



UNIVERSIDADE FEDERAL DO PIAUÍ
CENTRO DE CIÊNCIAS DA NATUREZA
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA
MESTRADO PROFISSIONAL EM MATEMÁTICA

Cássio Lima Macedo

**FUNÇÕES POLINOMIAIS DO PRIMEIRO E
SEGUNDO GRAU COM O
LUMIBOT-COPPELIASIM**

Teresina - 2023



Cássio Lima Macedo

Dissertação de Mestrado:

**FUNÇÕES POLINOMIAIS DO PRIMEIRO E SEGUNDO
GRAU COM O LUMIBOT-COPPELIASIM**

Dissertação submetida à Coordenação do Programa de Mestrado Profissional em Matemática - Profmat, da Universidade Federal do Piauí, como requisito parcial para obtenção do grau de Mestre em Matemática na modalidade profissional.

Orientador:

Prof. Dr. Cleidinaldo Aguiar Souza.

Coorientadora:

Prof. Dra Valmaria Rocha da Silva Ferraz

Teresina - 2023

Copyright © 2023 by Cássio Lima Macedo.

Direitos reservados, 2023 por Cássio Lima Macedo.

Universidade Federal do Piauí - UFPI, Centro de Ciência da Natureza - CCN, Programa de Pós-Graduação em Matemática, Mestrado Profissional em Matemática. Cep 64049-550 - Teresina, PI.

Nenhuma parte desta dissertação pode ser reproduzida sem a expressa autorização do autor.

FICHA CATALOGRÁFICA
Universidade Federal do Piauí
Biblioteca Comunitária Jornalista Carlos Castello Branco
Divisão de Representação da Informação

M141f **Macedo, Cássio Lima.**
 Funções polinomiais do primeiro e segundo grau com o Lumibot-CoppeliaSim / Cássio Lima Macedo. -- 2023.
 55 f.

Dissertação (Mestrado) – Universidade Federal do Piauí,
 Programa de Mestrado Profissional em Matemática, Teresina, 2023.
 “Orientador: Prof. Dr. Cleidinaldo Aguiar Souza. Coorientadora:
 Prof. Dra Valmaria Rocha da Silva Ferraz.”

1. Matemática. 2. Funções polinomiais. I. Souza, Cleidinaldo
 Aguiar. II. Ferraz, Valmaria Rocha da Silva. III. Título.

CDD 510

Bibliotecária: Milane Batista da Silva – CRB3/1005

Cássio Lima Macedo

**Funções Polinomiais do Primeiro e Segundo Grau com o
Lumibot-CoppeliaSim**

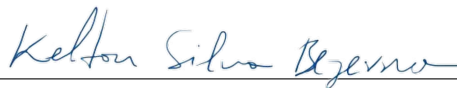
Dissertação submetida à banca examinadora
abaixo discriminada em defesa pública e apro-
vada em 24/02/2023.

BANCA EXAMINADORA



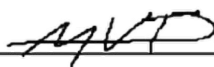
Cleidinaldo Aguiar Souza (Orientador)

Universidade Federal do Piauí



Kelton Silva Bezerra

Universidade Federal do Piauí



Marcos Vinicius Travaglia

Universidade Federal do Piauí

Teresina - 2023

Dedico essa dissertação de mestrado à minha mãe e ao meu pai, que são minha maior fonte de inspiração.

Agradecimentos

Agradeço a Deus.

Agradeço aos meus pais, Antônio Carlos Fernandes de Macedo e Rosana Pereira Lima Macedo, por sempre estarem comigo em todos os momentos.

Aos meus irmãos, por me apoiarem sempre.

Agradeço ao meu orientador, Cleidinaldo Aguiar Sousa, por todos os conselhos, pela paciência e ajuda nesse período.

As minhas amigas Leila Marcia e Neyla Moreira pelos incentivos durante todo período de curso.

Aos professores que ministraram aulas no curso, cada um de vocês foram essenciais para adquirir novas experiências e conhecimentos.

Resumo

Neste trabalho utilizaremos o ambiente de simulação robótica, CoppeliaSim, para aplicarmos o método de aprendizagem STEM.

Mais precisamente, apresentaremos uma abordagem para o conteúdo de funções polinomiais do primeiro e segundo grau através do robô Lumibot dentro do ambiente de simulação CoppeliaSim. Todo este processo faz parte do modo de aprendizagem STEM, onde problemas matemáticos são resolvidos através de interdisciplinaridade com outras áreas apresentadas aos estudantes, de um modo geral. Neste trabalho utilizaremos o modo de aprendizagem STEM como ferramenta educacional para estudantes da educação básica. O método utilizado neste trabalho evidencia como as novas profissões necessitam cada vez mais de criatividade matemática. Consequentemente, este trabalho tem grande potencialidade de despertar esta criatividade matemática necessária os estudantes.

Palavras-Chave: coppeliaSim; STEM; função polinomial do primeiro grau; função polinomial do primeiro grau; lumibot.

Abstract

In this work we will use the robotic simulation environment, CoppeliaSim, to apply the STEM learning method.

More precisely, we will present an approach to the content of polynomial functions of the first and second degree through the Lumibot robot within the CoppeliaSim simulation environment. This whole process is part of the STEM learning mode, where mathematical problems are solved through interdisciplinarity with other areas presented to students, in general. In this work we will use the STEM learning mode as an educational tool for basic education students. The method used in this work shows how new professions increasingly require mathematical creativity. Consequently, this work has great potential to awaken this necessary mathematical creativity in students.

Key words : coppeliaSim; STEM; polynomial functions of the first degree; second degree polynomial functions; lumibot.

Sumário

Resumo	iii
Abstract	iv
Sumário	1
1 CoppeliaSim	5
2 Resultado para Função Polinomial do Primeiro Grau	18
2.1 Cenário 1	19
2.2 Cenário 2	23
2.3 Cenário 3	35
2.4 Cenário 4	38
3 Resultado para Função Polinomial do Segundo Grau	47
3.1 Cenário 1	48
3.2 Cenário 2	50
3.3 Cenário 3	51
3.4 Cenário 4	52
4 Conclusão	55

Introdução

Historicamente, a indústria foi pautada por mudanças tecnológicas e pela a inovação, chamadas de revoluções industriais. A primeira revolução industrial que aconteceu em meados do século XVIII foi caracterizada pelo uso da mecanização, e com esse advento surgiu a máquina de fiar, o tear mecânico, a máquina a vapor, sendo essa última responsável pela revolução no transporte de passageiros e cargas. A segunda surgiu por volta da segunda metade do século XIX e foi marcada pelo uso da energia elétrica, do aço e do petróleo, enquanto a terceira que teve início meados do século XX foi marcada pelo uso da eletrônica e automação. Segundo (Lasi, et al., [9]), todas essas revoluções industriais influenciaram a produção e o sistema de educação. Como consequência novas profissões surgiram e algumas profissões desapareceram. Atualmente estamos diante da quarta revolução industrial, chamada de indústria 4.0, que é caracterizada pela digitalização e pela robótica, ou seja, é baseada em inovações tecnológicas digitais que alteram as interfaces entre o trabalho humano e os processos controlados por sistemas computacionais.

Assim como as revoluções industriais anteriores (primeira, segunda e terceira revolução), a nova indústria tem feito mudanças no sistema educacional com um expressivo diferencial- atualmente a velocidade com que novas tecnologias são desenvolvidas está varias vezes à frente do ritmo das mudanças do sistema de educação. Isto tem feito com que vários países priorizem o processo de capacitação para atender a indústria 4.0.

É cada vez mais evidente que as novas tecnologias estão fundindo os mundos físico, digital e biológico, influenciando toda sociedade e modificando todas disciplinas. Em (Rosa e Orey , [14]), os autores apontam que a industria 4.0 exige que os humanos tenham além do conhecimento, habilidades para colaborar, resolver problemas, pensar criticamente e trabalhar em equipe. É um desafio fazer com que os estudantes de hoje tenham criatividade e habilidade necessária para trabalhar com as novas tecnologias, e no futuro possam desenvolver novas tecnologias importantes para a sociedade.

Uma alternativa para preparar gerações para os desafios das novas profissões é o modo de aprendizagem STEM que inglês significa (Science, Technology, Engineering, and Mathematics). Este modo de aprendizagem nos anos de 1990 foi originalmente chamado pela National Science Foundation (NSF) de SMET (Science, Mathematics, Engineering, and Technology). Segundo (Hallinen, [6]), a sigla SMET foi reordenada em 2001 para a sigla STEM pela bióloga americana Judith Ramaley.

Em (Bybee, [1]), o autor define STEM como a interdisciplinaridade entre ciência, tecnologia, engenharia e matemática. Esta definição concorda com a definição dada em (Gonzalez e Kuenzi, [4]). Em (Yildirim e Selvi, [20]), os autores definem STEM como o processo de aprendizagem através de problemas da vida real. Além disso, em (Pratama et al., [12]) os autores afirmam que o uso de STEM é adequado para o ensino e aprendizagem, desenvolvendo um pensamento criativo e crítico dos alunos. Por sua vez, (Struyf, [15]) descobriu que STEM é uma excelente abordagem para promover o envolvimento dos alunos na aprendizagem, pois os alunos são fisicamente e emocionalmente envolvidos no ambiente de aprendizagem.

Basicamente temos uma ampla gama de definições para o modo de aprendizagem STEM, algumas diferindo de outras apenas no que diz respeito ao seu significado. Independente da definição que é dada para o modo de aprendizagem STEM, de acordo com (Shaughnessy, [16]) todas tem um objetivo em comum, focar na resolução de problemas baseado em conceitos e processos a partir de ciências e matemática utilizando tecnologias aplicada a técnicas de engenharias

O modo de aprendizagem STEM tem sido desenvolvido em muitos países, como por exemplo: Taiwan (Chen e Lin, [2]), Estados Unidos (Gonzalez e Kuenzi, [3]), Coreia (Park et al., [13]), Suíça (Hinojo-Lucema, et al., [7]), Japão (Yata, et al., [21]), Finlândia, Noruega, Rússia (Tomperi et al., [18]) e muitos outros. Recentemente, (Oliveira e Souza, [11]) fizeram uma breve abordagem desta interdisciplinaridade através de matrizes de ordem dois, posteriormente (De Aguiar e Souza, [3]), expandiram esta ideia utilizando matrizes de ordem três.

De acordo com (Naya, et al., [10]) e (Takacs, et al., [17]) uma maneira de combinar todas as áreas do modo de aprendizagem STEM é utilizando robótica no processo de ensino e aprendizagem. Além disso, através da robótica a experiência de aprendizagem fica mais atraente, pois as plataformas são compostas por recursos que permitem prender a atenção dos alunos por mais tempo, ao passo em que ele aprendem construindo. Em toda educação básica podemos utilizar robôs como ferramenta educacional. Em particular, no ensino médio o professor em colaboração com o aluno podem aprofundar a capacidade robótica e as aplicações.

A robótica como ferramenta educacional está em rápida expansão, onde professores, pesquisadores e empresas estão caminhando em conjunto para criarem um novo ambiente de aprendizagem nas escolas. Existem várias plataformas robóticas para fins educacionais. As plataformas robóticas educacionais mais populares são as seguinte: AlphaBot2, Lego Ev3, Dash&Dot, Edison, EUROPA, Ranger, Robobo, Mbot, ThymioII. Existem ainda simuladores robóticos voltados para a indústria 4.0 que embora não sejam plataformas educacionais podem ser facilmente adaptadas para o modo de aprendizagem STEM, como por exemplo: CoppeliaSim e RobotStudio.

Neste trabalho utilizaremos a robótica aplicada ao modo de aprendizagem STEM. Mais precisamente, utilizaremos o simulador robótico CoppeliaSim, no processo de ensino

aprendizagem de funções polinomiais do primeiro e segundo grau, onde apresentaremos uma maneira do aluno aprender funções resolvendo problemas atuais do mundo real, através destas ferramentas. Este trabalho está organizado da seguinte maneira: inicialmente o trabalho é formado por esta breve introdução; na seção (2) apresentamos o ambiente de simulação CoppeliaSim; na seção (3) através da função polinomial do primeiro grau, daremos uma funcionalidade para o robô, trabalhando diretamente com problemas do mundo real; na seção (3) por meio da função polinomial do segundo grau, o robô também terá funcionalidades ligadas a situações cotidianas; seção (4) é dedicada para a conclusão do trabalho.

Capítulo 1

CoppeliaSim

Nesta seção faremos uma breve apresentação do ambiente de simulação CoppeliaSim, onde passaremos uma ideia do seu funcionamento e apresentaremos o robô que utilizaremos ao longo do trabalho.

CoppeliaSim, antigamente chamado de V-REP, é um simulador robótico para dinâmica de corpos rígidos, que simula com um alto grau de precisão as partes que compõem um robô e sua interação com o ambiente. O CoppeliaSim foi desenvolvido pela Toshiba e atualmente pertence a empresa Coppelia Robotics AG, localizada em Zurique na Suíça. Existem três versões do software CoppeliaSim: player, edu e pró. A versão pró é de uso puramente comercial, enquanto que as versões player e edu são versões livres. Neste trabalho utilizaremos a versão edu, que é a versão do software gratuita para estudantes, professores e pesquisadores.

O CoppeliaSim é um simulador multiplataforma que suporta um ambiente de desenvolvimento integrado, podendo se comunicar com vários ambientes de programação, ou seja, o software possui uma flexibilidade e uma portabilidade que facilita o ajuste do código de programação. O usuário tem a possibilidade de escolher entre utilizar a linguagem de programação do CoppeliaSim, chamada de Lua, ou utilizar outras linguagens, como por exemplo Python.

Neste trabalho utilizaremos a linguagem de programação Python, onde o código será escrito e compilado através do ambiente de desenvolvimento integrado, PyCharm. Ou seja, integraremos o PyCharm ao CoppeliaSim para sistema operacional windows. Para que a integração aconteça, os seguintes softwares devem estar instalados no PC que será utilizado:

- CoppeliaSim Edu 4.0, que pode ser baixado através do link:
[https:// www.coppeliarobotics.com/downloads](https://www.coppeliarobotics.com/downloads).
A escolha desta versão se deve fato que a mesma possui o robô que iremos trabalhar;
- Python utilizada deve ser a versão 2.7.18, que pode ser baixada através do link:
<https://www.python.org/downloads/release/python-270/> .

A escolha desta versão é justamente para o bom funcionamento com a versão do CoppeliaSim escolhida;

- PyCharm, que pode ser baixado através do link:
<https://www.jetbrains.com/pycharm/download/#section=windows>.
Sugiro escolher a versão comunidade, que é mais leve que a profissional. Após baixar o PyCharm crie uma conta como professor ou aluno para utilizar o software.

Com os três softwares já instalados, integraremos o CoppeliaSim com o PyCharm, obtendo assim um ambiente adequado para o nosso trabalho, que não é construir algoritmo. Ou seja, neste trabalho apresentaremos o código que deve ser digitado no PyCharm. O procedimento para integrar o PyCharm com o CoppeliaSim é feito de modo bem simples seguindo os passos abaixo:

- (1) vá até a pasta onde está instalado o CoppeliaSim, abra a pasta programming, depois abra a pasta remoteApiBindings, em seguida abra a pasta python, copie os arquivos: sim.py e simConst.py;
- (2) procure a pasta PycharmProjects, abra a pasta Project e cole dentro desta pasta os arquivos copiados no passo anterior;
- (3) retorne para a pasta remoteApiBindings, abra a pasta lib, depois a pasta windows copie o arquivo remoteApi.dll e cole este arquivo junto aos demais arquivos do passo anterior na pasta Project do PycharmProjects;
- (4) finalmente estão interligados o CoppeliaSim com o python, agora é só abrir o CoppeliaSim e o PyCharm para que possamos trabalhar.

O CoppeliaSim é uma plataforma dinâmica, bastante interativa e muito elegante. Agora teremos contato com o Lumibot, robô que será utilizado ao longo deste trabalho. A seguir apresentaremos o passo a passo para utilização desta plataforma. Embora esteja numerado cada passo, não significa que esta deva ser a ordem adota.

- (1) Na barra de botão modelo as pastas são divididas por categorias, selecionamos na pasta robot a subpasta mobile indicada através da seta em vermelho, como ilustra a Figura 1.1.

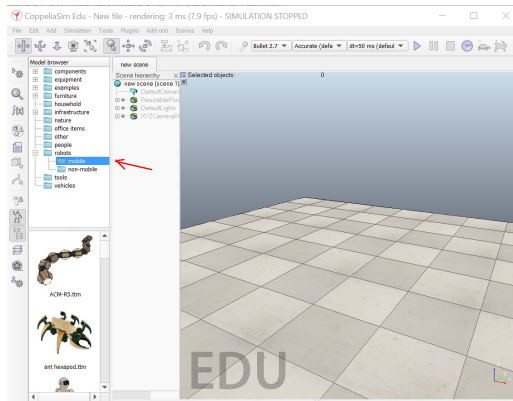


Figura 1.1: selecionando mobile

- (2) Dentro do ambiente robot escolhemos o robô Lumibot e arrastamos com o mouse para a área de trabalho, como ilustra a Figura 1.2.

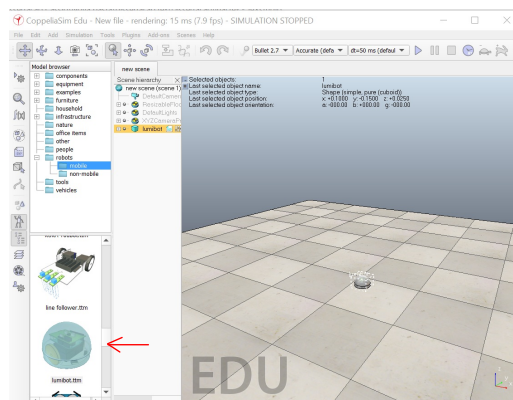


Figura 1.2: selecionando o robô Lumibot

- (3) Temos o Lumibot na área de trabalho, indicado através da seta em vermelho, como ilustra a Figura 1.3

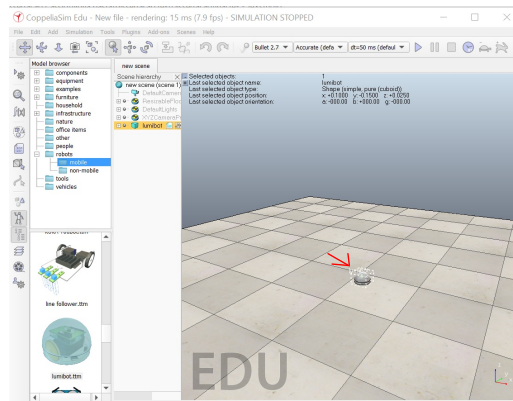


Figura 1.3: Lumibot na área de trabalho

- (4) Clicando com o botão direito do mouse em qualquer local da área de trabalho, aparecerá uma caixa, onde podemos clicar em add, em seguida escolher a opção primitive shape e então a opção disc. Automaticamente o círculo e qualquer outra forma primitiva será colocada na origem do plano que representa a área de trabalho, como ilustra a Figura 1.4.

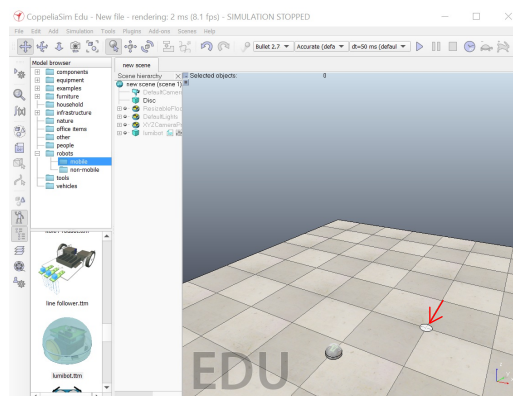


Figura 1.4: Inserindo o disco

- (5) Fazemos o mesmo que o passo anterior e escolhemos a opção plane indicada através da seta em vermelho, como ilustra d Figura 1.5.

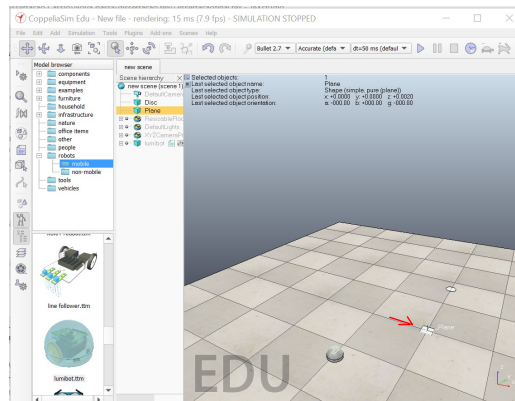


Figura 1.5: Inserindo o plano

(6) Podemos alterar o nome dos objetos que estão na área de trabalho de duas maneiras:

- clicando diretamente sobre o objeto.
- selecionando os objetos diretamente clicando sobre os seus nomes, indicado através da seja em vermelho, como ilustra a Figura 1.6.

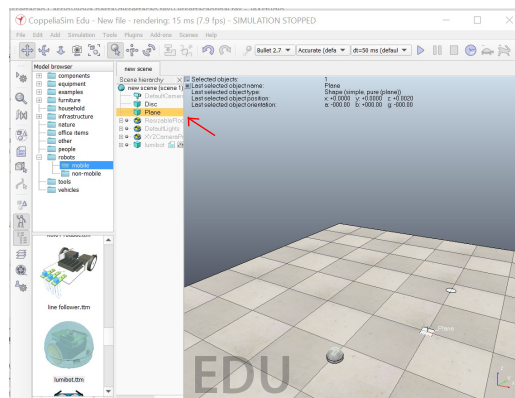


Figura 1.6: Alterando o nome do objeto

(7) Após selecionar os objetos que estão na área de trabalho, clicando sobre o ícone indicado através da seta em vermelho, como ilustra a Figura 1.7,

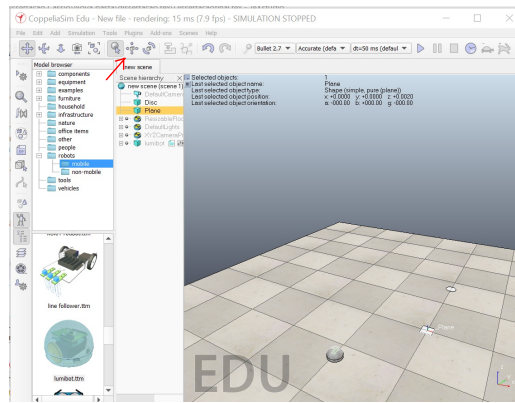


Figura 1.7: Ícone para mudança de posição dos objetos

aparecerá uma caixa indicada através da seta em vermelho, como ilustra a Figura 1.8 que nos permite modificar a posição dos objetos na área de trabalho.

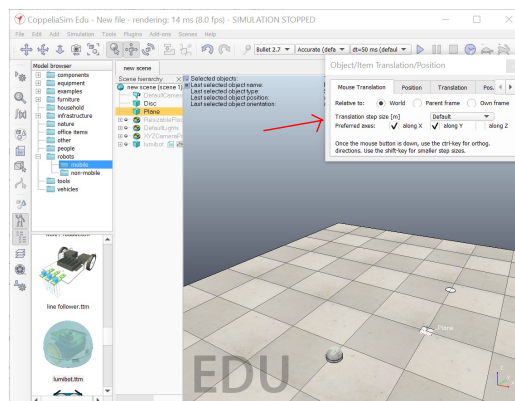


Figura 1.8: Caixa que permite alterar as posições

(8) Clicando sobre este ícone indicado através da seta em vermelho, como ilustra a Figura 1.9

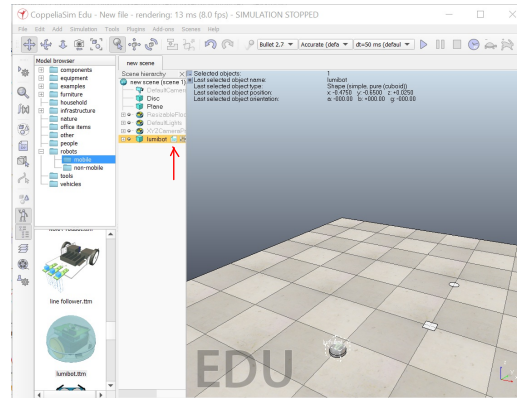


Figura 1.9: Seleção do código em Lua

abrimos o código em linguagem Lua do Lumibot, como ilustra a Figura 1.10

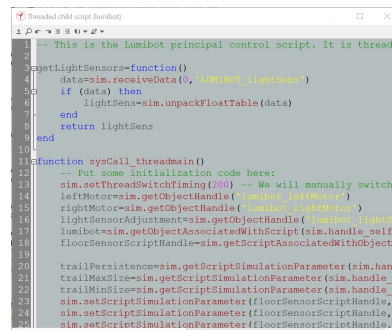


Figura 1.10: Código em Lua

apagaremos todo o código Lua que está dentro desta caixa, e digitaremos `simRemoteApi.start(19999)`, como ilustra a Figura 1.11 em seguida fechamos a caixa.

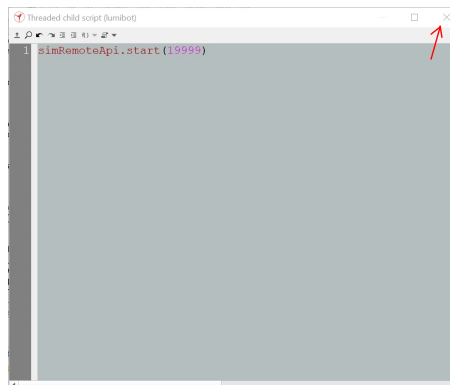


Figura 1.11: Excluindo o código em Lua

- (9) Na barra de botão modelo, podemos selecionar diversos objetos, como por exemplo garras, plantas, cadeiras, janelas, etc, como ilustra a Figura 1.12.

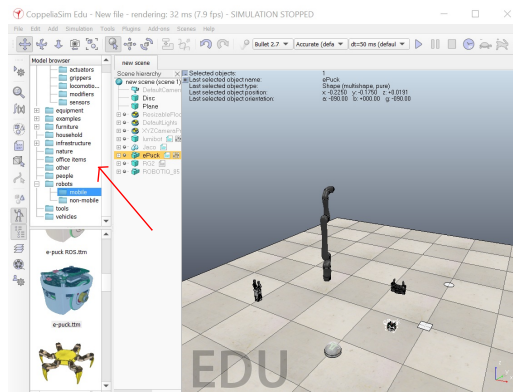


Figura 1.12: Colocando vários objetos sobre o plano

- (10) Iniciamos o processo que o Lumibot ou outros robôs executarão, através do ícone de start indicado através da seta em vermelho, como ilustra a Figura 1.13.

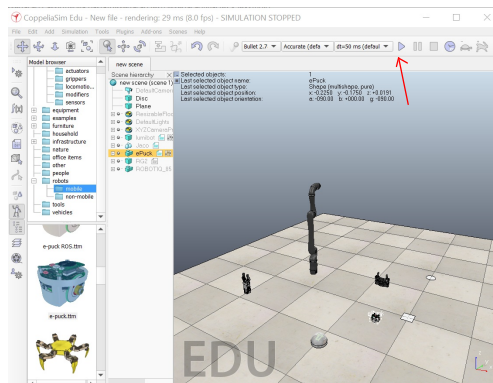


Figura 1.13: Iniciando o funcionamento

Para os cenários que iremos trabalhar na próxima seção, o processo de construção é semelhante a este.

O Lumibot é um robô móvel autônomo com duas rodas, envolvido por uma carenagem de formato esférico com 0,2 metros de diâmetro em sua base e por 0,2 metros de altura. Sob sua carenagem transparente que se assemelha a uma concha, podem-se ver seus equipamentos eletrônico que se iluminam quando o robô esta em funcionamento. Este robô é parecido com criaturas que moram no fundo do mar ou águas vivas, como ilustra a Figura 1.14. O Lumibot, quando esta em movimento, deixa rastros brilhantes que desaparecem lentamente.



Figura 1.14: Lumibot real

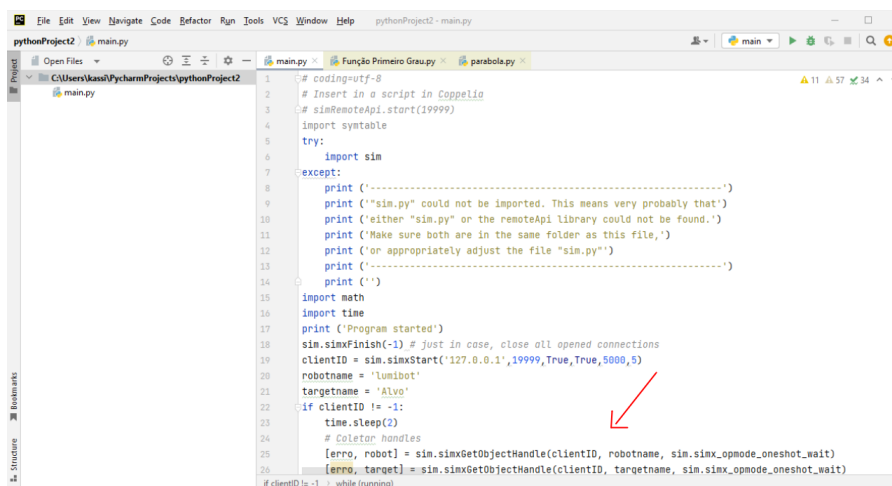
A versão virtual do Lumibot produzida dentro da plataforma de simulação CoppeliaSim, reproduz com fidelidade a versão real do robô, como ilustra a Figura 1.15.



Figura 1.15: Lumibot versão virtual

Por sua vez, a seguir indicaremos os passos que serão aplicados neste trabalho para a utilização do PyCharm.

- (1) Digitamos o código em linguagem Python que será utilizado com o CoppeliaSim neste ambiente indicado através da seta em vermelho, como ilustra a Figura 1.16.



```
1 # coding=utf-8
2 # Insert in a script in Coppelia
3 # simRemoteApi.start(19999)
4 import sys
5 try:
6     import sim
7 except:
8     print ('-----')
9     print ("sim.py" could not be imported. This means very probably that)
10    print ("either "sim.py" or the remoteApi library could not be found.")
11    print ("Make sure both are in the same folder as this file,")
12    print ("or appropriately adjust the file "sim.py"")
13    print ('-----')
14    print ('')
15 import math
16 import time
17 print ('Program started')
18 sim.simxFinish(-1) # just in case, close all opened connections
19 clientID = sim.simxStart('127.0.0.1',19999,True,True,5000,5)
20 robotname = 'Lumibot'
21 targetname = 'Alvo'
22 if clientID != -1:
23     time.sleep(2)
24     # Coletar handles
25     [erro, robot] = sim.simxGetObjectHandle(clientID, robotname, sim.simx_opmode_oneshot_wait)
26     [erro, target] = sim.simxGetObjectHandle(clientID, targetname, sim.simx_opmode_oneshot_wait)
27 # clientID != -1 while (running)
```

Figura 1.16: Ambiente de digitação

- (2) O código é numerado por linhas e colunas, cada linha dessa indicada através da seta em vermelho, como ilustra a Figura 1.17, chama-se linha de comando.

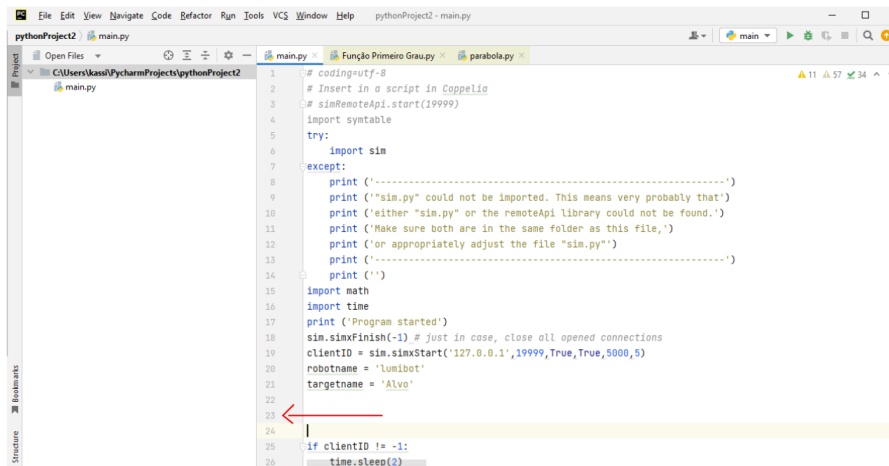


Figura 1.17: Linhas e colunas

- (3) Podemos abrir um código que esteja salvo, bem como salvar o código que acabamos de criar utilizando o ícone indicado através da seta em vermelho, como ilustra a Figura 1.18.

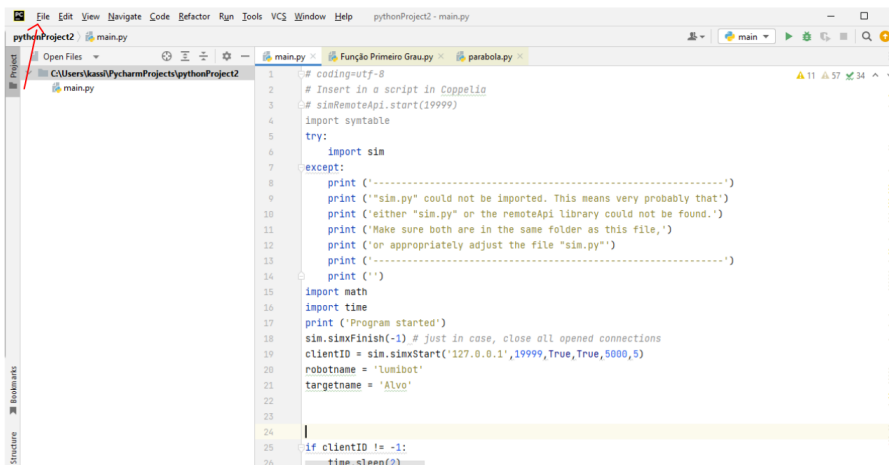


Figura 1.18: Ícone para abrir e salvar um código

- (4) Os códigos que estão salvos dentro da pasta Project, assim como os arquivos que copiamos, quando estávamos interligando o CoppeliaSim com o PyCharm, da pasta onde está instalado o CoppeliaSim ficam todos listados nesta parte indicada pela seta em vermelho, como ilustra a Figura 1.19.

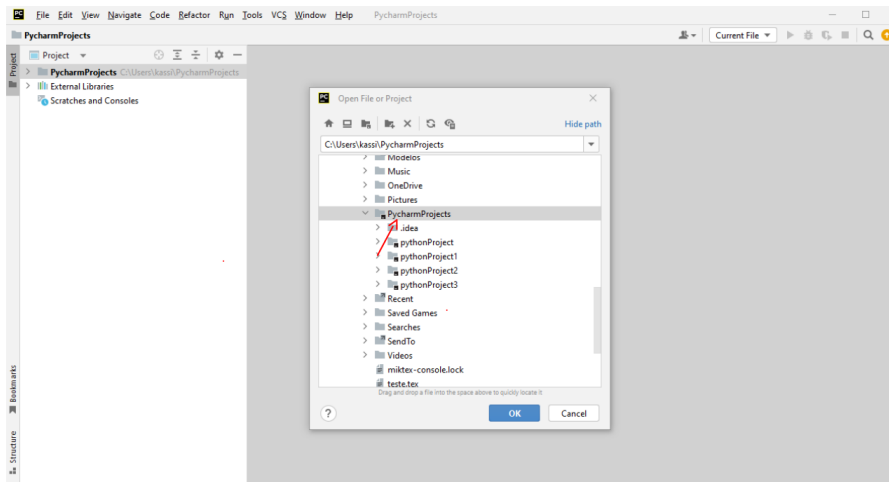


Figura 1.19: Os arquivos que já forma criados

- (5) Quando este símbolo amarelo indicado através da seta em vermelho, como ilustra a Figura 1.20, fica vermelho é uma aviso de que o código tem erros, além de avisar a quantidade de erros que temos.

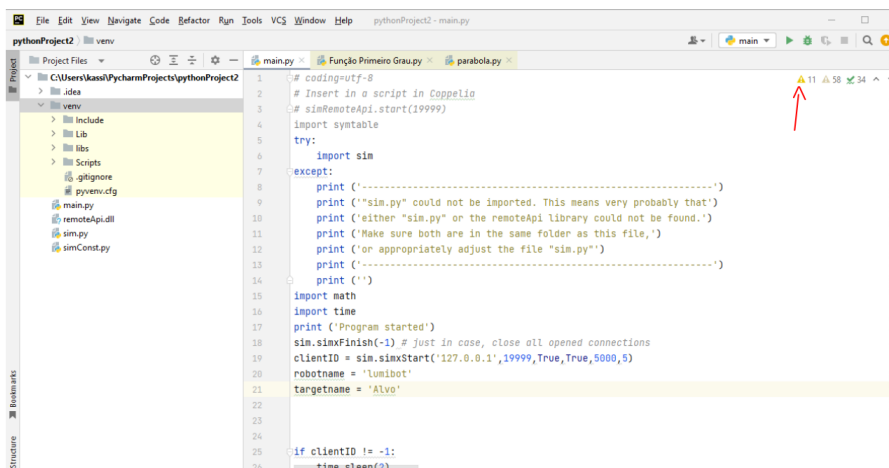


Figura 1.20: Indicação de erros dentro do código

- (6) Quando mais de um código estiverem abertos podemos escolher qual o código será compilado, utilizando o ícone indicado através da seta em vermelho, como ilustra a Figura 1.21. Sugiro que a opção que aí aparece, Current File, esteja sempre selecionada.

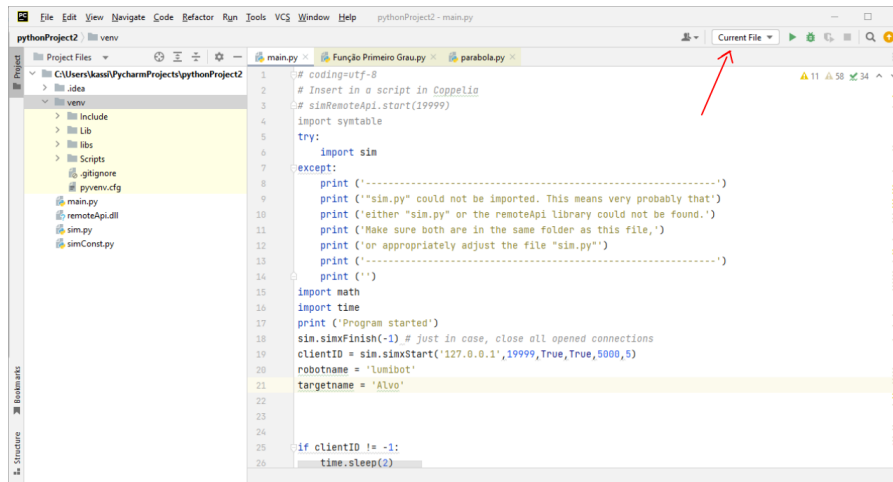


Figura 1.21: Ícone para escolha do código

- (7) Para compilar (ou iniciar) o código clicamos neste ícone indicado através da seta em vermelho, como ilustra a Figura 1.22.

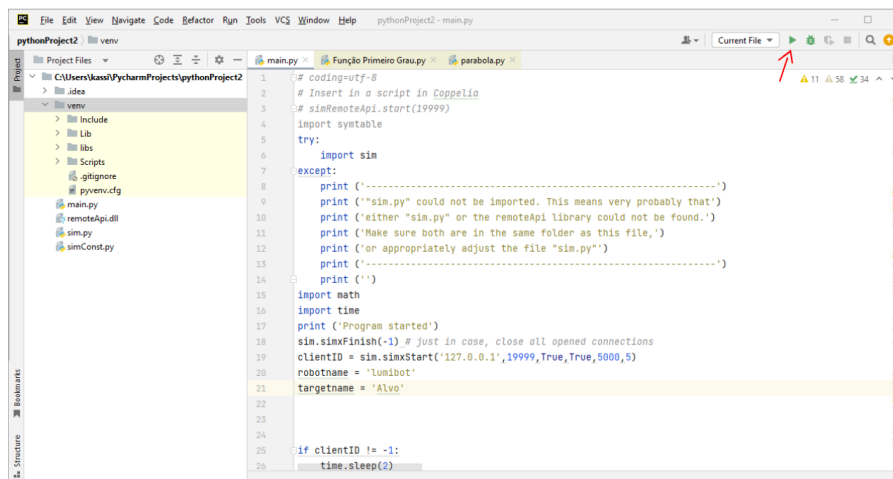


Figura 1.22: Botão iniciar ou compilar

- (8) Quando a compilação tiver sido um sucesso a mensagem indicada através da seta em vermelho, como ilustra a Figura 1.23 deverá ser exibida. Caso contrário, o código foi digitado de modo errado.

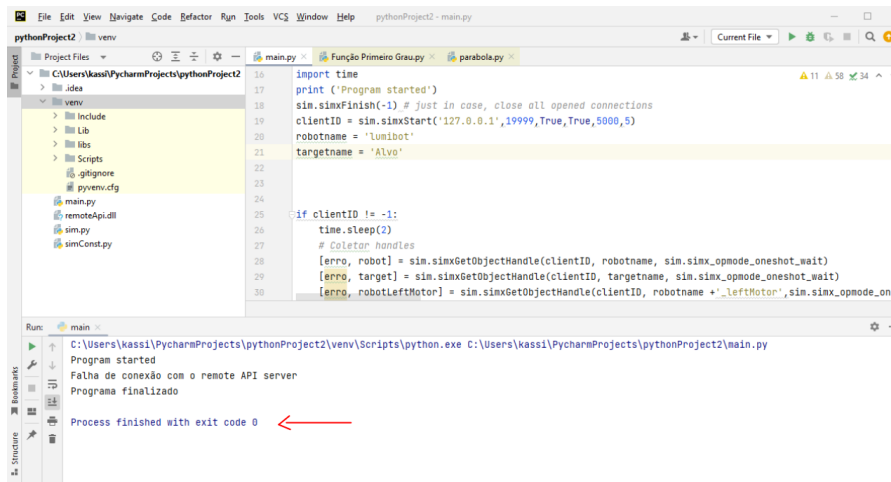


Figura 1.23: Mensagem de sucesso

Fique atento, pois cada frase do código que apresentaremos deve ser digitado, rigorosamente, obedecendo a mesma ordem para linhas e colunas. Tudo ocorrendo bem, o código já digitado, iremos colocar o CoppeliaSim para funcionar através do PyCharm. Para isto, inicie o processo no CoppeliaSim (passo (10) ilustrado na Figura 1.13), em seguida execute o código com PyCharm (passo (7) ilustrado na Figura 1.22).

Finalizamos nesta seção 1 as explicações das instalações dos softwares. Nas próximas duas seções (seção 2 e 3) apresentaremos o código em Python que será utilizado neste trabalho (veja Figuras 30 e 31 na seção 2 e Figuras 81 e 82 na seção 3).

Capítulo 2

Resultado para Função Polinomial do Primeiro Grau

Nesta seção utilizaremos funções polinomiais do primeiro grau para que possamos dá funcionalidade para o Lumibot apresentando uma maneira de resolver um problema do mundo real.

Função polinomial do primeiro grau, é um dos principais conteúdos ensinado no ensino médio. É de grande importância que os estudantes dominem completamente a função polinomial do primeiro grau, isto significa entender suas características e possíveis aplicações.

O método de ensino, comumente, utilizado nas escolas de ensino médio consiste em explicar a definição de função, fazer exemplos e apresentar suas diferentes formas. A forma mais comum utilizada é dada da seguinte maneira:

$$f(x) = ax + b \quad a \neq 0. \quad (2.1)$$

Assim, partindo de (2.1) explora-se a raiz, o gráfico que é uma reta, a intersecção com o eixo vertical, sinal, monotonicidade, e os casos em que a reta passa por cada um dos quatro quadrantes.

Um dos requisitos mais importantes para os alunos é que eles entendam corretamente a conexão de dependência dos coeficientes a , b em (2.1). De certa forma, conhecendo estes coeficientes a função polinomial do primeiro grau fica completamente determinada. Segue-se que conhecendo pelo menos dois valores para x e $f(x)$ em (2.1), obtemos os coeficientes a , e b , e conseqüentemente toda a função $f(x)$.

Dividiremos esta seção em quatro subseções que serão chamadas de cenários. Cada cenário ilustra uma real possibilidade, em que encontraremos uma função polinomial do primeiro grau que será utilizada para determinar a trajetória do robô Lumibot dentro do ambiente CoppeliaSim.

2.1 Cenário 1

A primeira situação que consideraremos é básica, onde construiremos um cenário através do CoppeliaSim. Com o CoppeliaSim aberto, escolhemos o robô que será utilizado clicando sobre o ícone indicado através da seta em vermelho, como ilustra a Figura 2.1

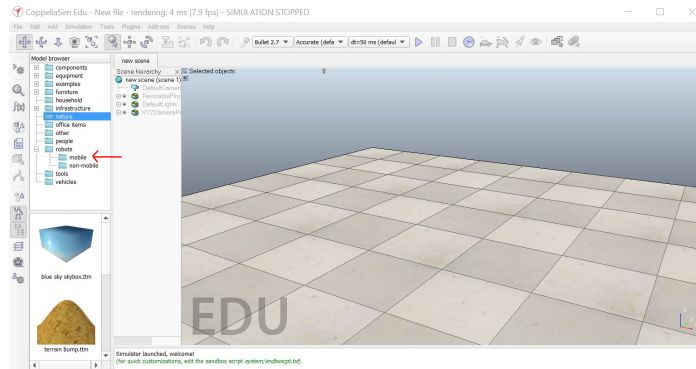


Figura 2.1: Escolha do robô

Em seguida seguimos os passos ilustrados na Figura 2.2

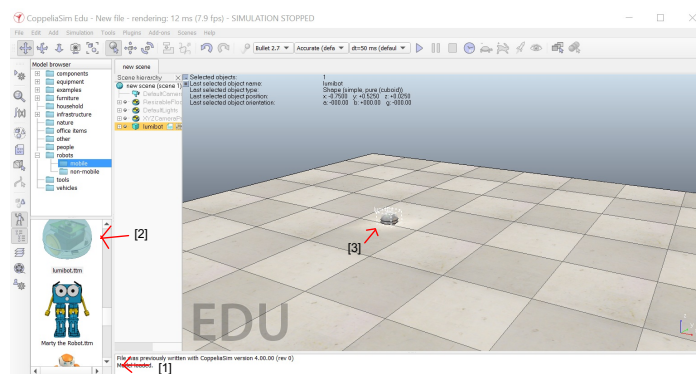


Figura 2.2: Colocando o Lumibot no ambiente

que são descritos da seguinte maneira:

- [1] rolamos para baixo até encontramos o robô Lumibot;
- [2] clicamos sobre o Lumibot e arrastamos para a área de trabalho do CoppeliaSim;
- [3] estamos diante de Lumibot.

Já com o Lumibot na área de trabalho do CoppeliaSim, adicionaremos algumas formas primitivas, a primeira forma primitiva será o disco, que obtemos clicando com o botão

direito do mouse sobre a área de trabalho, escolhendo a opção add, depois primitive shap, e então escolhemos a opção disc. O disco ficará posicionado da origem do plano cartesiano que representa o piso, como indica a seta [2] ilustrada na Figura 2.3

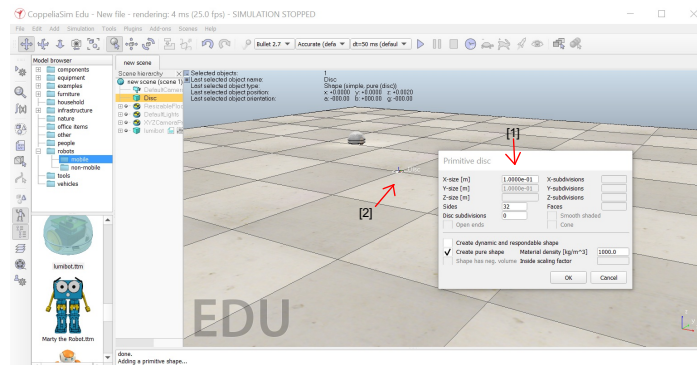


Figura 2.3: Escolha de formas primitivas

No instante que selecionamos o disco, aparecerá a caixa indicada através da seta [1], como ilustra a Figura 2.3. Nesta caixa mudaremos as dimensões do disco.

Em seguida adicionaremos outra forma primitiva, seguindo o mesmo procedimento anterior. As formas primitivas escolhidas desta vez serão planos, como ilustra a Figura 2.4

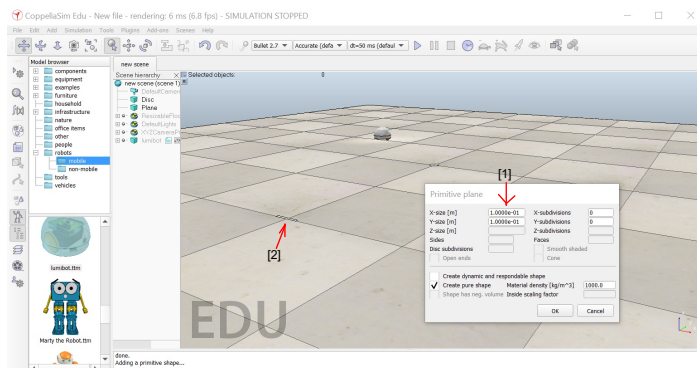


Figura 2.4: Forma primitiva

Através do procedimento indicado com a seta [1], como ilustra a Figura 2.4, modificamos as dimensões do plano. Aqui mudaremos para cada plano adicionado a coordenada x preservando a coordenada y, ou o contrário. Com isto conseguimos produzir os lados do quadrado e da figura X, como ilustra a Figura 2.5

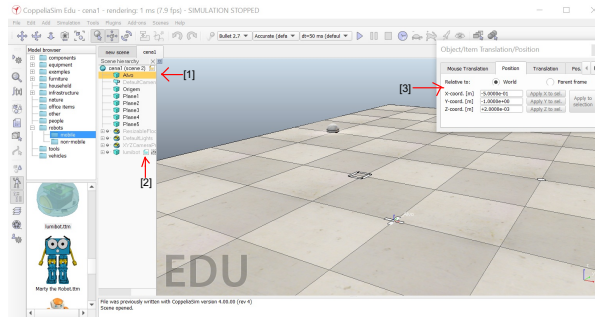


Figura 2.5: Modificando o código

Para finalizar o cenário seguiremos os passos indicados na Figura 2.5, que são dados da seguinte maneira:

- [1] após selecionar o disco, com este procedimento mudaremos o nome do disco para Alvo;
- [2] Clicando sobre este ícone abriremos o código em Lua do Lumibot, que devemos apagar e escrever `simRemoteApi.start(19999)`;
- [3] após selecionar cada objeto presente sobre o piso, com este procedimento podemos colocar os objetos sobre posições escolhidas sobre o plano.

Estamos diante do cenário que é formado apenas por um piso de centro na origem que é exatamente o círculo branco, formado por quadrados de lados medindo 0.5 metros, ilustrado na Figura 2.6.

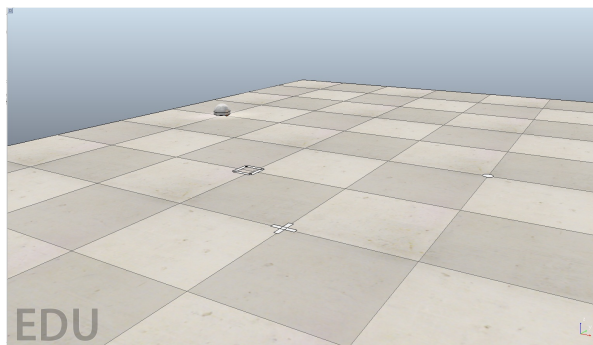


Figura 2.6: Plano centrado sobre o círculo

Partindo do círculo como referência, o robô Lumibot está localizado na posição $(-2, 0.5)$, o quadrado branco em que o robô deve passar, com lados medindo 0.01 metros está localizado na posição $(-1, -0.5)$, e o alvo que o Lumibot deve atingir representado

por uma cruz localizada na posição $(-0.5, -1)$, como ilustra a Figura 2.6. Todas estas posições são consideradas em relação ao centro de massa dos objetos (que são o centro de cada objeto) e são dadas em metros.

Queremos encontrar uma trajetória para o Lumibot sair de sua posição inicial, passar pelo quadrado branco e chegar até o alvo. Considerando a expressão (2.1), através dos valores das posições dadas acima, obtemos que a trajetória pode ser descrita através do gráfico da função:

$$f(x) = -x - \frac{3}{2}.$$

Assim, através da função polinomial acima e seguindo as notações do código que será escrito no PyCharm, como ilustra as Figuras 2.7 – 2.8, temos que

$$f(xp) = -xp - \frac{3}{2}$$

```

1 # coding=utf-8
2 # Insert in a script in Coppelia
3 # simonets@pi.start(1999)
4 import sys
5 try:
6     import sim
7 except:
8     print('-----')
9     print("'sim.py' could not be imported. This means very probably that:")
10    print('either "sim.py" or the remotapi library could not be found.')
11    print('Make sure both are in the same folder as this file,')
12    print('or appropriately adjust the file "sim.py"')
13    print('-----')
14    print('')
15 import math
16 import time
17 print ('Program started')
18 sim.simFinish(-1) # just in case, close all opened connections
19 clientID = sim.simxStart('127.0.0.1',1999,True,True,5000,5)
20 robotname = 'lumibot'
21 targetname = 'Alvo'
22 if clientID != -1:
23     time.sleep(2)
24     # Colocar handles
25     [erro, robot] = sim.simGetObjectHandle(clientID, robotname, sim.simx_opmode_oneshot_wait)
26     [erro, target] = sim.simGetObjectHandle(clientID, targetname, sim.simx_opmode_oneshot_wait)
27     [erro, robotLeftMotor] = sim.simGetObjectHandle(clientID, robotname + '.leftMotor', sim.simx_c
28     [erro, robotRightMotor] = sim.simGetObjectHandle(clientID, robotname + '.rightMotor', sim.simx_opmode_oneshot_wa
29     # Definir posição de ambos
30     [erro, positionRobot] = sim.simGetObjectPosition(clientID, robot, -1, sim.simx_opmode_streaming)
31     [erro, positionTarget] = sim.simGetObjectPosition(clientID, target, -1, sim.simx_opmode_streaming)
32     [erro, orientationRobot] = sim.simGetObjectOrientation(clientID, robot, -1, sim.simx_opmode_streaming)
33     time.sleep(2)
34     # Posicionar os motores
35     sim.simSetJointTargetVelocity(clientID, robotRightMotor, 0.0, sim.simx_opmode_oneshot)
36     sim.simSetJointTargetVelocity(clientID, robotLeftMotor, 0.0, sim.simx_opmode_oneshot)
37     state = 'stopped'
38     dist_set = 0.1
39     vref = 0.5
40     running = True
41     while(running):
42         # Coletar dados robôs e alvos
43         [erro, [xv, yv, zv]] = sim.simGetObjectPosition(clientID, robot, -1, sim.simx_opmode_buffer)
44         [erro, [xt, yt, zt]] = sim.simGetObjectPosition(clientID, target, -1, sim.simx_opmode_buffer)
45         [erro, [alpha, beta, gamma]] = sim.simGetObjectOrientation(clientID, robot, -1, sim.simx_opmode_buffer)
46         dist = math.sqrt((xt-xv)**2+(yt-yv)**2+(zt-zv)**2)
47         ap = -3
48         bp = -3/2
49         xv = -0.5
50         xf = -1
51         #Transições
52         if (state == 'stopped') & (dist > dist_set):

```

Figura 2.7: Código em Python

Continuação do código

```

53     if (state == 'stopped') & (dist > dist_set):
54         state = 'align'
55     elif (state == 'align') & (gamma < (-math.pi/2)+math.atan(ap)):
56         state = 'forward'
57     elif (state == 'forward') & (dist < dist_set):
58         state = 'stopped'
59     #Ações
60     if state == 'stopped':
61         vRightMotor = 0.0
62         vLeftMotor = 0.0
63         running = False
64     elif state == 'align':
65         vRightMotor = -vref
66         vLeftMotor = vref
67     elif state == 'forward':
68         vRightMotor = ((xi-xf)/abs(xi-xf))*v
69         vLeftMotor = ((xi-xf)/abs(xi-xf))*v
70     # Comandos motores
71     sim.simPauseCommunication(clientID, True)
72     sim.simSetJointTargetVelocity(clientID, robotRightMotor, vRightMotor, sim.simx_opmode_oneshot)
73     sim.simSetJointTargetVelocity(clientID, robotLeftMotor, vLeftMotor, sim.simx_opmode_oneshot)
74     sim.simPauseCommunication(clientID, False)
75     sim.simPauseSimulation(clientID, sim.simx_opmode_oneshot_wait)
76     sim.simFinish(clientID)
77 else:
78     print ('Falha de conexão com servidor API remoto')
79     print ('Program ended')
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Figura 2.8: Continuação do Código

este mesmo código será utilizado para os Cenários 2, 3 e 4 a seguir.

Substituiremos os valores diretamente no código ilustrado nas Figuras 2.7 – 2.8. Iniciamos com o valor da constante $ap = -1$ que será substituído na linha 47, depois o valor da constante $bp = \frac{3}{2}$ que será substituído na linha 48, em seguida o valor inicial

de x , $x_i = -2$, será substituído na linha 49, e por fim o valor final de x , $x_f = -\frac{1}{2}$, que será substituído na linha 50. Segue-se, que o Lumibot percorrerá o trajeto passando pelo quadrado até atingir o obstáculo, com ilustra a Figura 2.9. Todo o percurso pode ser acompanhado através do link <https://youtu.be/wUoLiCAfXuY>

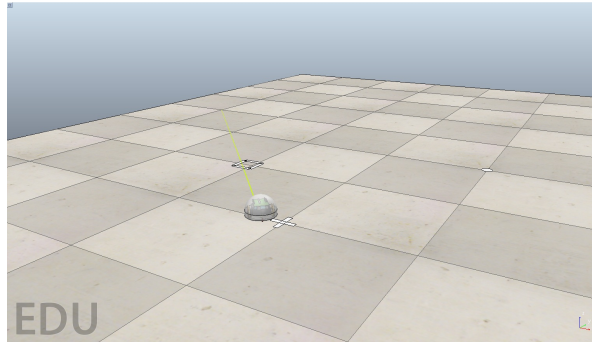


Figura 2.9: Trajeto descrito até atingir o alvo

2.2 Cenário 2

Construiremos uma sala de jantar conjugada com uma sala de estar e uma pequena cozinha. Para isto, considere a área de trabalho do CoppeliaSim com o Lumibot já posto, como ilustra a Figura 2.10

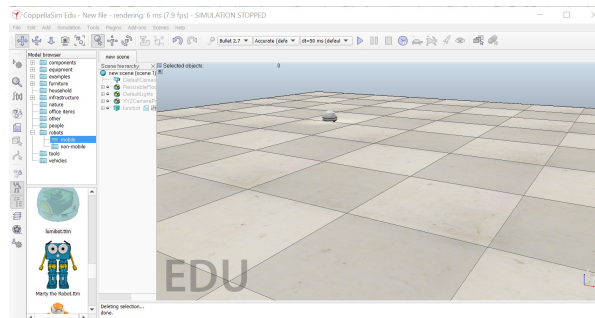


Figura 2.10: Lumibot no ambiente

Diante disso, seguindo os passo indicados através da Figura 2.11, colocaremos a mesa de jantar.

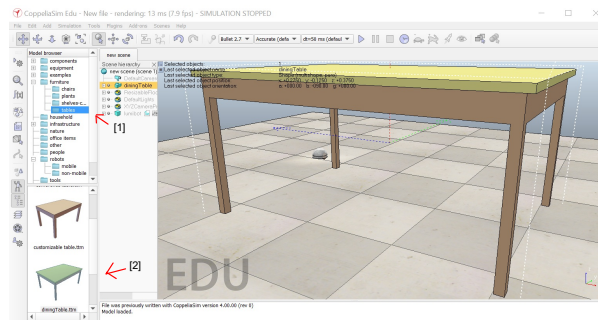


Figura 2.11: Incluindo a mesa de jantar

Os passos para colocação da mesa são os seguintes:

- [1] dentro de furniture,selecione a opção tables;
- [2] em seguida escolhemos a mesa desejada e arrastamos para a área de trabalho.

Com a mesa escolhida, podemos então escolher as cadeiras para compor a mesa de jantar, para isto seguindo os passo indicados através da Figura 2.12, colocaremos as cadeiras.

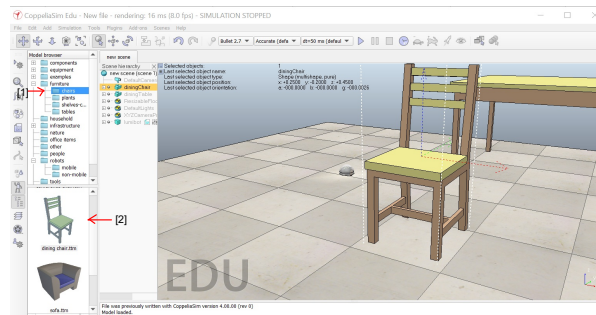


Figura 2.12: Incluindo as cadeiras

Os passos para colocação das cadeiras são os seguintes:

- [1] dentro de furniture,selecione a opção chairs;
- [2] em seguida escolhemos a cadeira desejada e arrastamos para a área de trabalho.

Dessa forma, obtemos toda a sala de jantar e construiremos em seguida a sala de estar. Inicialmente, colocaremos as cadeiras seguindo o mesmo procedimento feito anteriormente, onde escolheremos a cadeira de cor azul, como indica a seta ilustrada na Figura 2.13.

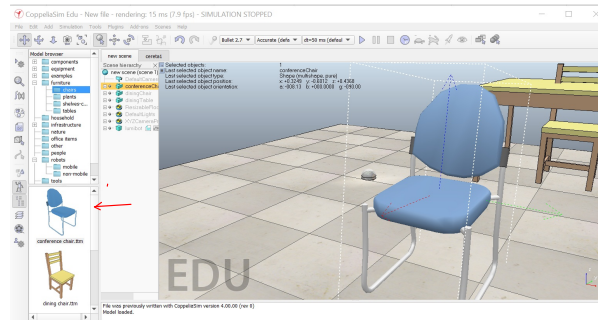


Figura 2.13: Cadeiras para sala de estar

Em seguida acrescentaremos um tapete para a sala de estar, para isto seguindo os passos indicados através da Figura 2.14, colocaremos o tapete.

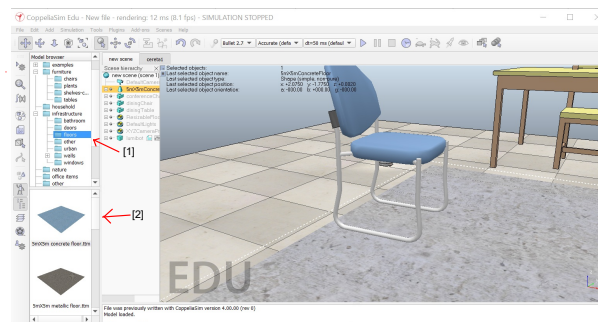
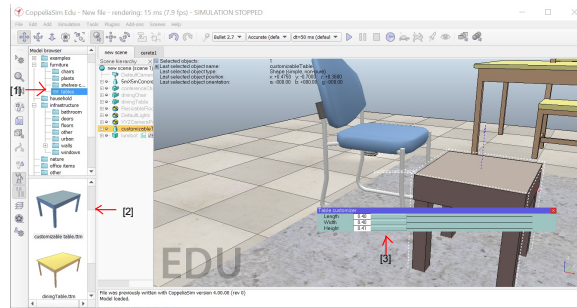


Figura 2.14: Tapete para sala de estar

Os passos para colocação do tapete são os seguintes:

- [1] dentro de infrastructure, selecione a opção floors;
- [2] em seguida escolhemos o piso desejada que representará o tapete e arrastamos para a área de trabalho.

Agora, adicionaremos a mesa de centro, para isto seguindo os passos indicados através da Figura 2.15, colocaremos a mesa de centro.



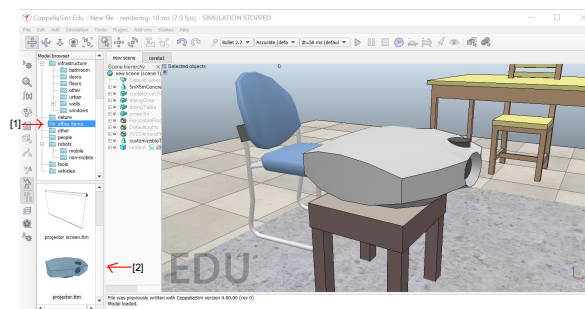
[3]

Figura 2.15: Mesa de centro para a sala de estar

Os passos para colocação da mesa de centro são os seguintes:

- [1] dentro de furniture,selecione a opção tables;
- [2] em seguida escolhemos a mesa desejada e arrastamos para a área de trabalho;
- [3] nesta caixa podemos redimensionar a mesa.

Em seguida colocaremos o projetor de imagem sobre a mesa de centro, para isto seguindo os passo indicados através da Figura 2.16, colocaremos o projetor de imagem na área de trabalho.



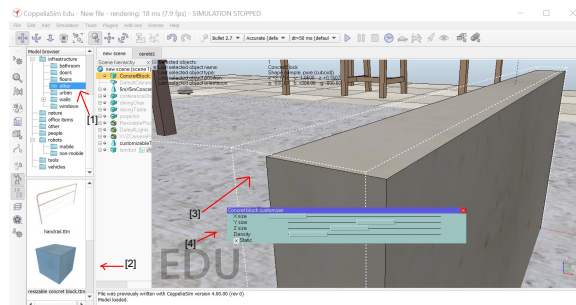
[3]

Figura 2.16: Projetor de imagem

Os passos para colocação do projetor são os seguintes:

- [1] selecione a opção office items;
- [2] em seguida escolhemos o equipamento desejada e arrastamos para a área de trabalho.

O próximo objeto que colocaremos na sala de estar é uma parede de frente para o projetor de imagem, para que possamos projetar a imagem. Para isto seguindo os passo indicados através da Figura 2.17, colocaremos a parede.



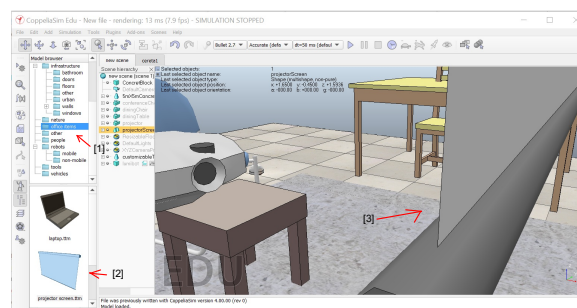
[3]

Figura 2.17: Parede ao fundo da sala

Os passos para colocação da parede são os seguintes:

- [1] dentro de infrastructure, selecione a opção other;
- [2] em seguida escolhemos o bloco de concreto e arrastamos para a área de trabalho;
- [3] temos o bloco de concreto sobre a sala de estar;
- [4] redimensionamos o bloco de concreto para adequação com a parede da sala de estar.

Após a colocação da parede na sala de estar, colocaremos a tela de projeção retrátil para que a imagem do projetor seja criada. Para isto seguindo os passo indicados através da Figura 2.18, colocaremos a tela de projeção.



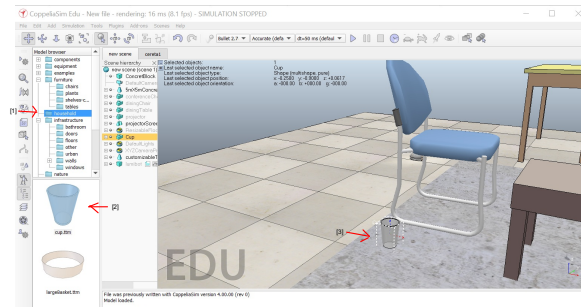
[3]

Figura 2.18: Tela para projeção de imagem

Os passos para colocação da tela de projeção retrátil são os seguintes:

- [1] selecione a opção officeitems;
- [2] em seguida escolhemos o projetor de imagem retrátil e arrastamos para a área de trabalho;
- [3] temos o projetor de imagem sobre a área de trabalho.

Adicionaremos os copos de vidro próximo as cadeiras, para isto seguindo os passo indicados através da Figura 2.19, colocaremos os copos sobre o piso.



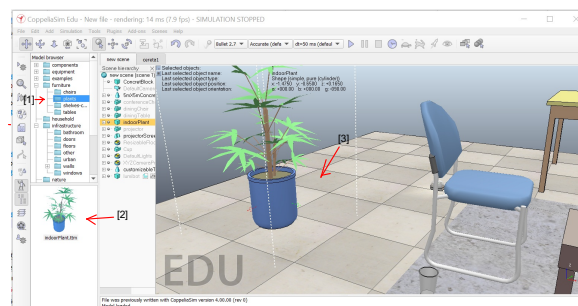
[3]

Figura 2.19: Copos de vidro

Os passos para colocação dos copos são os seguintes:

- [1] selecione a opção household;
- [2] em seguida escolhemos o copo desejado e arrastamos para a área de trabalho;
- [3] temos o copo próximo a cadeira na sala de estar.

Para finalizar a sala de estar adicionaremos a planta. Para isto seguindo os passos indicados através da Figura 2.12, colocamos uma planta.



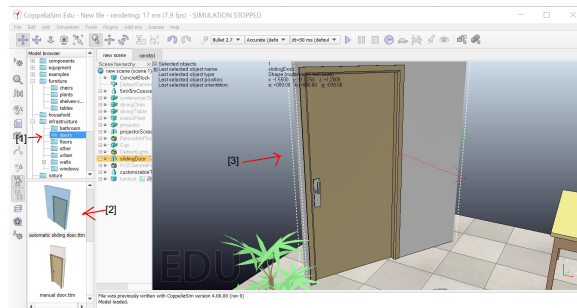
[3]

Figura 2.20: Planta

Os passos para colocação da planta são os seguintes:

- [1] dentro de furniture, selecione a opção plants;
- [2] em seguida escolhemos o jarro com a planta desejada e arrastamos para a área de trabalho;
- [3] temos a planta na sala de estar.

Para finalizar o cenário, montaremos a cozinha iniciando pela parede de frente para a sala de estar, esta parede é formada por uma porta de abertura e fechamento automático. Para adicionar a parede com a porta seguimos os passos indicados através da Figura 2.21, colocaremos as cadeiras.



[9]

Figura 2.21: Porta automática

Os passos para colocação da porta automática são os seguintes:

- [1] dentro de infrastructure, selecione a opção doors;
- [2] em seguida escolhemos a porta desejada e arrastamo-na para a área de trabalho;
- [3] obtemos a porta de abertura e fechamento automático na área de trabalho.

Finalizamos a cozinha acrescentando uma janela que desempenhará o papel de uma bancada entre a cozinha e a sala de jantar. Para adicionar a bancada seguimos os passos indicados através da Figura 2.22, colocaremos as cadeiras.

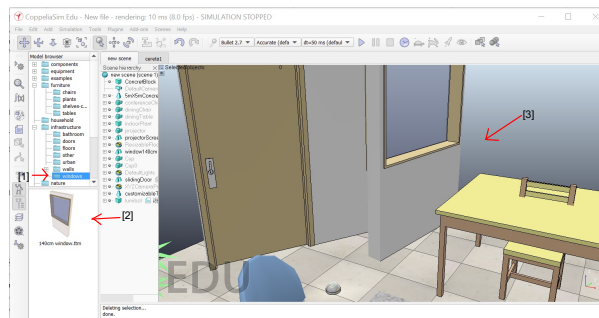
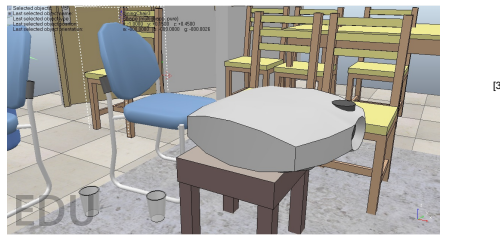


Figura 2.22: Bancada

Os passos para colocação da bancada são os seguintes:

- [1] dentro de infrastructure, selecione a opção windows;
- [2] em seguida escolhemos a única janela que temos e arrastamo-na para a área de trabalho;
- [3] obtemos a janela que funcionará como uma bancada na área de trabalho.

Assim, obtemos todos os objetos necessários para compor o cenário, porém estes objetos estão desorganizados, como ilustra a Figura 2.23



[3]

Figura 2.23: Salas de jantar e estar conjugadas desorganizada

A visualização panorâmica é ilustrada na Figura 2.24



[3]

Figura 2.24: Organização das salas de jantar e estar conjugadas psnorâmica

Diante disso, organizaremos o cenário seguindo os passos ilustrado na Figura 2.25

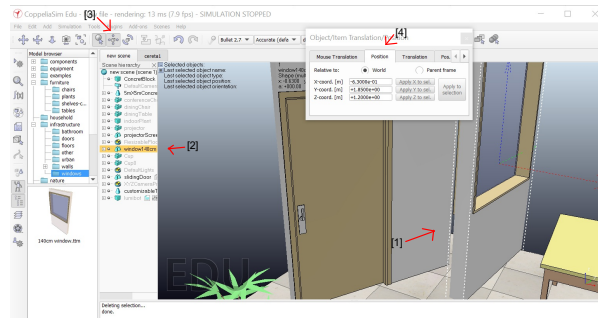


Figura 2.25: Organização das salas de jantar e estar

que são descritos como segue:

- [1] selecione o objeto clicando diretamente sobre o objeto;
- [2] selecione o objeto clicando diretamente sobre o nome do objeto, esta etapa é opcional e devendo ser usada quando não for possível visualizar o objeto;
- [3] clicando sobre este ícone podemos modificar a posição dos objetos sobre o plano;
- [4] através desta caixa podemos modificar a posição do objeto, após a etapa anterior.

Faremos este procedimento para todos os objetos que julgarmos necessário. Em seguida mudaremos a orientação dos objetos selecionado o objeto e seguindo os passos ilustrado na Figura 2.26

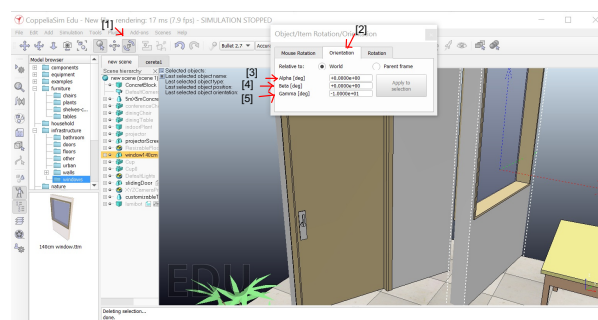


Figura 2.26: Mudando a orientação das salas de jantar e estar

que são descritos da seguinte maneira:

- [1] clicando sobre ícone abriremos a caixa para alteração da orientação do objeto;
- [2] através desta caixa que podemos modificar a orientação do objeto;

[3] rotação em torno do eixo x;

[4] rotação em torno do eixo y;

[5] rotação em torno do eixo z.

Feito todos os procedimentos descritos acima, obtemos o cenário bem organizado, como ilustra a Figura 2.27



Figura 2.27: Salas de jantar e estar conjugadas

que visualizamos panorâmico, como ilustra a Figura 2.28.

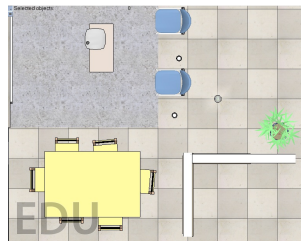


Figura 2.28: Salas de jantar e estar conjugadas vistas de cima

e ainda pode-se ter uma visão organizada, como ilustra a Figura 2.28



Figura 2.29: Visão panorâmico

Para o cenário ficar completo está faltando apenas o alvo que será representado por um disco branco, que pode ser adicionado de modo fácil sobre o plano. Feito todos os procedimentos, obtemos o cenário como ilustra a Figura 2.30.



Figura 2.30: Salas de jantar e estar conjugadas

Ou seja, obtemos uma sala de estar composta por: uma planta, duas poltronas de cor azul, com dois copos de vidros bem próximo das poltronas, uma mesa de centro sobre um tapete; em cima da mesa de centro temos um projetor de imagem que está projetando em uma tela de projeção retrátil sobre a parede ao fundo da sala. A sala de jantar é composta por uma mesa e seis cadeiras. Por sua vez a cozinha tem uma bancada para a sala de jantar e uma porta de abertura e fechamento automático. Todo este cenário é ilustrado de modo panorâmico na Figura 2.31.

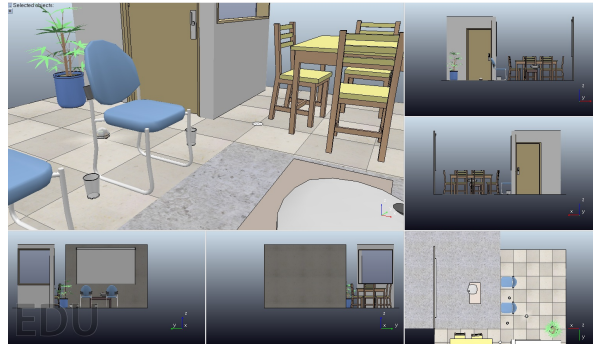


Figura 2.31: Vista panorâmica das salas com a cozinha

Queremos que o robô Lumibot situado na sala de estar bem atrás da poltrona, atinja o alvo dado por um disco branco situado na sala de jantar, bem ao lado da mesa, como ilustra a Figura 2.30. Sabendo que cada quadrado que compõe o piso tem lados medindo 0.5 metros, obtemos que o Lumibot estar situado na posição $(-1, 0)$ e o alvo está situado na posição $(0, 1)$. Queremos encontrar a trajetória que será seguida pelo robô para sair de sua posição inicial e chegar até o seu alvo. Utilizaremos a trajetória descrita por uma reta, para isto precisamos encontrar a função polinomial do primeiro grau cuja o seu gráfico é a reta desejada.

Através da expressão (2.1) e utilizando as coordenadas do robô e do alvo, obtemos a função polinomial que representa o trajeto que será seguido pelo Lumibot. Esta função é dada da seguinte maneira:

$$f(x) = x + 1$$

Ou seja, seguindo as notações do código obtemos a função

$$fxp = xp + 1.$$

Substituiremos os valores diretamente no código ilustrado nas Figuras 3.2 – 3.3. Iniciamos com o valor da constante $ap = 1$ que será substituído na linha 47, depois o valor da constante $bp = 1$ que será substituído na linha 48, em seguida o valor inicial de x, $xi = -1$, será substituído na linha 49, e por fim o valor final de x, $xf = 0$, que será substituído na linha 50. Segue-se, que o Lumibot percorrerá o trajeto passando pelo quadrado até atingir o obstáculo, com ilustra a Figura 2.32. Todo o percurso pode ser acompanhado através do link <https://youtu.be/O-y9RrBAMW0>



Figura 2.32: Trajeto descrito até atingir o alvo

2.3 Cenário 3

Considere o cenário anterior com duas modificações: o robô Lumibot está situado bem próximo a poltrona na posição $(-0.5, 0)$ e o alvo está escondido dentro da cozinha ocupando a posição $(-1, 1.5)$, como ilustra Figura 2.33



Figura 2.33: O Lumibot próximo a poltrona e o alvo dentro da cozinha

Todos os cômodos podem ser visualizados de modo panorâmico, incluindo o alvo dentro da cozinha, como ilustra a Figura 2.34



Figura 2.34: Visão panorâmica

Queremos encontrar a trajetória que será seguida pelo robô para sair de sua posição inicial e chegar até o seu alvo, dentro da cozinha, passando pela porta automática, como ilustra a Figura 2.35, então podemos utilizar a reta para determinar a trajetória.



Figura 2.35: Alvo atrás da porta automática

Precisamos modificar a velocidade de fechamento da porta automática, para evitar que a mesma colida com o robô. Para isto, inicialmente clicamos duas vezes sobre o ícone branco indicado com a seta em vermelho, como ilustra a Figura 2.36.

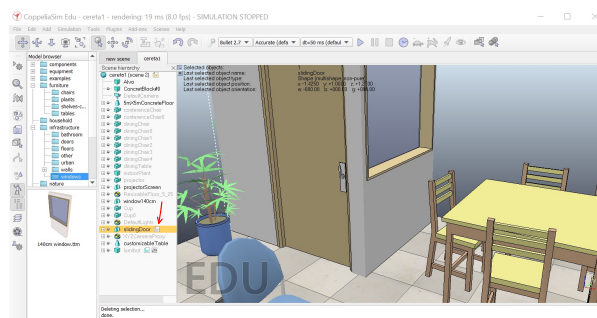


Figura 2.36: Modificação do código da porta

a caixa com o código em linguagem Lua será aberto, como ilustra a Figura 2.37.

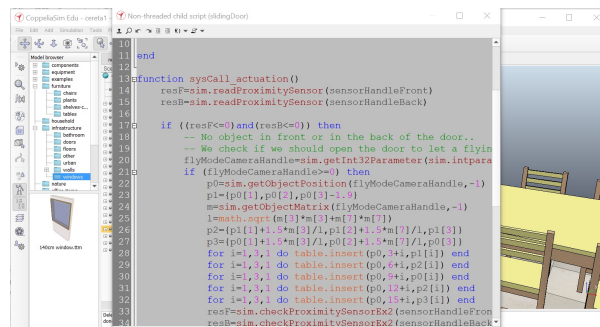


Figura 2.37: Código da porta automática

Agora basta a alteração da velocidade de fechamento; veja linha 41, em destaque ilustrada na Figura 2.38,

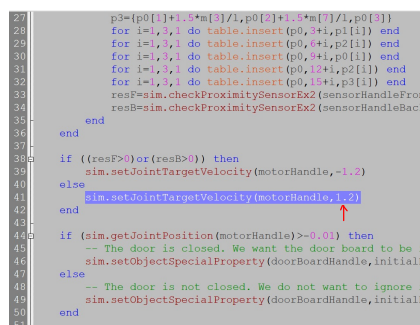


Figura 2.38: Alterando a velocidade da porta automática

Tal alteração consiste em trocar 1.2 por 0.1 e em seguida basta fechar a caixa com código que o processo está concretizado.

Através da expressão (3.1) e utilizando as coordenadas do robô e do alvo, obtemos a função polinomial que representa o trajeto que será seguido pelo Lumibot. Esta função será dada da seguinte maneira:

$$f(x) = -3x - \frac{3}{2}$$

Ou seja, seguindo as notações do código obtemos a função dada por

$$fxp = -3xp - \frac{3}{2}.$$

Substituiremos os valores diretamente no código ilustrado nas Figuras 3.2 – 3.3. Iniciamos com o valor da constante $ap = -3$ que será substituído na linha 47, depois o valor da constante $bp = -\frac{3}{2}$ que será substituído na linha 48, em seguida o valor inicial de

x , $x_i = -0.5$, será substituído na linha 49, e por fim o valor final de x , $x_f = -1$, que será substituído na linha 50. Segue-se que o Lumibot percorrerá o trajeto passando pelo quadrado até atingir o obstáculo, com ilustra a Figura 2.39. Todo o percurso pode ser acompanhado através do link <https://youtu.be/yN0m32HKfhY>



Figura 2.39: Trajetória descrita pelo robô até atingir o alvo

2.4 Cenário 4

Criaremos o cenário de uma sala de estar, para isso, considere o robô Lumibot já dentro da área de trabalho, como ilustra a Figura 2.40.

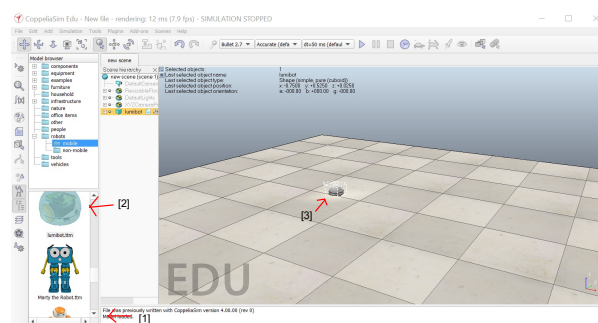


Figura 2.40: Lumibot dentro do ambiente

Partindo deste fato, iniciamos inserindo uma planta no ambiente, seguindo os passos ilustrado na Figura 2.41.

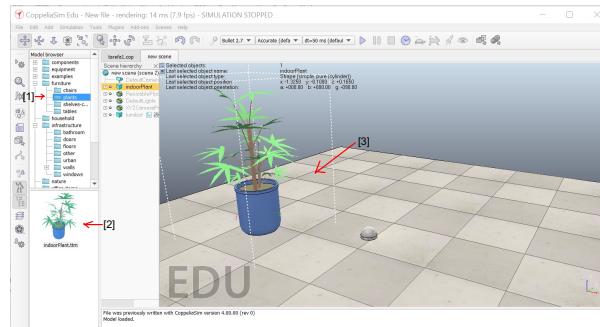


Figura 2.41: Inserindo a planta

Os passos para a inserção da planta são os seguintes:

- [1] dentro do ambiente furniture, escolha a opção plants;
- [2] escolha a única planta que temos como opção e arraste para a área de trabalho;
- [3] temos a planta dentro do ambiente.

O próximo objeto inserido no ambiente será uma poltrona, inserindo a poltrona no ambiente, seguindo os passos ilustrado na Figura 2.42.

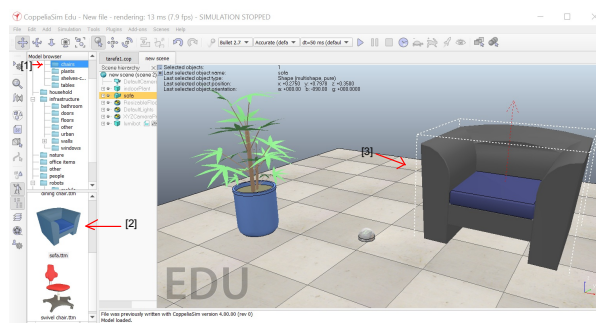


Figura 2.42: Poltrona

Mais precisamente, tais passos são os seguintes:

- [1] dentro do ambiente furniture, escolha a opção chairs;
- [2] escolha a poltrona indicada pela seta vermelha e arraste para a área de trabalho;
- [3] temos a poltrona dentro do ambiente.

Em seguida, colocaremos um bloco de concreto que representará uma parede dentro do ambiente, seguindo os passos ilustrado na Figura 2.43.

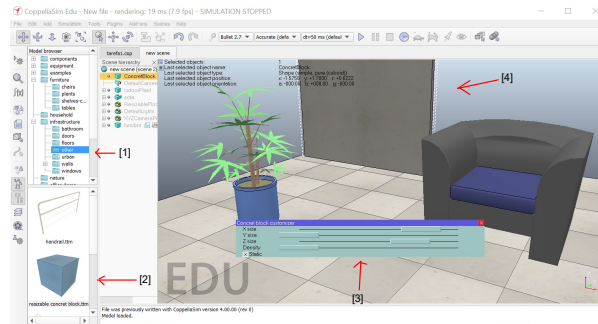


Figura 2.43: Parede

Os passos para a inserção de uma parede são os seguintes:

- [1] dentro do ambiente infrastructure, escolha a opção others;
- [2] escolha o bloco de concreto indicado através da seta em vermelho e arraste para a área de trabalho;
- [3] redimensione o bloco de concreto através da caixa indicada com a seta em vermelho;
- [4] temos uma parede dentro do ambiente.

Continuamos com a construção colocando uma janela dentro do ambiente, seguindo os passos ilustrado na Figura 2.44,

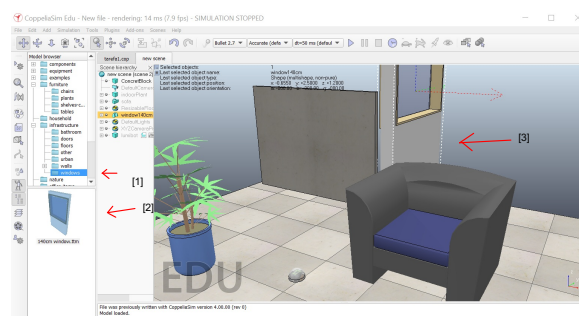


Figura 2.44: Janela

Tais passos são os seguintes:

- [1] dentro do ambiente infrastructure, escolha a opção windows;
- [2] escolha a única janela que temos como opção e arraste para a área de trabalho;
- [3] temos a janela dentro do ambiente.

Inserimos uma mesa de centro dentro do ambiente, seguindo os passos ilustrado na Figura 2.45.

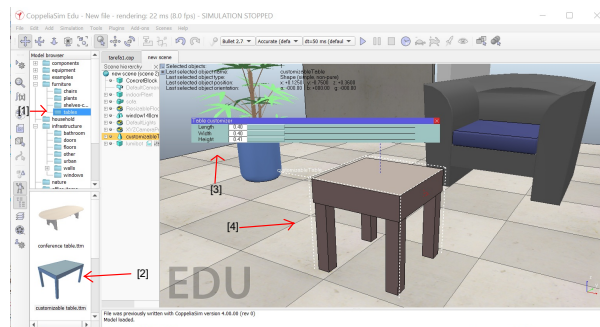


Figura 2.45: Mesa de centro

Os passos para inserção da mesa de centro são os seguintes:

- [1] dentro do ambiente furniture, escolha a opção tables;
- [2] escolha a mesa indicada com a seta em vermelho e arraste para a área de trabalho;
- [3] altere as dimensões da mesa através desta caixa;
- [4] temos a mesa de centro dentro do ambiente.

Incluiremos um computador dentro do ambiente, seguindo os passos ilustrado na Figura 2.46.

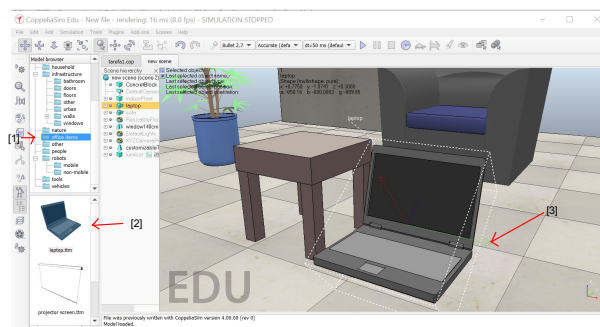


Figura 2.46: Computador

Os passos para incluir um computador no ambiente são os seguintes:

- [1] escolha a opção office items;
- [2] escolha o computador indicado com a seta e arraste para a área de trabalho;

[3] temos o computador dentro do ambiente.

Adicionaremos um pequeno pilar que representará uma jarra de vidro, seguindo os passos ilustrado na Figura 2.47.

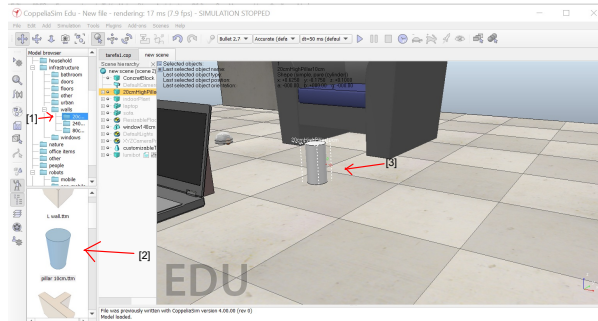


Figura 2.47: Jarra

Os passos para a inserção da jarra são os seguintes:

- [1] dentro do ambiente infrastructure, escolha a opção walls em seguida escolha 240 centímetros;
- [2] escolha o pilar indicado através da seta em vermelho e arraste para a área de trabalho;
- [3] temos uma jarra de vidro dentro do ambiente.

Também adicionaremos um copo de vidro dentro do ambiente, seguindo os passos ilustrado na Figura 2.48.

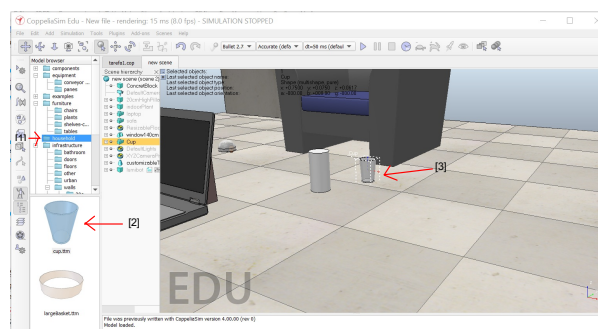


Figura 2.48: Copo de vidro

Os passos para adicionar o copo de vidro são os seguintes:

- [1] escolha a opção household;
- [2] escolha o copo de vidro indicado através da seta em vermelho e arraste para a área de trabalho;
- [3] temos o copo de vidro dentro do ambiente.

Por último inserimos a caixa cúbica dentro do ambiente. Para isso, basta clicar com o botão direito do mouse sobre a área de trabalho, em seguida escolher add, depois prime shape e por fim cuboide. Aparecerá uma caixa como indicada através da seta [1], ilustrada na figura 2.49.

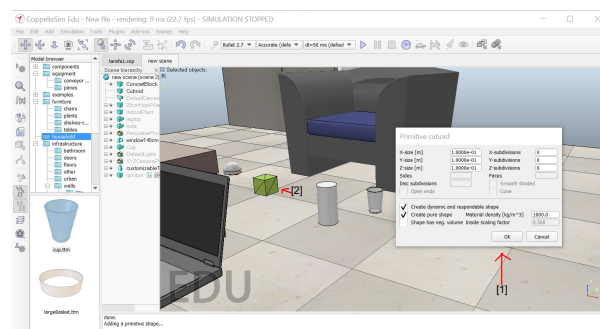


Figura 2.49: Caixa cúbica

Ao clicar em ok, obtemos a caixa cúbica, como ilustra a Figura 2.49. Note que a caixa cúbica não aparece com a cor verde como ilustrada na Figura 2.49. Para mudar a cor da caixa cúbica, basta clicar com o botão direito do mouse sobre a caixa cúbica e escolher a opção edit e depois a opção morph selection into convex shapes.

Assim, adicionando o alvo, representado por um disco branco, dentro da área de trabalho obtemos todos os objetos que compõem o cenário, como ilustra a Figura 2.50.



Figura 2.50: Objetos dentro do ambiente

Já a Figura 2.51 temos o correspondente modo panorâmico.

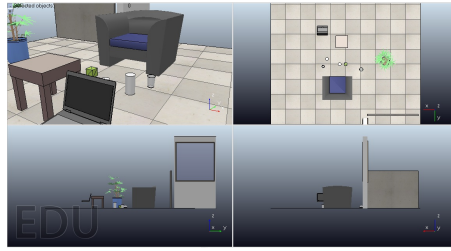


Figura 2.51: Ambiente panorâmico

Porém, os objetos estão todos desorganizados e precisamos organizar todos os objetos dentro do ambiente, para isso, após selecionar o objeto que será modificado, seguimos os passos ilustrados na Figura 2.52,

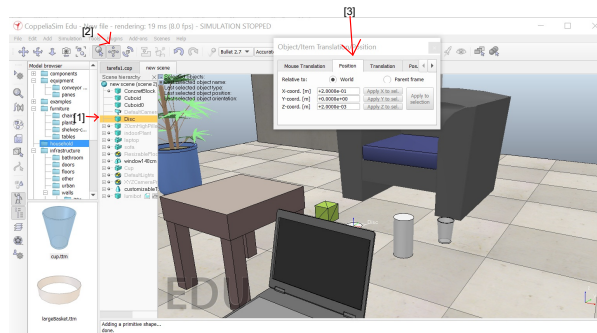


Figura 2.52: Organização do ambiente

que são descritos da seguinte maneira:

- [1] troque o nome do disco para Alvo;
- [2] clicando sobre este ícone abriremos uma caixa que nos permite deslocar os objetos ao longo do ambiente;
- [3] através desta caixa podemos alterar a posição dos objetos dentro do ambiente.

Além disso, fazendo rotações de alguns objetos deixamos o ambiente ainda mais bem organizado. Para fazer estas rotações necessárias, após selecionar o objetos que queremos modificar sua orientação, seguimos os passos ilustrado na Figura 2.53

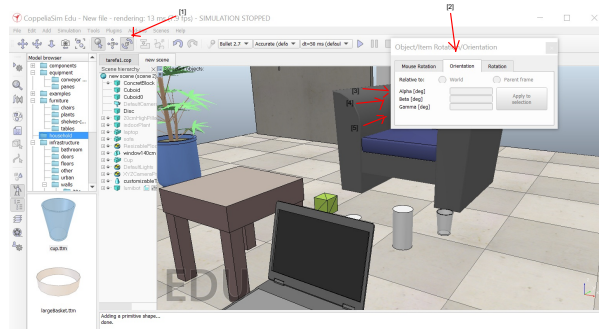


Figura 2.53: Orientando os objetos

que são descritos da seguinte maneira:

- [1] clicando sobre este ícone abriremos a caixa que nos permite alterar a orientação do objetos que foi selecionado;
- [2] dentro desta caixa podemos alterar a orientação de cada objeto;
- [3] rotação em torno do eixo x;
- [4] rotação em torno do eixo y;
- [5] rotação em torno do eixo z.

Assim, obtemos o cenário de uma sala de estar, composta por uma poltrona, uma mesa de centro, um notebook, um copo de vidro, uma jarra de vidro, uma planta, uma caixa verde e o Lumibot, como ilustra a Figura 2.54,



Figura 2.54: Sala de estar

que ainda podemos visualizar de modo panorâmico, como ilustra a Figura 2.55

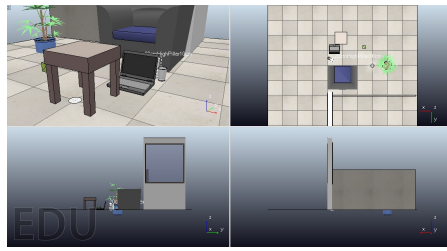


Figura 2.55: Sala de estar panorâmica

A origem do plano que forma o piso, é exatamente o centro de massa da poltrona. Como cada quadrado que forma o piso tem lados medindo 0,5 metros, obtemos que a posição do robô no plano é dada pelo par $(-1, 0)$. Por sua vez, o alvo que o Lumibot deverá atingir o alvo localizado em baixo da mesa de centro, assim a posição do alvo no plano é dada pelo par ordenado $(0, -1)$. Levando em consideração que cada lado da caixa possui 0.01 metros de comprimento, e a mesa possui 0.04 de largura por 0.04 de comprimento e 0.04 de altura, segue-se que o gráfico da função polinomial $f(x) = -x - 1$ descreve a trajetória que o Lumibot irá seguir sem tocar a caixa verde e a mesa até atingir o alvo, como ilustra a Figura 2.56. Todo o percurso pode ser acompanhado através do link <https://youtu.be/IYgiAVM0oks>



Figura 2.56: Trajetória descrita pelo robô até atingir o alvo

Capítulo 3

Resultado para Função Polinomial do Segundo Grau

Nesta seção utilizaremos funções polinomiais do segundo grau para que possamos dar funcionalidade para o Lumibot apresentando uma maneira de resolver um problema do mundo real.

Função polinomial do segundo grau, também chamada de função quadrática, é um dos principais conteúdos ensinados no ensino médio. É de grande importância que os estudantes dominem completamente a função quadrática, isto significa entender suas características e possíveis aplicações.

O método de ensino, comumente, utilizado nas escolas de ensino médio consiste em explicar a definição de função quadrática, fazer exemplos e apresentar suas diferentes formas. A forma mais comum utilizada é dada da seguinte maneira:

$$f(x) = ax^2 + bx + c \quad a \neq 0. \quad (3.1)$$

Assim, partindo de (3.1) explora-se as raízes, o gráfico que é uma parábola, vértices com o valor mínimo e máximo, sinal, monotonicidade, e os casos em que a parábola tem concavidade para cima e para baixo.

Um dos requisitos mais importantes para os alunos é que eles entendam corretamente a conexão de dependência dos coeficientes a , b e c em (3.1). De certa forma, conhecendo estes coeficientes a função quadrática fica completamente determinada. Segue-se que conhecendo pelo menos três valores para x e $f(x)$ em (3.1), obtemos os coeficientes a , b e c , e conseqüentemente toda a função $f(x)$.

Dividiremos esta seção em quatro subseções que serão chamadas de cenários. Cada cenário ilustra uma real possibilidade, em que encontraremos uma função polinomial do segundo grau que será utilizada para determinar a trajetória do robô Lumibot dentro do ambiente CoppeliaSim.

3.1 Cenário 1

Inicialmente consideraremos um cenário bem simples criado através do CoppeliaSim, formado apenas por um piso de centro na origem, que coincide com o centro círculo branco. Além disso, sobre o piso temos um quadrado branco de lados medindo 0.01 metros e um símbolo em X de cor branca e o robô Lumibot, como ilustra a Figura 3.1. Partindo do círculo como referência, como cada lado dos quadrados que compõem o piso tem lados medindo 0.5 metros, obtemos que o robô Lumibot localiza-se na posição $(-2, 0.24)$. Por sua vez, o quadrado está localizado na posição $(-1, -0.75)$ e o alvo que o Lumibot deve atingir representado pelo símbolo X está localizada na posição $(0, 0.25)$, como ilustra a Figura 3.1. Todas estas posições são consideradas em relação ao centro de massa dos objetos e são dadas em metros.

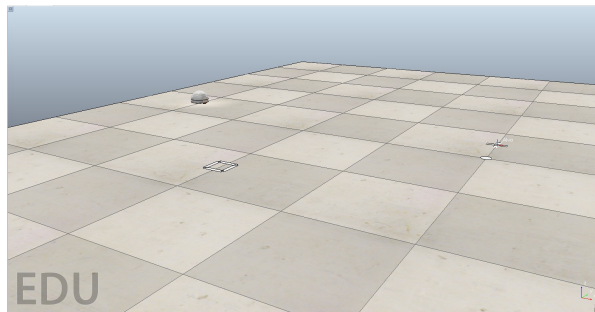


Figura 3.1: Plano centrado sobre o círculo

Queremos encontrar uma trajetória para que o Lumibot saia de sua posição inicial, passe pelo quadrado branco e chegue até o alvo. Considerando a expressão (3.1), através dos valores das posições dadas acima, obtemos que a trajetória pode ser dada através do gráfico da função:

$$f(x) = x^2 + 2x + 0.25.$$

Assim, através da função quadrática acima e seguindo as notações do código em Python, ilustrados nas Figuras 3.2 – 3.3, temos que

$$fxp = xp * xp + 2 * xp + 0.25$$

```

1 # coding=utf-8
2 # Insert in a script in Coppelia
3 # simRemoteApi.start(19999)
4 import sys
5 try:
6     import sim
7 except:
8     print('-----')
9     print("'sim.py' could not be imported. This means very probably that")
10    print("either 'sim.py' or the remoteApi library could not be found.")
11    print("Make sure both are in the same folder as this file.")
12    print("or appropriately adjust the file 'sim.py'")
13    print('-----')
14    print('')
15    import math
16    import time
17    print("Program started")
18    sim.simFinish(-1) # just in case, close all opened connections
19    clientID = sim.simStart(127.0.0.1, 19999, True, True, 5000, 5)
20    robotname = 'lumibot'
21    targetname = 'alvo'
22    if clientID != -1:
23        time.sleep(2)
24        # Colocar nomes
25        [erro, robot] = sim.simGetObjectHandle(clientID, robotname, sim.sim_opmode_oneshot_wait)
26        [erro, target] = sim.simGetObjectHandle(clientID, targetname, sim.sim_opmode_oneshot_wait)
27
28        [erro, robotLeftMotor] = sim.simGetObjectHandle(clientID, robotname + "_leftMotor", sim.sim_opmode_oneshot_wait)
29        [erro, robotRightMotor] = sim.simGetObjectHandle(clientID, robotname + "_rightMotor", sim.sim_opmode_oneshot_wait)
30        # Criar stream de dados
31        [erro, positionRobot] = sim.simGetObjectPosition(clientID, robot, -1, sim.sim_opmode_streaming)
32        [erro, positionTarget] = sim.simGetObjectPosition(clientID, target, -1, sim.sim_opmode_streaming)
33        [erro, orientationRobot] = sim.simGetObjectOrientation(clientID, robot, -1, sim.sim_opmode_streaming)
34        time.sleep(3)
35        # Comandos dos motores
36        sim.simSetJointTargetVelocity(clientID, robotRightMotor, 0, 0, sim.sim_opmode_oneshot)
37        sim.simSetJointTargetVelocity(clientID, robotLeftMotor, 0, 0, sim.sim_opmode_oneshot)
38        state = 'stopped'
39        dist_set = 0.1
40        vref = 2.0
41        running = True
42        while(running):
43            # Colocar dados robô e alvo
44            [erro, [x, y, z]] = sim.simGetObjectPosition(clientID, robot, -1, sim.sim_opmode_buffer)
45            [erro, [xt, yt, zt]] = sim.simGetObjectPosition(clientID, target, -1, sim.sim_opmode_buffer)
46            [erro, [alpha, beta, gamma]] = sim.simGetObjectOrientation(clientID, robot, -1, sim.sim_opmode_buffer)
47            dist = math.sqrt((x-xt)*(x-xt)+(y-yt)*(y-yt))
48            # Estado de dados de função
49            xp = x
50            yp = y + 0.5
51            bp = 2. #constante b

```

Figura 3.2: Código em Python

Continuando o código

```

52    cp = 1 #constante c
53    xi = 0 #valor inicial
54    xf = -1.07 #valor final
55    dr = 4.5 #distancia de equilíbrio
56    fdp = ap + xp + bp + cp
57    fdr = 2 * ap + xp + bp
58    gxp = 1/(math.sqrt(abs(fdp + fdr + yr + yr)))
59    hxp = 1 / (math.sqrt(abs(fdp + fdr + yr + yr)))
60    Gxp = (ap/abs(ap))*(-yp - fdp)
61    Hxp = ((xi-xf)/(abs(xi-xf)))*math.sqrt(abs(1 - Gxp + Gxp))
62    #Comandos
63    if (state == 'stopped') & (dist > dist_set):
64        state = 'align'
65    elif (state == 'align') & (gamma < (ap/abs(ap))*math.pi/2) :
66        state = 'forward'
67    elif (state == 'forward') & (dist < dist_set):
68        state = 'stopped'
69    #Ações
70    if state == 'stopped':
71        vRightMotor = 0.0
72        vLeftMotor = 0.0
73        running = False
74    elif state == 'align':
75        vRightMotor = (ap/abs(ap))*vref
76        vLeftMotor = (ap/abs(ap))*vref
77    elif state == 'forward':
78        vRightMotor = -(math.cos(gamma))*((xi-xf)/(abs(xi-xf)))+(1/(0.05))*((gxp+fxp+fdr+hxp+Gxp)+(0.18*math.sin(gamma)
79            -math.cos(gamma))*(gxp+Gxp-hxp+Hxp+fdr)+(0.18*math.cos(gamma))*((2*dr)*math.sin(gamma)))-
80            vLeftMotor = (ap/abs(ap))*((xi-xf)/(abs(xi-xf)))+(1/(0.05))*((gxp+fxp+fdr+hxp+Gxp)+(-0.18*math.sin(gamma)
81            -math.cos(gamma))*(gxp+Gxp-hxp+Hxp+fdr)+(math.sin(gamma)-(0.18*math.cos(gamma)))/(2*dr)))
82    # Comandos motores
83    sim.simPauseCommunication(clientID, True)
84    sim.simSetJointTargetVelocity(clientID, robotRightMotor, vRightMotor, sim.sim_opmode_oneshot)
85    sim.simSetJointTargetVelocity(clientID, robotLeftMotor, vLeftMotor, sim.sim_opmode_oneshot)
86    sim.simPauseCommunication(clientID, False)
87    sim.simPauseSimulation(clientID, sim.sim_opmode_oneshot_wait)
88    sim.simFinish(clientID)
89 else:
90    print ('falha de conexão com o remote API server')
91    print ('Programa finalizado')
92
93 #vRightMotor = -(1/(0.05+2*math.sqrt((gxp+fxp+fdr+hxp+Gxp)))+(2*fxp+fxp*(gxp+yp+yp)))+2*yp*math.sqrt(abs(1-(gxp
94 #vLeftMotor = -(1/(0.05+2*math.sqrt((gxp+fxp+fdr+hxp+Gxp)))+(2*fxp+fxp*(gxp+yp+yp)))+2*yp*math.sqrt(
95
96 #vRightMotor = -(1/(0.05*math.sqrt((gxp+fxp+fdr+hxp+Gxp)))+(gxp+fxp+fdr+hxp+Gxp)+math.sqrt(abs(1-(gxp+fxp+fdr+hxp+Gxp)))
97 #vLeftMotor = -(1/(0.05*math.sqrt((gxp+fxp+fdr+hxp+Gxp)))+(gxp+fxp+fdr+hxp+Gxp)+math.sqrt(abs(1-(gxp+fxp+fdr+hxp+Gxp)))

```

Figura 3.3: Continuação do Código

Substituiremos os valores diretamente no código ilustrado nas Figuras 3.2 – 3.3. Iniciamos com o valor da constante $ap = 1$ que será substituído na linha 50, depois o valor da constante $bp = 2$ que será substituído na linha 51, por sua vez o valor da constante 0.25 que será substituído na linha 52; em seguida o valor inicial de x , $xi = -2$, será substituído na linha 53, e por fim o valor final de x , $xf = 0$, que será substituído na linha 54 e o valor de controle de trajetória $dr = 4.5$ será substituído na linha 55. Segue-se, que o Lumibot percorrerá o trajeto descrito por uma parábola passando pelo quadrado até atingir o alvo, como ilustra a Figura 3.4. Todo a trajetória pode ser acompanhada na íntegra através do link <https://youtu.be/i1cv8HZYD5w>

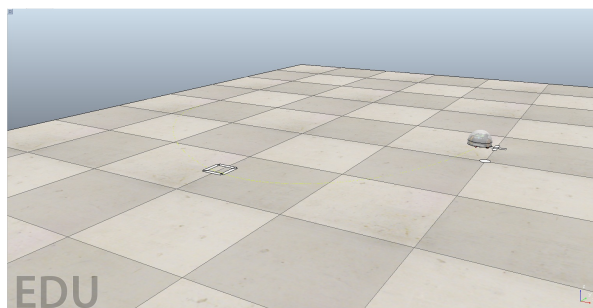


Figura 3.4: Trajeto descrito até atingir o alvo

O valor do ponto de controle pode ser escolhido entre 1 e 10, sendo que a escolha deste valor influencia diretamente no percurso do robô.

3.2 Cenário 2

Considere o cenário de uma sala de estar, composta por uma poltrona, uma mesa de centro, um notebook, um copo de vidro, uma jarra de vidro, uma planta, uma caixa verde (escondida pela mesa de centro) e o Lumibot, como ilustra a Figura 3.5



Figura 3.5: O Lumibot a direita da poltrona e alvo embaixo da mesa

Sabendo que o centro de massa da poltrona é exatamente a origem do plano, e que cada quadrado que compõe o piso tem lados medido 0.5 metros, obtemos que o Lumibot situa-se na posição $(-1, 0)$ sobre o plano e o alvo estar situado na posição $(0, -1)$. Queremos encontrar a trajetória que será seguida pelo robô para sair de sua posição inicial e chegar até o seu alvo, representado pelo círculo embaixo da mesa de centro. Como entre o robô e o alvo temos uma caixa, como ilustra a Figura 3.6, então não podemos utilizar uma reta como a trajetória.



Figura 3.6: Caixa verde como obstáculo

Uma alternativa para este problema, é dada através da expressão (3.1), pois

utilizando as coordenadas do robô e do alvo, obtemos a função quadrática cuja o gráfico representa o trajeto que será seguido pelo Lumibot, até atingir o seu alvo, dada da seguinte maneira:

$$f(x) = x^2 - 1$$

Ou seja, seguindo as notações do código obtemos a função quadrática

$$fxp = xp * xp - 1.$$

Substituiremos os valores diretamente no código ilustrado nas Figuras 3.2 – 3.3. Iniciamos com o valor da constante $ap = 1$ que será substituído na linha 50, depois o valor da constante $bp = 0$ que será substituído na linha 51, por sua vez o valor da constante -1 que será substituído na linha 52; em seguida o valor inicial de x , $xi = -1$, será substituído na linha 53, e por fim o valor final de x , $xf = 0$, que será substituído na linha 54 e o valor de controle de trajetória $dr = 4.5$ será substituído na linha 55. Segue-se, que o Lumibot percorrerá o trajeto descrito por uma parábola desviando do obstáculo até atingir o alvo, como ilustra a Figura 3.7. Toda a trajetória pode ser acompanhada na íntegra através do link <https://youtu.be/wvAcMiRbHSg>

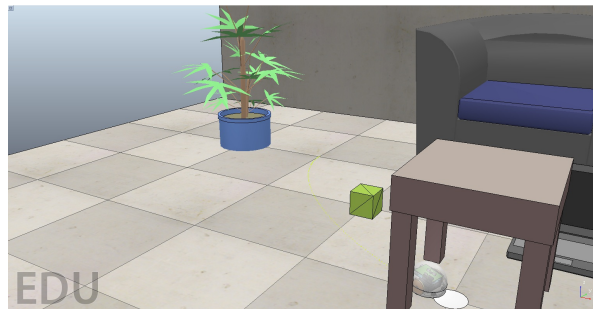


Figura 3.7: Trajetória desviando da caixa verde

3.3 Cenário 3

Considere o mesmo cenário ilustrado anteriormente com apenas uma mudança, o alvo que estava embaixo da mesa agora localiza-se ao lado do sofá, como ilustra a Figura 3.8.



Figura 3.8: Alvo á esquerda do sofá

Como cada quadrado que forma o piso tem lado medindo 0,5 metros, obtemos que a posição do alvo no plano é dada pelo par ordenado $(1, 0)$. Assim, o gráfico da mesma função quadrática $f(x) = x^2 - 1$ será a trajetória que o Lumibot irá seguir desviando da caixa verde até atingir o alvo, com valores inicial e final para x sendo, $x_i = -1$ e $x_f = 1$, respectivamente, como ilustra a Figura 3.9. Toda a trajetória pode ser acompanhada na íntegra através do link <https://youtu.be/ZCTPnD80Anw>



Figura 3.9: Trajetória descrita pelo robô até atingir o alvo

3.4 Cenário 4

Considere uma sala de jantar conjugada com uma sala de estar e uma pequena cozinha, como ilustra a Figura 3.10.



Figura 3.10: Salas de jantar e estar conjugadas

A sala de estar é composta por, uma planta, duas poltronas de cor azul, com dois copos de vidros bem próximo das poltronas, uma mesa de centro sobre um tapete; em cima da mesa de centro temos um projetor de imagem que está projetando em uma tela de projeção retrátil sobre a parede ao fundo da sala. A sala de jantar é composta por uma mesa e seis cadeiras. Por sua vez a cozinha tem uma bancada para a sala de jantar e uma porta de abertura e fechamento automático. Todo este cenário é ilustrado de modo panorâmico na Figura 3.11.



Figura 3.11: Vista panorâmica das salas com a cozinha

Queremos que o robô Lumibot situado na sala de jantar bem próximo a mesa, atinja o alvo dado por um círculo branco situado na cozinha, como ilustra a Figura 3.10. Sabendo que cada quadrado que compõe o piso tem lados medindo 0.5 metros, obtemos que o Lumibot estar situado na posição $(0, 1)$ e o alvo situa-se na posição $(-1.07, 1.15)$. Queremos encontrar a trajetória que será seguida pelo robô para sair de sua posição inicial e chegar até o seu alvo. Utilizaremos a trajetória dada por uma parábola. Para isto precisamos encontrar a função polinomial do segundo grau cujo seu gráfico seja o trajeto desejado.

Através da expressão (3.1) e utilizando as coordenadas do robô e do alvo, obtemos a função polinomial que representa o trajeto que será seguido pelo Lumibot. Esta função

será dada:

$$f(x) = 2x^2 + 2x + 1$$

Ou seja, seguindo as notações do código obtemos a função

$$f_{xp} = 2 * xp * xp + 2 * xp + 1.$$

Como através da função quadrática encontramos os valores de ap , bp e cp , então o valor $ap = 2$ será substituído na linha 50, $bp = 2$ que será substituído na linha 51 e $cp = 1$ que será substituído na linha 52. Em seguida atribuímos o valor inicial de x , $xi = 0$ na linha 53 e o valor final $xf = -1.07$ na linha 54. Finalmente, adotando o valor de controle $kp = 4.5$, e atribuindo este valor na linha 55 do código ilustrado nas Figuras 3.2 – 3.3; obtemos que o Lumibot percorre o trajeto passando pelo quadrado até atingir o alvo, como ilustra a Figura 3.11. Toda a trajetória pode ser acompanhada na íntegra através do link <https://youtu.be/hD7qcXIcCcc>



Figura 3.12: Trajeto descrito até atingir o alvo

Capítulo 4

Conclusão

Neste trabalho, através do ambiente de simulação CoppeliaSim conseguimos uma interdisciplinaridade envolvendo o conteúdo matemático de funções polinomiais do primeiro e segundo grau. Mais precisamente, através do gráfico dessas funções polinomiais, conseguimos apresentar uma funcionalidade para o robô Lumibot. Desta forma, obtemos uma importante ferramenta que permite despertar a criatividade matemática dos alunos da educação básica, ou do público em geral, para atuarem com as novas profissões que a indústria 4.0 exige. Tudo isso está em comum acordo com o modo de aprendizagem STEM.

Observamos aqui que a trajetória descrita pelo Lumibot não corresponde com exatidão ao gráfico das funções polinomiais que estamos considerando. Entretanto, todas estas trajetórias podem ser melhoradas. Tais melhorias estariam além do foco de um primeiro trabalho como esse. Queremos ressaltar que o presente trabalho propõe todo um ambiente que desperta a criatividade matemática que é cada mais necessária para novas profissões.

Referências Bibliográficas

- [1] Bybee, R. W. (2013). The Case for STEM Education: Challenges and Opportunities. Washington DC: National STEM Teachers Association. Disponível em: <https://www.nsta.org/resources/case-stem-education-challenges-and-opportunities> Acessado em: 15 jul. 2022.
- [2] Chen, C. S., Lin, J. W. (2019). A Practical Action Research Study of the Impact of Maker-Centered STEM-PjBL on a Rural Middle School in Taiwan. International Journal of Science and Mathematics Education, 17(1), 85–108. Disponível em: <https://eric.ed.gov/?id=EJ1220538> Acessado em: 20 mar. 2022.
- [3] De Aguiar, G. B. P. e Souza, C. A. (2022).(Livro no PRELO) PRAGMATISMO MATEMÁTICO: matrizes, editora EDUFPI, Teresina-Pi, 188, ISBN: 978-65-5904-202-9.
- [4] Gonzalez, H. B. and Kuenzi, J. (2012). Science, technology, engineering, and mathematics (STEM):A Primer. Congressional Research Service, August, 1–15. Disponível em: <http://www.stemedcoalition.org/wp-content/uploads/2010/05/STEM-Education-Primer.pdf>. Acessado em: 06 fev. 2022.
- [5] Gleason, N. W. (2018). Higher Education in the Era of the Fourth Industrial Revolution. Palgrave Macmillan. Singapore.
- [6] Hallinen, J. STEM. Disponível em: <https://www.britannica.com/topic/STEMeducation>.
- [7] Hinojo-Lucena, F. J., Dúo-Terrón, P., Navas-Parejo, M. R., Rodríguez-Jiménez, C., Moreno-Guerrero, A. J. (2020). Scientific performance and mapping of the term STEM in education on the web of science. Sustainability (Switzerland), 12(6), 1–20. Disponível em: <https://www.mdpi.com/665100>. Acessado em: 12 mai. 2022.
- [8] Li, M, Ren, F., (2019). Application of industrial robot in automatic forging production line [J]. Forging and stamping, (17): 64-66.

- [9] Lasi, H., Fettke, P., Kemper, H. G., Feld, T., & Hoffmann, M. (2014). Industry 4.0. *Business & information systems engineering*, 6(4), 239-242. Disponível em: <https://core.ac.uk/download/pdf/301363122.pdf>
Acessado em: 08 mai. 2022.
- [10] Naya, M. et al. (2017). A versatile robotic platform for educational interaction. In *Proceedings of the 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Bucharest, Romania, 21-23; Volume 1, pp. 138-144. Disponível em: http://idaacs.net/storage/conferences/2/abstracts/i17-052-camera_ready.pdf
Acessado em: 18 abr. 2022.
- [11] Oliveira, M. F. e Souza, C. A. (2020). A Circunferência de Centro na Origem como Produto de Matrizes. *Professor de Matemática online-PMO*, volume 8, 535-550. Disponível em:
https://scholar.archive.org/work/5nizl5is5zhznljm5zgmmys7oi/access/wayback/http://pmo.sbm.org/content/uploads/sites/16/dlm_uploads/2020/10/art39_ol8PMO_SBM_2020b.pdf
Acessado em: 20 jul. 2022.
- [12] Pratama, I., Permanasari, A. E., Ardiyanto, I., & Indrayani, R. (2016). A review of missing values handling methods on time-series data. In *2016 international conference on information technology systems and innovation (ICITSI)*, 1-6.
- [13] Park, W., Wu, J. Y., & Erduran, S. (2020). The Nature of STEM Disciplines in the Science Education Standards Documents from the USA, Korea and Taiwan. *Science & Education*, 29, 899-927. Disponível em: <https://link.springer.com/article/10.1007/s11191-020-00139-1>
Acessado em: 02 mar. 2022.
- [14] Rosa, M., Orey, D. C. (2018). STEM Education in the Brazilian Context: An Ethnomathematical Perspective. In R. Jorgensen & K. Larkin (Eds.), *STEM Education in the Junior Secondary: The State of Play* (pp. 221-247). Springer Singapore. Disponível em: https://link.springer.com/chapter/10.1007/978-981-10-5448-8_11
Acessado em: 14 mar. 2022.
- [15] Struyf, A., Loof, H. De, Pauw, J. B., & Petegem, P. Van. (2019). Students' engagement in different STEM learning environments : integrated STEM education as promising practice ? *International Journal of Science Education*, 41(10), 1387-1407. Disponível em: <https://www.tandfonline.com/doi/abs/10.1080/09500693.2019.1607983>
Acessado em: 14 mar. 2022
- [16] Shaughnessy, J. M. (2013). Mathematics in a STEM context. *Mathematics Teaching in the Middle school*, 18(6), 324-324. Disponível em:

<https://www.jstor.org/stable/10.5951/mathteacmidscho.18.6.0324>

Acessado em: 16 mar. 2022.

- [17] Takacs, A. G. et al. (2016). Teachers kit: development, usability and communities of modular robotic kits for classroom education. *IEEE Robotics & Automation Magazine* 23(2): 30-39.
- [18] Tomperi, P., Kvivesen, M., Manshadi, S., Uteng, S., Shestova, Y., Lyash, O., ... & Lyash, A. (2022). Investigation of STEM Subject and Career Aspirations of Lower Secondary School Students in the North Calotte Region of Finland, Norway, and Russia. *Education Sciences*, 12(3), 192. Disponível em: <https://www.mdpi.com/2227-7102/12/3/192>
Acessado em: 06 jan. 2023.
- [19] Vaz, J. M.; Fabro, J. A.,(1999). Snnap-sistema neural de navegação em ambientes pré-mapeados. In: *Proceedings of the IV Brazilian Conference on Neural Networks-IV Congresso Brasileiro de Redes Neurais*. p. 118-123.
- [20] Yildirim, B. and Selvi, M. (2016) J. Examination of the effects of STEM education integrated as a part of science, technology, society and environment courses. *Journal of Human Science*, 13(2), 684-3695. Disponível em:
<https://avesis.gazi.edu.tr/yayin/f3a35b99-cc1d-4f50-b75c-ba09f4501b5b/examination-of-the-effects-of-stem-education-integratedas-a-part-of-science-technology-society-and-environmentcourses/document.pdf>
Acessado em: 18 jul. 2022.
- [21] Yata, C., Ohtani, T., Isobe, M., (2020). Conceptual framework of STEM based on Japanese subject principles. *International Journal of STEM Education*, 7,1-10. Disponível em:
<https://stemeducationjournal.springeropen.com/articles/10.1186/s40594-020-00205-8>
Acessado em: 20 mai. 2022.