeliaSim oferece APIs para integração com linguagens de programação externas como Python, C/C++, Java, MATLAB, entre outras, permitindo controle remoto da simulação.

4. Controle e Sensoriamento

Sensores Virtuais: O *software* inclui uma variedade de sensores virtuais, como câmeras, sensores de proximidade, giroscópios, acelerômetros, entre outros, que podem ser utilizados para obter dados do ambiente simulado.

Controle de Atuadores: Atuadores como motores, juntas e servomecanismos podem ser controlados através de *scripts* ou APIs, permitindo o desenvolvimento de sistemas de controle sofisticados.

5. Visualização e Interação

Visualização em Tempo Real: O CoppeliaSim oferece visualização em tempo real da simulação, permitindo que os usuários observem e interajam com o ambiente simulado enquanto a simulação está em execução.

Modo de Gravação: Os usuários podem gravar as simulações para análise posterior, facilitando a revisão e o diagnóstico de comportamentos de robôs.

6. Colaboração e Extensibilidade

Extensibilidade: O CoppeliaSim é altamente extensível, permitindo a adição de novos *plugins*, *scripts* e bibliotecas que podem expandir suas capacidades.

Colaboração: Suporta simulações distribuídas e colaboração entre vários usuários, facilitando projetos em equipe e pesquisa colaborativa.

c) 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.

Tabela1: Principais diferença entre os modos Síncrono e Assíncrono

	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, porem possui uma 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.

Autoria: própria.

d) Relação Hierárquica dos objetos

No CoppeliaSim, a hierarquia dos objetos é um conceito fundamental para a organização e manipulação de cenas. A hierarquia define como os objetos estão relacionados uns aos outros, como pai e filho, e isso afeta o comportamento dos objetos em termos de movimento, transformações e dependências.

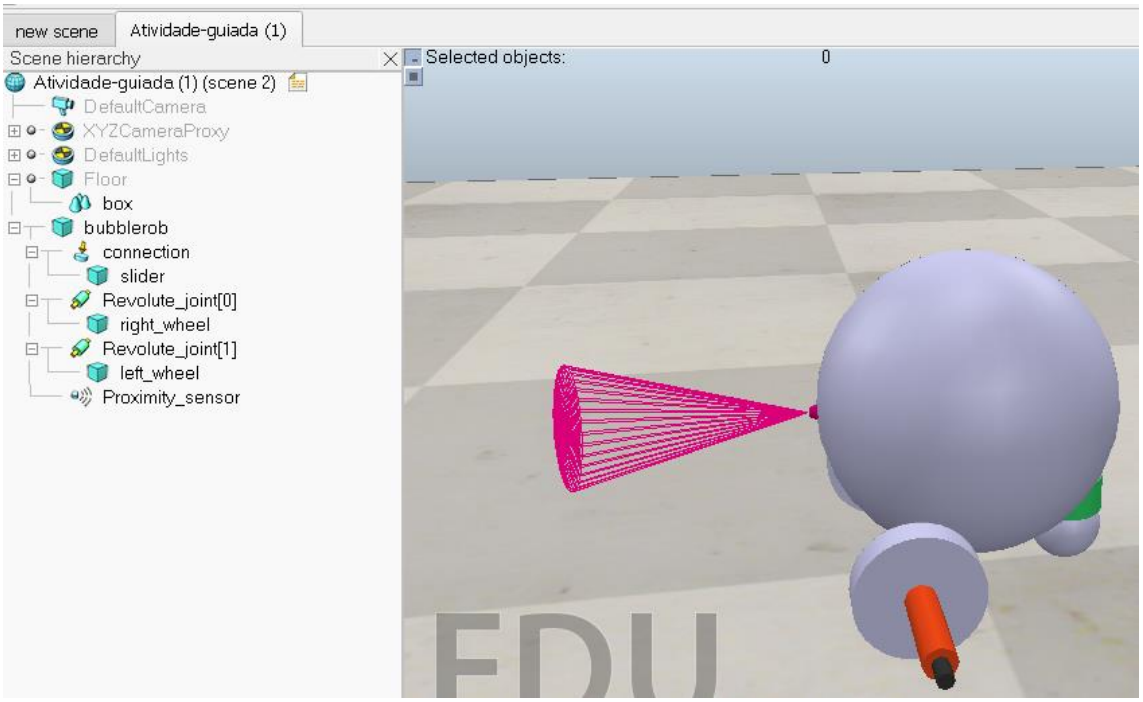
1. Estrutura Hierárquica

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.

Figura 5: Estrutura hierárquica



Autoria: própria

2. Tipos de Objetos

Shapes (Formas): Objetos tridimensionais que podem ser primitivos (cubos, esferas) ou formas mais complexas.

Joints (Juntas): Conectores que permitem a articulação entre objetos, como juntas rotativas e prismaticas.

Sensors (Sensores): Dispositivos que podem detectar várias propriedades do ambiente, como proximidade, visão, etc.

Actuators (Atuadores): Dispositivos que podem aplicar forças ou movimentos, como motores.

Dummy Objects (Objetos Dummy): Objetos auxiliares usados para referência ou pontos de ancoragem.

3. Transformações Hereditárias

Quando um objeto pai é transformado (movido, rotacionado, escalado), seus objetos filhos também sofrem transformações correspondentes. Isso é essencial para simulações em que a movimentação conjunta de vários componentes é necessária.

4. Scene Hierarchy (Janela de Hierarquia de Cena)

No CoppeliaSim, a hierarquia dos objetos é visualizada e manipulada através da janela de Hierarquia de Cena (*Scene Hierarchy*). Aqui você pode:

Arrastar e soltar objetos para mudar suas relações pai-filho.

Expandir ou contrair a visualização da hierarquia para melhor navegação.

Ver e editar propriedades específicas de cada objeto.

5. Manipulação de Hierarquia

Criação: Novos objetos podem ser criados e imediatamente atribuídos como filhos de um objeto selecionado.

Movimentação: Objetos podem ser movidos na hierarquia, alterando suas relações pai-filho.

Eliminação: Ao deletar um objeto pai, todos os seus filhos também são deletados (a menos que sejam reatribuídos).

6. Importância da Hierarquia

A correta configuração hierárquica dos objetos é crucial para simulações precisas. Ela permite que você:

Controle conjuntos de objetos de forma coesa.

Aplique transformações complexas com facilidade.

Gerencie comportamentos dinâmicos de sistemas interligados, como robôs ou veículos.

e) Configurações de propriedade dos objetos

No CoppeliaSim, cada objeto tem um conjunto de propriedades que podem ser configuradas para ajustar seu comportamento, aparência e interação com outros objetos na simulação.

1. Propriedades Comuns para Todos os Objetos

Nome: Identificador único do objeto na cena.

Camadas (*Layers*): Permite definir em quais camadas o objeto está visível e interagível.

Renderização (*Rendering*): Configurações de aparência, como visibilidade, cor e transparência.

Transformações: Posição, rotação e escala do objeto.

2. Propriedades Específicas de Shapes (Formas)

Material: Define as propriedades físicas do objeto, como coeficiente de fricção e densidade.

Dinâmica: Ativa ou desativa a física para o objeto. Pode incluir propriedades como massa, inércia e se o objeto é estático ou dinâmico.

Colisão: Configurações para detectar colisões com outros objetos.

Visualização: Ajuste de aparência, incluindo texturas, cores e iluminação.

3. Propriedades de Joints (Juntas)

Tipo de Junta: Rotativa, prismática ou esférica.

Intervalos de Movimento: Limites de rotação ou translação.

Controlador de Movimento: Modo de controle da junta, que pode ser dinâmico, cinemático ou em modo de posição.

Torque/Força: Define a força máxima que a junta pode aplicar.

Velocidade: Velocidade de movimento da junta.

4. Propriedades de Sensores

Tipo de Sensor: Proximidade, visão, força/torque, etc.

Alcance de Detecção: Distância máxima em que o sensor pode detectar objetos.

Resolução: Qualidade ou precisão da detecção.

Filtro: Tipos de objetos que o sensor pode detectar (por exemplo, apenas formas específicas ou todas as formas).

5. Propriedades de Atuadores

Tipo de Atuador: Pode ser motor ou atuador de força.

Força/Torque Máxima: Valor máximo que o atuador pode aplicar.

Velocidade de Operação: Define a velocidade com que o atuador opera.

Modo de Controle: Dinâmico, cinemático ou por posição.

6. Propriedades de Objetos *Dummy*

Tipo de Referência: Ponto de referência, ponto de destino, etc.

Alvo: Define o objeto ou posição que o dummy deve seguir ou apontar.

Linking: Pode ser usado para criar restrições de movimento ou para fins de simulação de laços fechados.

7. Propriedades Avançadas

Scripts Associados: *Scripts* podem ser associados aos objetos para controlar comportamento dinâmico, interações personalizadas e lógica específica.

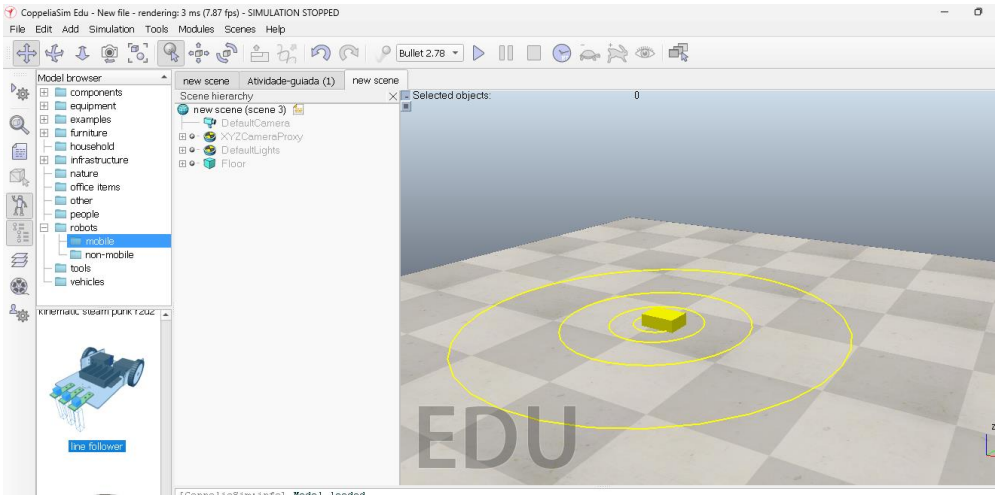
Propriedades de Interface: Parâmetros de comunicação com sistemas externos, como ROS, APIs, etc.

Sinais: Enviar e receber sinais para comunicação entre diferentes partes da simulação.

F) Acessar um modelo

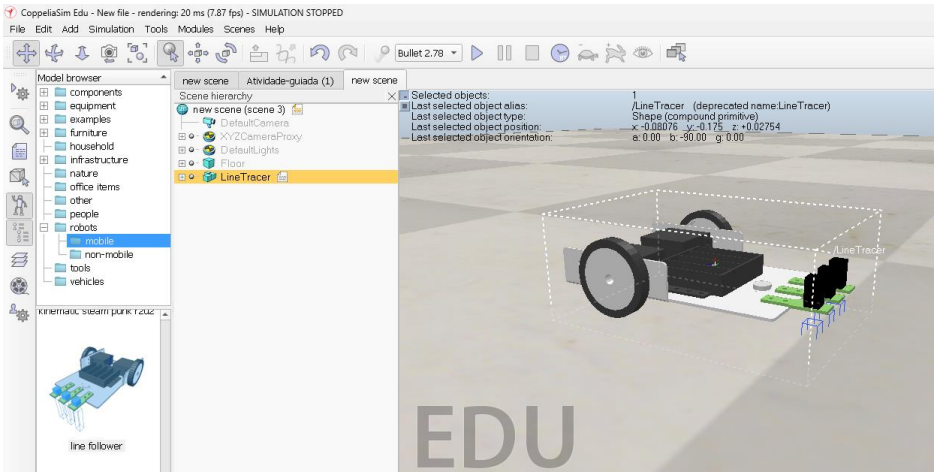
Para acessar um modelo dentro os diversos oferecidos pelo CoppeliaSim, clicamos em uma das opções da seção Model browser do lado esquerdo da tela, onde temos modelos humanoides, braço robóticos, entre outros. Seleciona o modelo desejado e arrasta-lo com o curso do mouse para a nossa cena.

Figura 6: Acessando o modelo line follower.



Autoria: própria

Figura 7: modelo line follower anexado na cena.



Autoria: própria

g) Configuração e utilização da API Remota em Python

A API remota do CoppeliaSim permite que você controle e interaja com as simulações em execução a partir de um script externo. Isso é especialmente útil para integrar o CoppeliaSim com outras ferramentas e linguagens de programação, como Python.