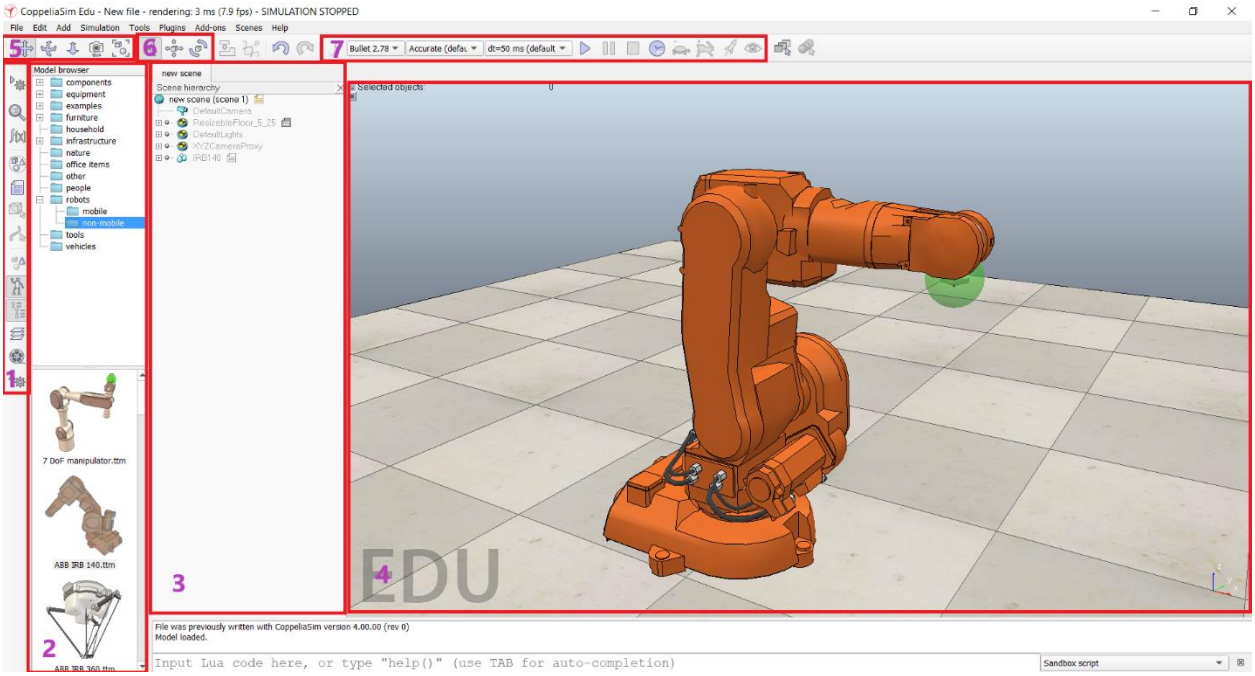
Методы и алгоритмы управления промышленными роботами (практика)

Знакомство с симулятором

Симулятор позволяет работать с моделью робота в виртуальной среде при разработке управляющей программы, проверке траекторий движения инструмента, тестировании различных алгоритмов управления и взаимодействия, а также для решения ряда других задач.

В данном курсе используется симулятор **CoppeliaSim** (бывший **V-REP**). «Educational» версия этого симулятора бесплатна для личного использования и в учебных целях, и может работать на базе основных операционных систем (**Windows**, **MacOS**, **Linux**).



Основные элементы интерфейса

1. Иконки быстрого доступа к элементам меню
2. Библиотека 3D моделей
3. Структура элементов сцены
4. Визуализация, окно взаимодействия с моделью
5. Управление положением/ориентацией/размером сцены
6. Управление положением/ориентацией элементов сцены
7. Управление процессом симуляции

Виды промышленных роботов

В разделе *robots/non-mobile* находятся модели различных типов роботов. Чтобы перенести модель на сцену, выделите её и перетащите в окно визуализации. Для удаления робота из сцены выделите его и нажмите *Delete*. С каждым роботом связан демонстрационный сценарий работы симулятора. Чтобы его запустить, нажмите кнопку с синим треугольником в меню управления.

1. Шестиосевой промышленный робот последовательной структуры на примере **ABB IRB 140**. При запущенной симуляции и выделенном роботе появляется меню управления, в котором можно изменять как углы отдельных шарниров, так и управлять положением/ориентацией в декартовом пространстве. Если выделить зелёную сферу и нажать кнопку *Object/item shift*

(куб со стрелками), можно изменять положение робота движением мыши. Аналогично, можно управлять ориентацией инструмента с помощью кнопки *Object/item rotate*.

2. Робот с параллельной структурой на примере **ABB IRB 360** (дельта-робот). Подобная конструкция широко используется для сортировки и упаковки мелких предметов (лекарства, конфеты и т.п.).
3. Робот с кинематической избыточностью на примере **KUKA LBR iiwa**. Наличие седьмой оси повышает манёвренность конструкции, позволяя поддерживать желаемые положение и ориентацию инструмента при различных конфигурациях робота. Данный робот относится также к коллаборативным, т.е. не рассчитан на высокие грузоподъёмность и быстродействие, зато оснащён силомоментными датчиками, позволяющими детектировать коллизию с препятствием. Это повышает его безопасность при работе в непосредственной близости с человеком.
4. Посмотрите любые другие модели, которые вам покажутся интересными.

Знакомство с Python

Python – интерпретируемый высокоуровневый язык программирования. Благодаря удобству синтаксиса, поддержке различных парадигм программирования и интеграции с **C/C++** приложениями данный язык программирования является одним из самых распространённых на сегодня. Программа на **Python** называется сценарием, для её выполнения используется интерпретатор.

При установке **Python** убедитесь, что используете 64-битную версию, а также не забудьте поставить галку в пункт «*Add to PATH*».

Основные элементы **Python**

- Строки и отступы. Программа выполняется построчно. Отступы используются для формирования логических блоков: все строки в одном блоке должны иметь одинаковый отступ. Строчные комментарии начинаются со знака «#».
- Переменные. Могут принимать любое значение (число, строка, список и т.п.). Могут быть объявлены в любой части кода в виде `имя_переменной = значение_переменной`

```
p = 123                # например, определение числового параметра
```

- Элементарные типы данных. Числа могут быть целыми и с плавающей запятой, и поддерживают основные арифметические операции. Строки задаются в виде последовательности символов, заключённой в одинарные или двойные кавычки. Булевы переменные могут иметь логические значения *True* и *False*.

```
p = 1.0 + 2*3          # числовая переменная
p = "abc"              # эквивалентно p = 'abc'
p = False              # логические значения
```

  Для создания блока текста, содержащего несколько строк, используются тройные кавычки в начале и конце этого блока. Данный подход может быть использован, в том числе, чтобы закомментировать блок кода.

- Структуры данных. Основными структурами данных в **Python** являются списки, словари и кортежи. Списки служат для доступа по индексу (порядковому номеру элемента), и являются динамически изменяемыми структурами, т.е. могут менять свой размер. Словари обеспечивают доступ по ключевому слову. Кортежи также обеспечивают доступ по индексу, но являются неизменяемыми. Индексация с нуля.

```
p = [2, "abc", 3.3]  # создание списка
print(p[0])          # вернёт 2
p = {key1: 2, key2: "abc", key3: 3.3]  # создание словаря
print(p.key2)        # вернёт "abc"
```

```
p = (2, ”abc”, 3.3)  # создание кортежа
print(p[2])          # вернёт 3.3
```

- Ветвление. Используется для изменения алгоритма поведения в зависимости от некоторого условия. Может содержать либо не содержать альтернативные варианты поведения.

```
if условие1:
    алгоритм1
elif условие2:
    алгоритм2
else:
    алгоритм3
```

- Циклы. Используются для выполнения повторяющихся действий и обхода элементов списков и словарей.

```
# выход из цикла, когда условие неверно
while условие:
    действие
# обход элементов списка
for элемент in список:
    действие
```

- Функции. Позволяют выполнять некоторые действия или вычислять значения.

```
# объявление функции
def имя_функции(аргументы):
    тело_функции
    return результат   # опционально
# вызов
p = имя_функции(аргументы)
# можно создавать синонимы
p = имя_функции
p(аргументы)           # эквивалентно имя_функции(аргументы)
```

- Объекты и методы. Язык позволяет работать в объектно-ориентированном стиле, и большинство библиотек написаны с использованием данного подхода.

```
p = конструктор_класса()   # создание объекта
p.метод()                  # вызов метода класса
```

- Библиотеки. Позволяют использовать код, созданный сторонними разработчиками.

```
import библиотека                # загрузка библиотеки
import библиотека as новое_имя   # доступ через заданное имя
from библиотека import элемент   # загрузка отдельных элементов
```

Задание. Напишите в Python функцию, которая принимает на вход массив чисел и возвращает сумму, среднее значение, а также новый массив, все элементы которого двукратно превышают исходные числа. Вам могут пригодиться элементы языка:

- `len(A)` – число элементов массива A
- `for i in range(3):` – объявление цикла, где i изменяется от 0 до 2 включительно
- `return a, b` – возврат нескольких значений
- `print(a,b)` – вывод на печать значений переменных

Настройка управления симулятором

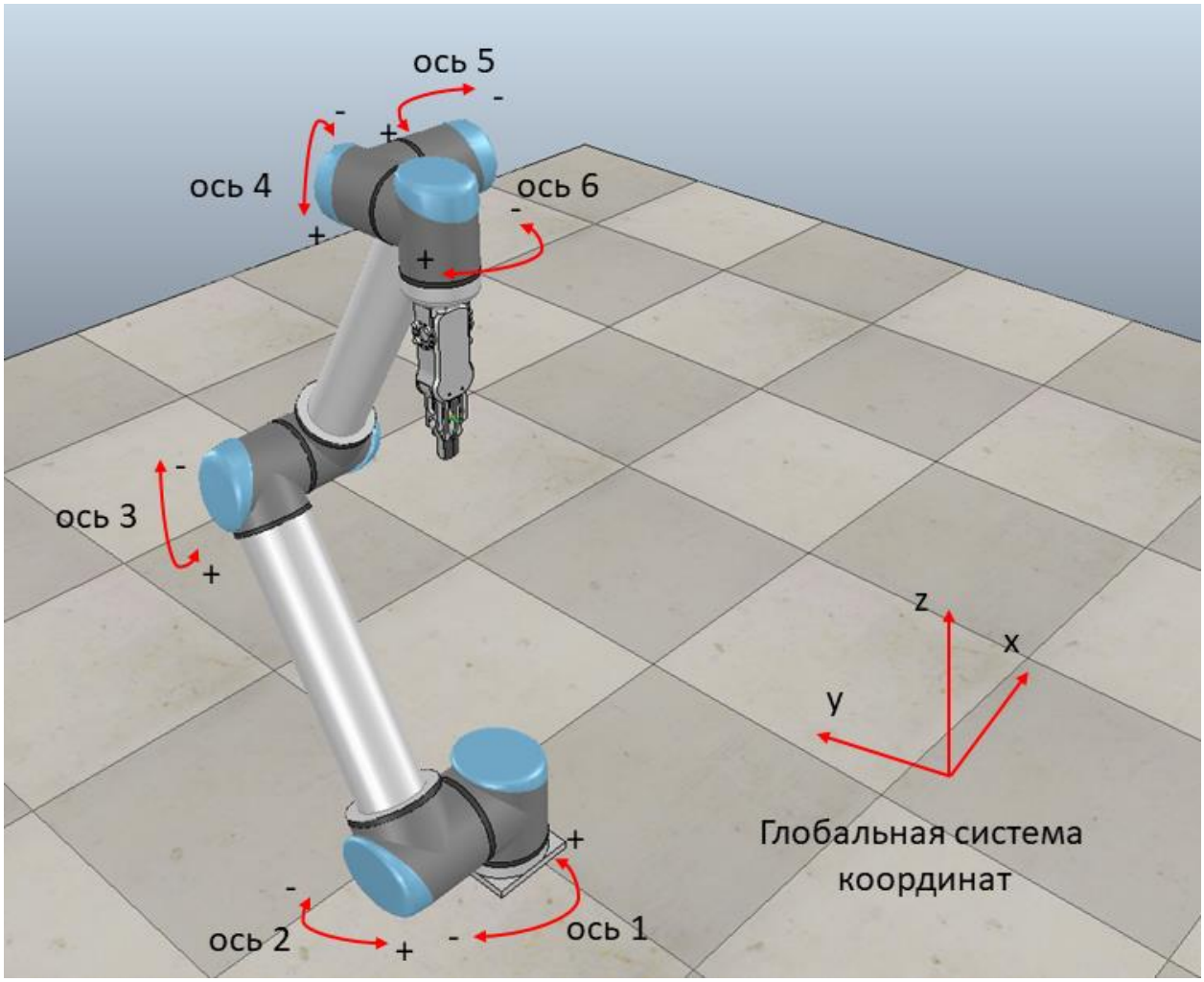
Для взаимодействия с роботом через внешний интерфейс необходимо выполнить ряд настроек.

- Создайте папку, в которой будут храниться файлы проектов, загрузите в неё сцены симулятора и прилагающиеся к ним файлы управления.

- Откройте директорию, в которую был установлен симулятор, скопируйте в созданную ранее папку следующие файлы:
    - *CoppeliaSimEdu\programming\remoteApiBindings\python\python\sim.py*
    - *CoppeliaSimEdu\programming\remoteApiBindings\python\python\simConst.py*
    - *CoppeliaSimEdu\programming\remoteApiBindings\python\python\simpleTest.py*
    - *CoppeliaSimEdu\programming\remoteApiBindings\lib\lib\ваша_ОС\remoteApi*
- Убедитесь, что номер порта в *CoppeliaSimEdu\remoteApiConnections.txt* равен 19997, если это не так, запомните свой порт.
- Чтобы убедиться, что удалённое управление работает:
    - Проверьте и, при необходимости, поменяйте номер порта в функции `sim.simxStart('127.0.0.1',ВАШ_ПОРТ,True,True,5000,5)` файла *simpleTest.py* в своей рабочей папке.
    - Откройте любую сцену в симуляторе. Запустите файл *simpleTest.py*. В случае успеха вы должны увидеть строку «Program started» и затем последовательность сообщений вида «Mouse position x: …».

Знакомство с системой управления

Для знакомства с основами управления манипулятором вам предлагается выполнить ряд заданий. При этом будет использоваться следующая модель шестиосевого робота.



Основные функции для работы с роботом содержатся в библиотеке *ur_vrep.py*, а для решения задач предлагается использовать *solutionN.py*. Каждый из файлов *solution* содержит часть необходимого кода, но основное решение пропущено.

Для взаимодействия с роботом могут быть использованы следующие команды.

- `ptp([10,20,30,40,50,60])` – движение в заданное положение в пространстве конфигураций
- `lin([0.1,0.2,0.3],[40,50,60])` – установить заданное положение и ориентацию в Декартовом пространстве
- `lin([0.1,0.2,0.3])` – движение в заданное положение с постоянной ориентацией
- `gripperOpen(True)` – открыть захват
- `getJoints()` – получить текущие углы в осях
- `getPosition()` – получить текущее положение инструмента
- `getOrientation()` – получить текущую ориентацию инструмента
- `setCvel(x=2)` – удвоить скорость по оси X
- `setCacc(y=3)` – утроить ускорение по оси Y
- `setJvel(q3=0.5)` – уменьшить в 2 раза скорость для 3ей оси
- `setJacc(q4=3)` – утроить ускорение по четвёртой оси

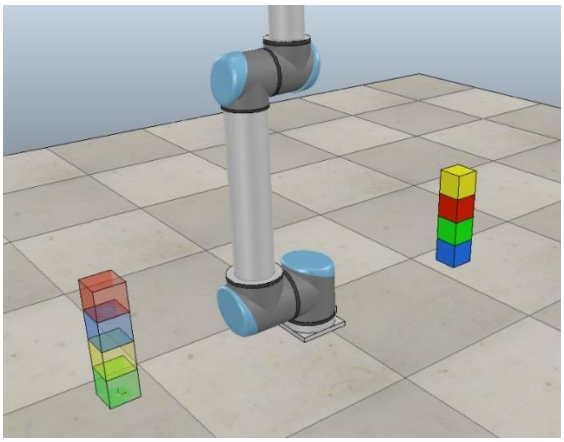
Углы задаются в градусах.

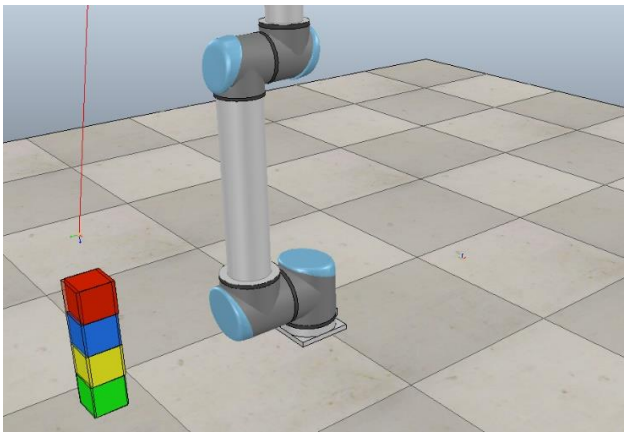
Задания

Задание 1. Откройте в симуляторе файл *scene1.ttt*. Напишите в *solution1.py* последовательность команд, которые выполняют следующие действия.

- Уменьшите угол второй оси на 20° и уменьшите четвёртый на 20°.
- Опустите инструмент по Z на 0,3 м.
- Выполните вращение инструментом по X и по Z на 30°, затем вернитесь к исходной ориентации.
- Уменьшите скорость по Y в 2 раза, выполните движение инструмента вдоль этой оси к координатам базы робота с сохранением высоты (т.е. в плоскости X0Y) с одновременным вращением инструмента на 180° по Z. Объясните поведение робота в конечной точке.

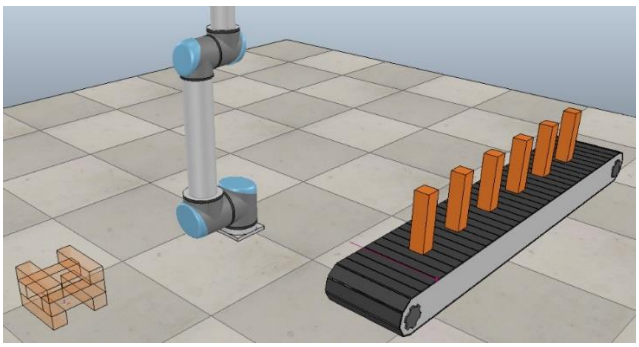
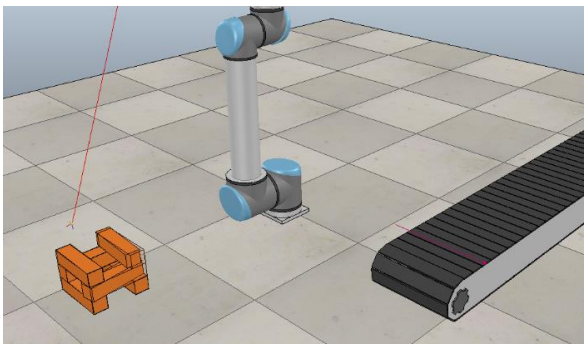
Задание 2. Напишите программу управления, чтобы переложить кубики из правой башни в левую с учётом заданных цветов (*scene2.ttt*).



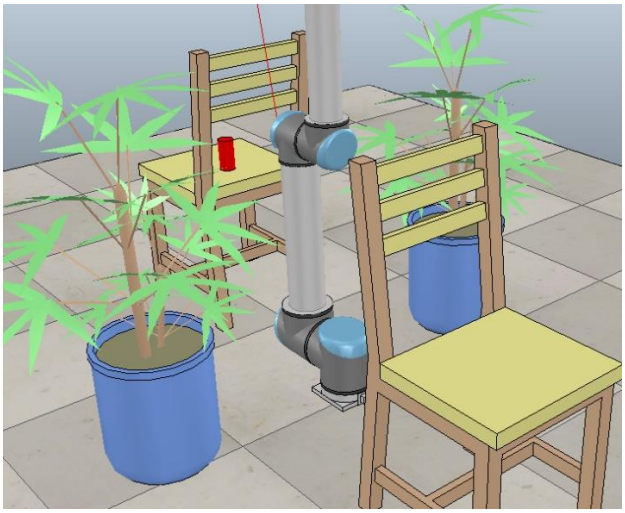


Подумайте, можно ли ускорить ваше решение, не изменяя настроек скорости?

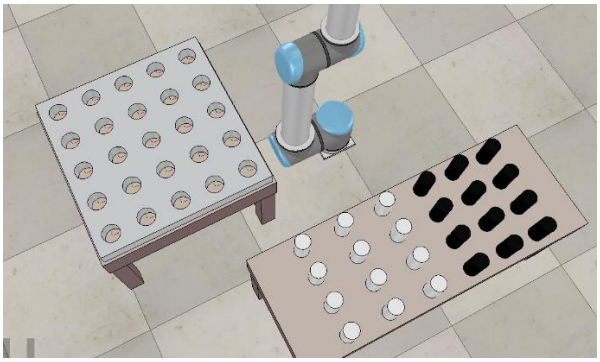
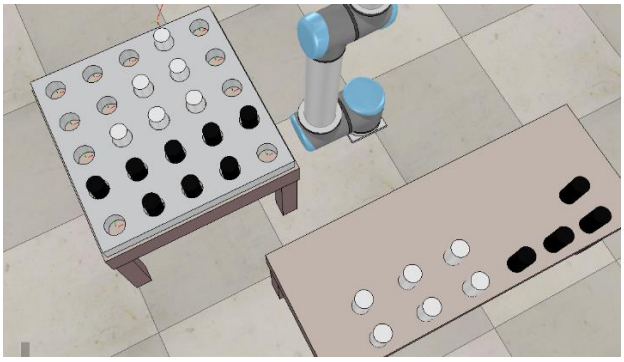
Задание 3. Напишите программу управления, чтобы робот брал бруски с конвейера и строил из них башню (*scene3.ttt*).

Задание 4. Напишите программу для перемещения цилиндра с одного стула на другой (*scene4.ttt*).

Задание 5. Напишите программу для создания рисунка из белых и чёрных цилиндров (*scene5.ttt*).

Подсказка: обратите внимание на регулярную структуру в расположении цилиндров и отверстий.

Ссылки

1. http://coppeliarobotics.com/downloads - симулятор
2. http://coppeliarobotics.com/helpFiles/index.html - руководство пользователя
3. https://www.python.org/downloads/ - Python3