

Robótica Móvel

Ferramental

Prof. Douglas G. Macharet
douglas.macharet@dcc.ufmg.br

Introdução

Python

CoppeliaSim

Python

- Simples e fácil de aprender → alto nível
 - Ótima ferramenta de prototipação
- Quantidade enorme de bibliotecas disponíveis
- Muito popular atualmente
 - Ciência dos Dados e Aprendizado de Máquina
- Características
 - Propósito geral, interpretada, dinamicamente tipada



Python

Instalação (Python \geq 3.7)

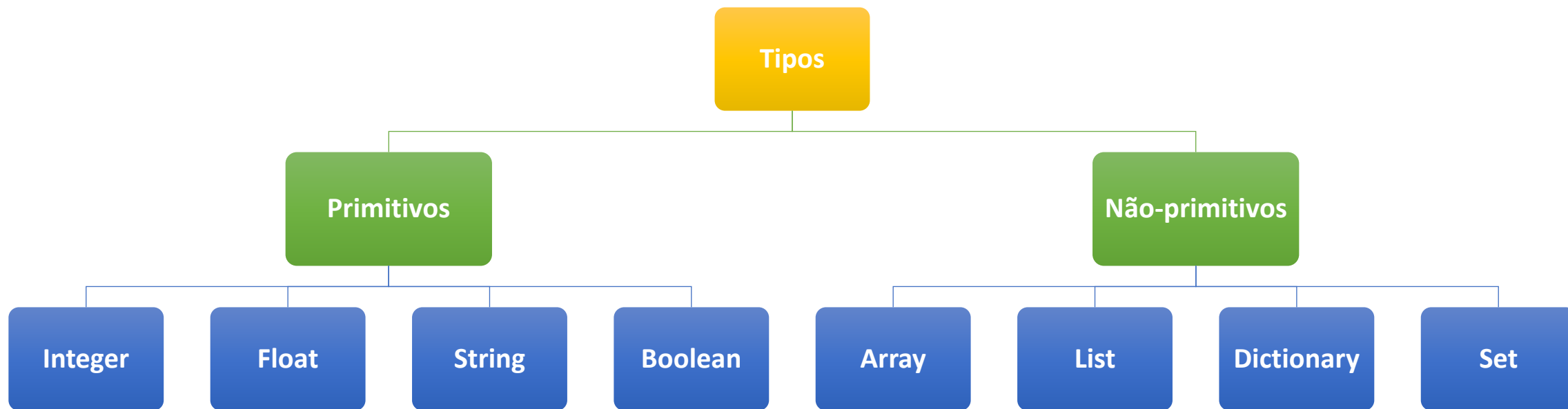
- Plataforma integrada
 - Interpretador da linguagem
 - Gerenciador de pacotes (distribuição)
 - Ambiente de desenvolvimento
- Gratuita e open-source
- Multiplataforma: Windows, Linux e Mac



- Bibliotecas

- SciPy (NumPy) → Computação científica
- Matplotlib → Visualização
- shapely / scikit-geometry → Geometria

- Caso necessário, instalar via “pip” ou Anaconda Navigator
 - Faça uso de ambientes virtuais para evitar conflitos



**Escopo definido
pela indentação**

```
def foo(choice):  
    if choice:  
        return "Oi, Mundo"  
    else:  
        return 3.1415
```

Tipagem dinâmica

Operadores

| | | | | | | | |
|-------------|-----|--------|----|----|-----|-----|-----|
| Aritméticos | + | - | * | / | % | // | ** |
| Comparação | > | < | == | != | >= | <= | |
| Lógicos | and | or | | | | | |
| Bitwise | & | | ~ | ^ | >> | << | |
| Atribuição | = | += | -= | *= | /= | %= | //= |
| | **= | &= | = | ^= | >>= | <<= | |
| Identidade | is | is not | | | | | |
| Membership | in | not in | | | | | |

Não disponíveis

exp++
exp--
++exp
--exp

Python

Listas

```
>>> a = ['spam', 'eggs', 100, 1234]
>>> a[0]
'spam'
>>> a[3]
1234
>>> a[-2]
100
>>> a[1:-1]
['eggs', 100]
```

Dicionários

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['guido'] = 4128
>>> tel
{'guido': 4128, 'irv': 4127, 'jack': 4098}
>>> tel.keys()
['guido', 'irv', 'jack']
>>> 'guido' in tel
True
```

Python

Iteradores

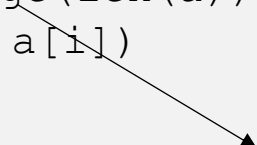
```
>>> a = ['cat', 'window', 'defenestrated']
>>> for x in a:
    print(x, len(x))

cat 3
window 6
defenestrated 12
```

Ranges

```
>>> a = ['Mary', 'had', 'a', 'little', 'lamb']
>>> for i in range(len(a)):
    print(i, a[i])

0 Mary
1 had
2 a
3 little
4 lamb
```



```
>>> range(1, 5)
[1, 2, 3, 4]

>>> range(1, 10, 2)
[1, 3, 5, 7, 9]

>>> range(5, 1, -1)
[5, 4, 3, 2]

>>> range(5)
[0, 1, 2, 3, 4]

>>> range(len(a))
[0, 1, 2, 3, 4]
```

- Número variável (indefinido) de argumentos para uma função
 - `*args` (non-keyworded arguments) → Listas (iterables)
 - `**kwargs` (keyworded arguments) → Dicionários
- Argumentos `args` e `kwargs` podem ter qualquer nome (convenção), o importante é o ***unpacking operator*** utilizado como precedente
- A declaração deve seguir uma ordem específica

```
def my_function(a, b, *args, **kwargs)
```

Outros parâmetros

Python

Iterables

```
def somatorio(numeros):  
    soma = 0  
    for n in numeros:  
        soma += n  
    return soma
```

```
lista = [1, 2, 3]  
print(somatorio(lista))
```

```
def somatorio(*args):  
    soma = 0  
    for n in args:  
        soma += n  
    return soma
```

```
print(somatorio(1, 2, 3))
```

Dictionaries

```
def print_info(**kwargs):  
    for key, value in kwargs.items():  
        print("%s: %s" % (key, value))
```

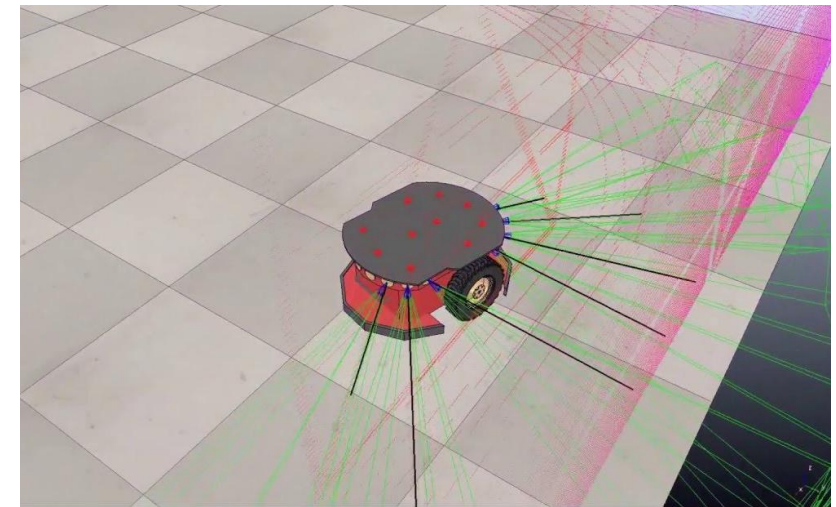
```
print_info(Nome="João", Sobrenome="Silva", Idade=18)
```

<https://docs.python.org/3/tutorial/>

<https://algoritmoempython.com.br/cursos/programacao-python/intro/>

CoppeliaSim

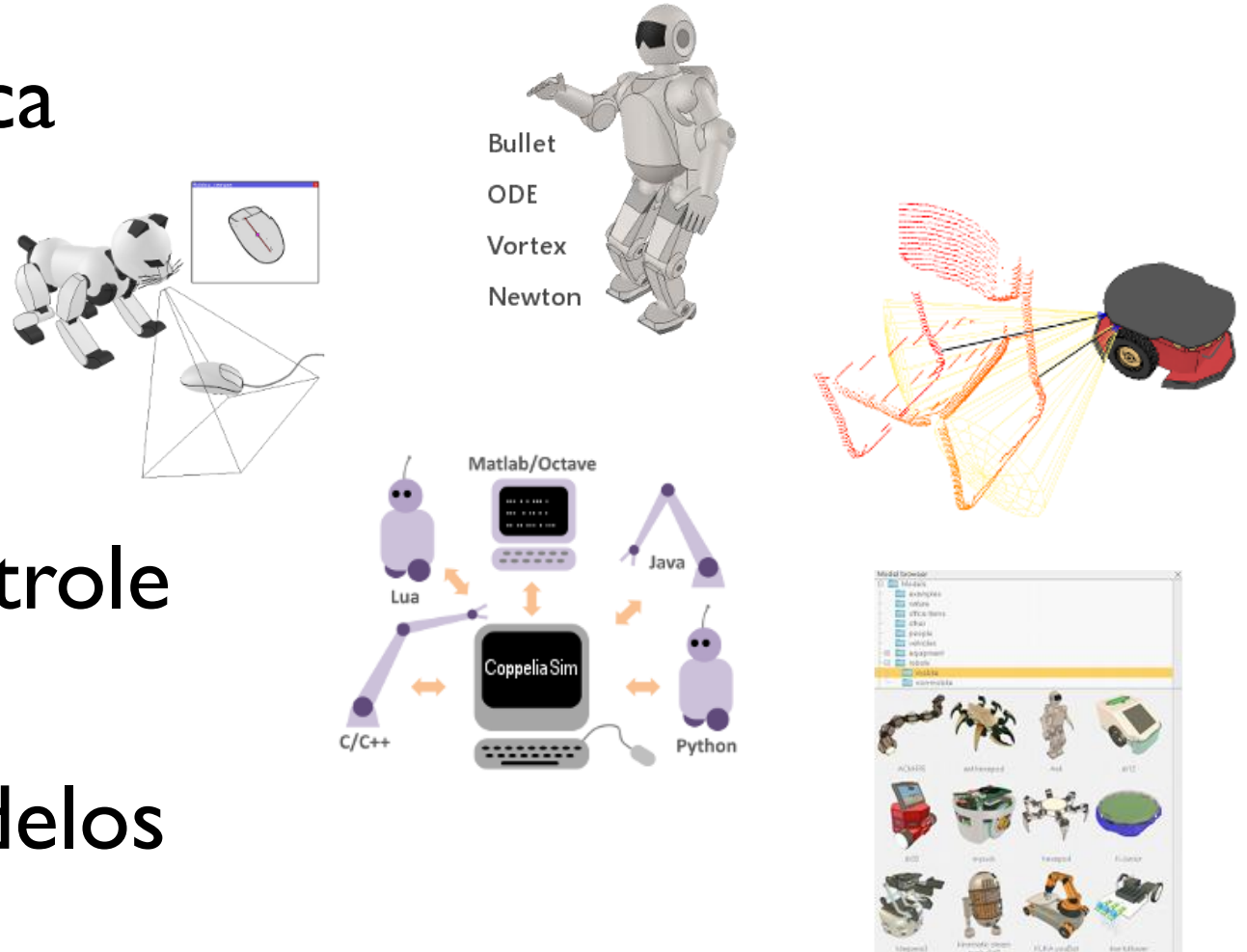
- Sucessor do V-REP (V4.2.0)
 - Virtual Robot Experimentation Platform
- Simulador de propósito geral
 - Hardware/software
 - Prototipagem e verificação rápidas
 - Desenvolvimento de algoritmos
- Gratuito (EDU) e open-source
- Multiplataforma: Windows, Linux e Mac



CoppeliaSim

Características

- Engines para física/dinâmica
- Simulação de sensores
- Diferentes formas de controle
- Grande biblioteca de modelos



CoppeliaSim

Elementos

Construção

**Scenes
Objects**

Simulação

**Calculation
Modules**

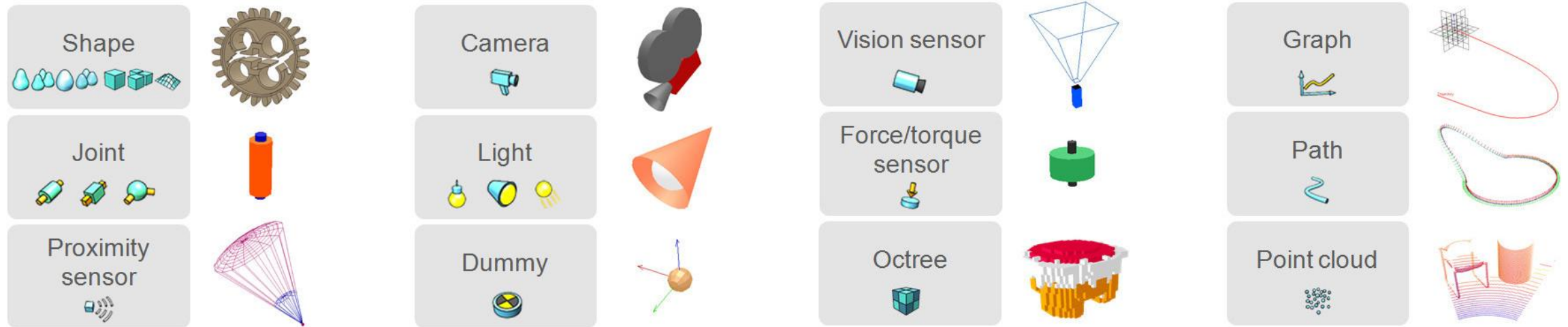
Controle

**Control
Mechanisms**

CoppeliaSim

Scene objects

- Usados para construir uma cena de simulação
 - Robô: estrutura hierárquica incluindo (pelo menos) formas e juntas



- Funcionalidades que operam em um ou vários objetos



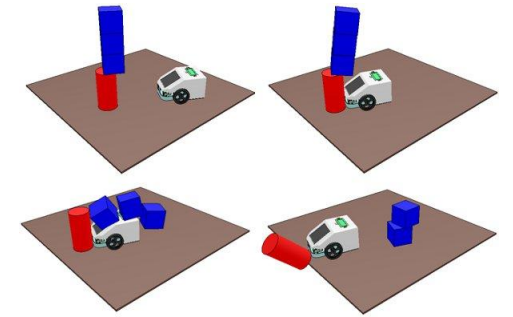
Collision detection



Distance calculation



Forward/Inverse kinematics



Dynamics

CoppeliaSim

Control mechanisms

■ Diferentes interfaces e linguagens (separadas/combinaadas)

| | Embedded script | Add-on / sandbox script | Plugin | Client application | Remote API client | ROS / ROS2 node | ZeroMQ node |
|--|------------------------------|------------------------------|--|--|---|---------------------------------|---------------------------------|
| Control entity is external (i.e. can be located on a robot, different machine, etc.) | No | No | No | No | Yes | Yes | Yes |
| Supported programming language | Lua, Python | Lua, Python | C/C++ | C/C++, Python | C/C++, Python, Java, JavaScript, Matlab, Octave | Any ¹ | Any |
| Code execution speed | Relatively fast ² | Relatively fast ² | Fast | Fast | Depends on programming language | Depends on programming language | Depends on programming language |
| Communication lag | None ³ | None ³ | None | None | Yes | Yes | Yes |
| Communication channel | Python: ZeroMQ ³ | Python: ZeroMQ ³ | None | None | ZeroMQ or WebSockets | ROS / ROS2 | ZeroMQ |
| Control entity can be fully contained in a scene or model, and is highly portable | Yes | No | No | No | No | No | No |
| Stepped operation ⁴ | Yes, inherent | Yes, inherent | Yes, inherent | Yes, inherent | Yes | Yes | Yes |
| Non-stepped operation ⁴ | Yes, via threads | Yes, via threads | No (threads available, but API access forbidden) | No (threads available, but API access forbidden) | Yes | Yes | Yes |

¹⁾ Depends on ROS / ROS2 bindings

²⁾ Depends on the programming language, but the execution of API functions is very fast

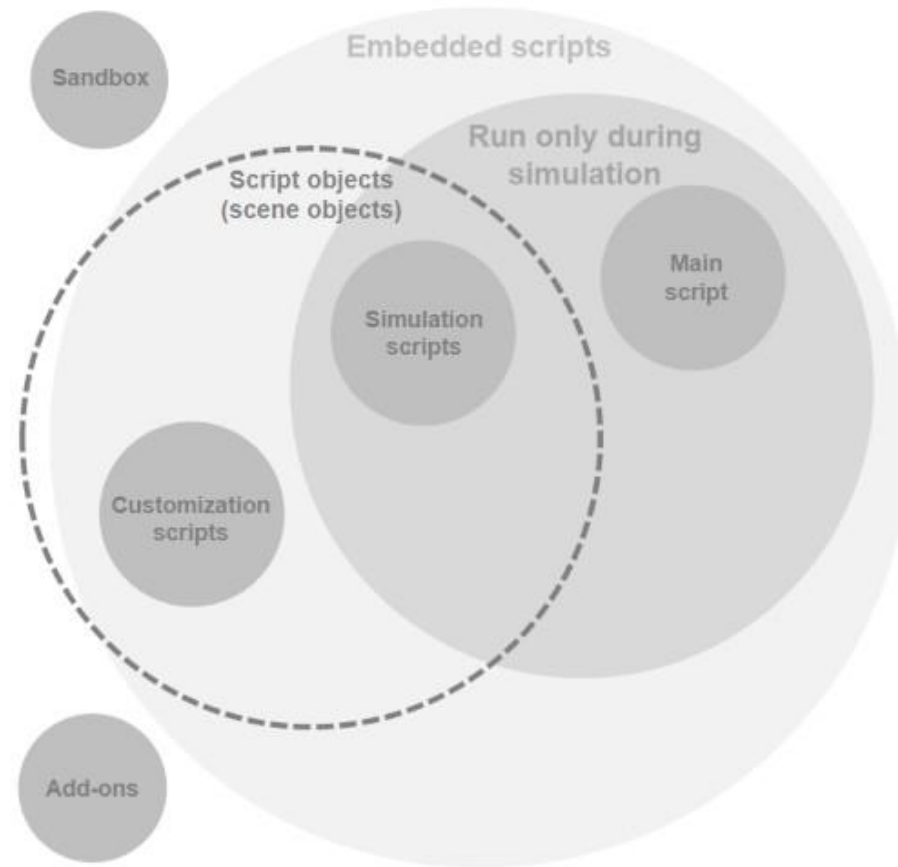
³⁾ Lua scripts are executed in CoppeliaSim's main thread, Python scripts are executed in separate processes

⁴⁾ Stepped as in *synchronized* with each simulation step

<https://www.coppeliarobotics.com/helpFiles/en/writingCode.htm>

CoppeliaSim

Control mechanisms



<https://www.coppeliarobotics.com/helpFiles/en/scripts.htm>

- Interface para enviar/receber comandos de/para o simulador
 - B0-based remote API: segunda versão da API remota, baseada no middleware BlueZero. Fácil de estender, muitas dependências.
 - Legacy remote API: É comparativamente mais leve e tem menos dependências do que a B0-based. No entanto, é menos flexível.
 - ZeroMQ remote API: Oferece uma combinação ideal de alto desempenho, baixo overhead e flexibilidade. Todas funções da API do CoppeliaSim.
- Linguagens
 - C/C++, Java, Python, Matlab, Octave e Lua

CoppeliaSim

ZeroMQ remote API – Exemplo 0

■ Instalação

```
$ python3 -m pip install coppeliasim-zmqremoteapi-client
```

■ Cliente (seu programa)

```
from coppeliasim_zmqremoteapi_client import RemoteAPIClient

client = RemoteAPIClient()
sim = client.require('sim')

sim.setStepping(True)

sim.startSimulation()
while (t := sim.getSimulationTime()) < 3:
    print(f'Simulation time: {t:.2f} [s]')
    sim.step()

sim.stopSimulation()
```

<https://manual.coppeliarobotics.com/en/zmqRemoteApiOverview.htm>

<https://manual.coppeliarobotics.com/en/simulation.htm#stepped>

■ Referenciando elementos na simulação → Handle

```
objectHandle = sim.getObject("/Path/to/Object")  
objectHandle = sim.getObject("/Object")
```

■ Acessando informações dos elementos

```
list position = sim.getObjectPosition(int objectHandle, int relativeToObjectHandle = sim.handle_world)  
list eulerAngles = sim.getObjectOrientation(int objectHandle, int relativeToObjectHandle = sim.handle_world)  
list linearVelocity, list angularVelocity = sim.getObjectVelocity(int objectHandle)
```

■ Enviando comandos de velocidade

```
sim.setJointTargetVelocity(int objectHandle, float targetVelocity, float[] motionParams = [])
```

CoppeliaSim

ZeroMQ remote API – Exemplo 1

```
import time
import matplotlib.pyplot as plt
from coppeliasim_zmqremoteapi_client import RemoteAPIClient

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

# Parar a simulação se estiver executando
initial_sim_state = sim.getSimulationState()
if initial_sim_state != 0:
    sim.stopSimulation()
    time.sleep(1)

objectPath = '/Bill/Bill'
objectHandle = sim.getObject(objectPath)

hist = []
```

```
sim.startSimulation()
while (t := sim.getSimulationTime()) < 25:
    print(f'Simulation time: {t:.2f} [s]')

    position = sim.getObjectPosition(objectHandle, sim.handle_world)
    print('Pos: ', position)

    orientation = sim.getObjectOrientation(objectHandle, sim.handle_world)
    print('Ori: ', orientation)

    hist.append([position[0], position[1]])

    sim.step()

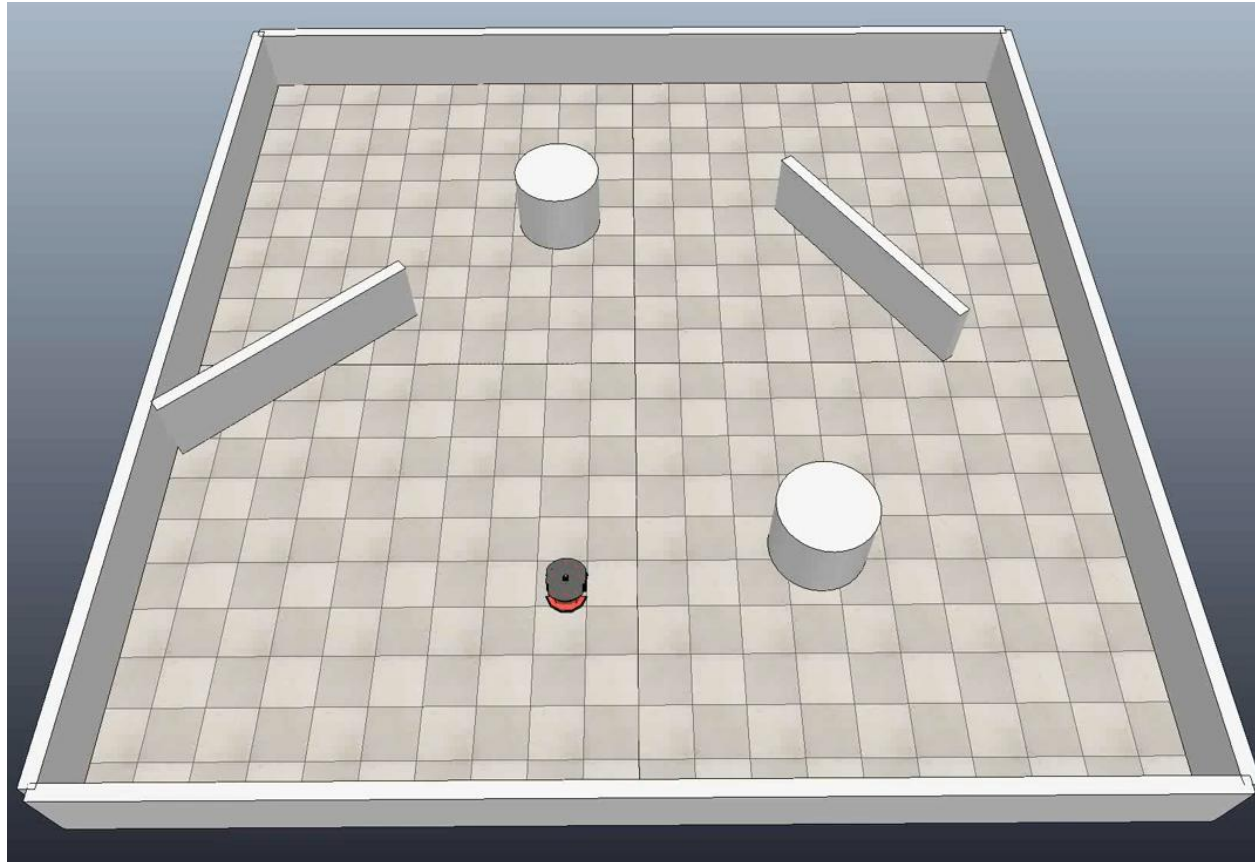
sim.stopSimulation()

fig = plt.figure(figsize=(6,6), dpi=100)
ax = fig.add_subplot(111, aspect='equal')

x, y = zip(*hist)
ax.scatter(x, y)
```

CoppeliaSim

ZeroMQ remote API – Exemplo 2



<https://youtu.be/Zt302rzdazE>

Considerações finais

- Vantagens

- Prototipação e avaliação rápidas
- Documentação online muito boa
- Curva de aprendizado razoavelmente pequena

- Desvantagens

- Média/Alta demanda computacional
- Quantidade de opções pode confundir