ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО

ОБРАЗОВАНИЯ

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ИМ. Р.Е. АЛЕКСЕЕВА

**ЛАБОРАТОРНАЯ РАБОТА №1**

**Выполнил**: Тронин А.А. 19-В-1

**Проверил**: Гай В.Е.

**Оценка**: \_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород

2021

**Тема**: Введение в программирование алгоритмов управления роботами с использованием симулятора Webots

**Цель**: получение навыков работы с симулятором Webots

**Вариант задания:**

**7.** [**https://cyberbotics.com/doc/guide/firebird6**](https://cyberbotics.com/doc/guide/firebird6)

1. Добавить робота, указанного в списке ниже и соответствующего выданному варианту в симулятор Webots

2. Создать для робота контоллер таким образом, чтобы робот двигался не по прямой (для этого можно для разных моторов робота установить разную скорость)

**Краткие теоретические сведения:**

Итак, представляю вашему вниманию webots. Из коробки с этим симулятором идёт большое количество демок, которые сразу можно открыть и поразглядывать. Вот одна из них. Robots - gctronics - e-puck - e-puck.wbt. После того как файл загрузился появляется виртуальное пространство в котором находятся статические объекты и робот. Этим роботом можно управлять с клавиатуры.

Webots для моделирования физики твёрдого тела использует Open Dynamics Engine. Поэтому в Webots можно создавать объекты с массой, моделировать трение, обнаруживать столкновения.

У роботов могут быть разные схемы движения, то есть роботы могут быть колёсным, летающими, шагающим, плавающими. В состав робота могут входить различные сенсоры и приводы.

Самое важное, что для робота можно написать программу управления, или как она называется в Webots - контроллер. Сейчас исходный код контроллер робота отображается в правой части окна.

Управлять движением робота в симуляторе можно на Си, Си++, Питоне, Джаве и Матлабе. Также можно создавать свои модели роботов и статичных объектов, но для этого нужны небольшие знания по созданию трёхмерных моделей и языка VRML97.

Вообще, в Webots симуляция включает следующие элементы:

1. Файл виртуального пространства Webots с расширением wbt, в котором находятся описания роботов и окружающего пространства.

2. Один или несколько контроллеров для роботов

3. Опционально, плагин, который может модифицировать физику Webots симулятора.

Виртуальное пространство в Webots содержит описание свойств робота и его окружения. Описание каждого объекта включает в себя: положение, ориентацию, геометрию, внешний вид, физические свойства. Это область в левой части экрана.

Виртуальное пространство организовано в виде иерархической структуры, то есть объект может включать другие объекты. Например, робот может включать четыре колеса, датчик расстояния и манипулятор, а в состав манипулятора может входить, например камера и ещё какие-то элементы.

Файл описания виртуального пространства не включает контроллер, который описывает поведение робота. Этот файл содержит только имя контроллера для робота.

Что такое контроллер? Контроллер это программа, которая управляет поведением робота. Он запускается при старте симуляции, при этом каждый контроллер запускается как отдельный процесс. И процесс контроллера связывается с моделью робота. Несколько роботов могут использовать один и тот же код контроллера, но для каждого робота в этом случае будут запускаться отдельные процессы.

При использовании для написания контроллера Си или Си++ контроллер должен быть скомпилирован. При использовании Матлаба или Питона в операционной системе должен быть установлен соответствующий интерпретатор, который нужно подключить к Webots.

Исходные и бинарные файлы каждого контроллера хранятся вместе в каталоге контроллера, который называется “controllers”. Эта папка должна находиться в папке проекта Webots.

В нижней части экрана отображаются результат компиляции и вывод контроллера.

Давайте теперь посмотрим на меню симулятора.

Меню Файл включает стандартные элементы по созданию нового виртуального пространства (мира), его сохранения или открытия существующего мира. Пункт Reload World загружает текущее виртуальное пространство из сохранённой версии и начинает симуляцию с начала.

Пункт Reset Simulation восстанавливает исходное состояние симуляции, то элементы виртуального пространства получают начальное состояние, а виртуальное пространство заново не загружается.

В меню Файл также можно создать и сохранить новый текстовый файл, импортировать VRML модель в конец дерева объектов. Обычно это используется для добавления пользовательских объектов в среду симулятора. Также виртуальное пространство можно экспортировать в VRML. Можно сделать скриншот сцены и запустить видеозапись симуляции.

В меню Правка или Edit находятся классические операции по правке текста.

Пункты меню View предназначены для управления обзором за виртуальным пространством.

Пункт меню Follow Object позволяет переключаться между статической точкой обзора на мобильный объект и точкой обзора, которая следует за объектом. Чтобы камера следила за объектом нужно выбрать объект на сцене и затем выбрать один из пунктов меню Follow Object.

Пункт Restore Viewpoint сбрасывает положение точки обзора в начальное значение. Обычно выполнить такое действие требуется когда вы потерялись в сцене и хотите вернуться к изначальной точке обзора.

Пункт Move viewpoint to object помещает точку обзора на выбранный элемент сцены.

Пункт Change View устанавливает точку зрения в соответствии с одним из шести положений: и показывает вид слева, справа, снизу, сверху, спереди или сзади.

Кнопка Projection позволяет выбрать между перспективной и ортографической проекцией, то есть проекцией в которой отображаются объекты в окне симулятора. Перспективная проекция соответствует естественной проекции в которой чем дальше объект от наблюдателя, тем меньше он выглядит. В ортографической проекции расстояние до наблюдателя не влияет на размер объекта. В Ортографической проекции линии, параллельные в модели рисуются параллельными на экране. В ортографической проекции не отображаются тени.

Кнопка Rendering позволяет выбирать между полным и каркасным рендером. В полном рендере все объекты отображаются с учётом материалов, текстур, цветов. В каркасном рендере отображаются только геометрические примитивы из которых состоят объекты. Такой режим полезен при отладке вновь создаваемых объектов и отладке столкновений. Также есть меню Optional Rendering, в нём можно выбирать какие именно элементы сцены и какая дополнительная информация должны отображаться.

Пункт Disable selection запрещает изменение выбранного объекта. Это полезно при моделировании, когда нужно поменять точку обзора без изменения видимых и выбранных объектов сцены.

Пункт Lock Viewpoint запрещает изменение положения и ориентации точки обзора.

Меню симуляции (кнопки на панели)

На панели инструментов есть кнопки, которые позволяют поставить симуляцию на паузу , выполнить пошагово , в реальном времени , в быстром и очень быстром режимах. В последнем режиме рендер отключается и этот режим подходит для симуляций, которые интенсивно используют процессор.

Меню Overlays. Это меню управляет окнами, в которых отображается показания камеры, лазерного дальномера и других датчиков. Некоторые пункты этого меню будут активны только при выбранном роботе.

Меню Tools показывает окна, которые открыты в Webots.

Пункт меню Tools-Preferences включает настройки Webots: выбранный язык, внешний вид среды, режим симуляции в котором запускается Webots, число потоков, отводимых Webots, то, как выглядит команда для запуска питона, настройки рендера, связанные с OpenGL.

Меню помощников (Визардов) содержит команды для запуска диалоговых окон для создания новых проектов и контроллеров.

Меню Help. Как мне кажется самое интересное, что есть в меню Help это информация об OpenGL, видеокарточке и драйвере. Почему? Чтобы понять проблемы с рендером.

На главной панели инструментов есть кнопка для добавления новых объектов и ярлыки на пункты меню Файл, View и Simulation.

Спидометр и виртуальное время. Спидометр показывает скорость симуляции на вашем компьютере. Он отображается на главной панели инструментов и показывает насколько симуляция быстрее или медленнее реального времени. Если там отображается 2, тогда симуляция быстрее реального времени в два раза.

Слева от спидометра показывается виртуальное время: часы, минуты, секунды и миллисекунды.

Шаг времени симуляции устанавливается в поле basicTimeStep. Он измеряется в миллисекундах и показывает время выполнения одного шага симуляции в режиме выполнения по шагам.

Давайте перейдём к созданию первой симуляции в Webots. Webots запущен. Теперь нужно поставить на паузу симуляцию и создать новый проект с помощью меню Wizards, выбрав пункт New Project Directory. Установим имя каталога, в котором будет храниться проект first\_sim, имя файла в котором хранится виртуальное пространство first\_sim.wbt и выберем все галочки, включая последний пункт add a rectangle area.

Webots отображает список каталогов и файлов, которые он создал и это стандартная иерархия файлов в проекте.

Теперь после создания симуляции можно изучить структуру сцены, которая расположена в левой части экрана. В ней сейчас находятся следующие узлы:

• WorldInfo: - глобальные параметры симуляции

• Viewpoint: - определяет параметры точки зрения главной.

• TexturedBackground: - фон сцены (горы в текущем примере)

• TexturedBackroundLight: - подсветка фона

• defines the light associated with the above background.

• RectangleArena: объект, который мы добавили в сцену.

Можно перемещать точку обзора с помощью мыши: левой и правой кнопкой мыши и колёсиком. Колёсико мыши масштабирует сцену, зажатая левая кнопка выполняет вращение, зажатая правая кнопка выполняет линейное перемещение.

У каждого узла или элемента сцены есть настраиваемые параметры. Давайте модифицируем некоторые параметры прямоугольной области. Нужно щёлкнуть два раза на узле RectangleArena в дереве сцены и это отобразит его поля.

Установим для поля floorTileSize значения 0.25 и 0.25 вместо 0.5 и 0.5. Это размер плитки, эффект можно увидеть сразу.

Установите значение поля wallHeight в 0.05 вместо 0.1. В результате уменьшится высота стены.

Теперь нужно два раза щёлкнуть на RectangleArena и нажать кнопку Add. В результате откроется диалоговое окно в котором надо выбрать пункт PROTO nodes (Webots Projects) / objects / factory / containers / WoodenBox (Solid). В результате в середине сцены появится большая коробка. Теперь давайте изменим поля этой коробки

1. Установим размер в 0.1 0.1 0.1 вместо 0.6 0.6 0.6.

2. Изменим положение в 0 0.05 0 вместо 0 0.3 0. Также можно потянуть за зелёную стрелку, чтобы изменить координату Игрек.

3. Теперь нужно нажать Shift и переместить коробку в любой угол области

4. Выберите коробку, нажмите Ctrl-C, Ctrl-V для того, чтобы копировать и вставить её, а после с шифтом переместите коробку в какой-нибудь угол.

5. Сохраните виртуальное пространство.

Давайте теперь добавим в сцену робота E-puck. E-puck это маленький робот с дифференциальным приводом, 10 светодиодами и несколькими сенсорами, включая 8 сенсоров расстояния и камеру.

Чтобы добавить робота сначала нужно остановить симуляцию и сбросить время в 0. Это стандартная операция перед добавлением любых объектов. То есть модификация виртуального пространства всегда должна выполняться в следующем порядке: пауза, сброс, модификация и сохранения симуляции.

Выберем последнюю добавленную коробку и нажмите кнопку Add и далее надо выбрать PROTO nodes (Webots Projects) / robots / gctronic / e-puck / E-puck (Robot). Робот должен появится в центре прямоугольной области. Если теперь запустить симуляцию робот должен начать движение, мигать светодиодами и пытаться обходить препятствия. Это поведение запрограммировано в контроллере робота. Также есть в левом верхнем углу чёрное окошко. Это вид с камеры робота. Оно будет оставаться чёрным, пока в контроллере не будет подключена камера. Окошко можно передвигать и закрыть, нажав на крестик. Снова открыть камеру можно через меню Overlays.

Теперь давайте попробуем применить силовое воздействие к роботу. Для этого нужно в режиме симуляции нажать Alt зажать левую клавишу мыши, потянуть и отпустить. Силовое воздействие нельзя применить к коробке, поскольку она не имеет массы и рассматривается как приклеенная к полу. Чтобы сделать так, чтобы коробка что-то весила нужно установить в поле mass определённое значение, например 0.2.

Давайте попробуем создать контроллер для управления роботом. Контроллер будем делать на Питон, который сначала нужно установить. Питон нужен версии 3.7 и обязательно при установке выберите пункт Customize installation (настройка установки) и добавьте Питон в переменную среды для Windows.

Поле controller робота указывает какой контроллер соответствует роботу. Создадим новый контроллер и укажем для него имя epuck\_go\_forward. Для этого нужно выбрать пункт меню Wizards / New Robot Controller. После выполнения этого действия в той папке, где хранятся все файлы виртуального пространство будет создана папка controllers. Шаблон контроллера теперь показан в редакторе кода. Теперь нужно связать робота и созданный контроллер.Для этого нужно выбрать поле controller робота E-puck, нажать Select и выбрать только что созданный контроллер. После этого нужно сохранить виртуальное пространство.

Теперь давайте модифицируем имеющийся код контроллера. Добавим две строчки, которые получают доступ к моторам робота:

# get the motor devices

leftMotor = robot.getMotor('left wheel motor')

rightMotor = robot.getMotor('right wheel motor')

И две строчки для установки расстояния, которые должны пройти колёса.

# set the target position of the motors

leftMotor.setPosition(10.0)

rightMotor.setPosition(10.0)

Информацию о роботе, о том как называются его моторы, о сенсорах можно найти в документации к Webots. Вот например на этой странице https://cyberbotics.com/doc/guide/epuck можно покрутить колёса, посмотреть как именуются сенсоры и где они установлены.

Давайте перейдём в симулятор и я покажу как в реальном времени посмотреть на показания датчиков робота. Для этого нужно два раза щёлкнуть по роботу левой клавишей мыши. Откроется окно, в котором показана схема робота и показания датчиков, но только тех, которые подключены в контроллере. Чтобы отобразить показания всех датчиков нужно нажать кнопку Enable All. После нажатия видны сразу показания датчиков расстояния, датчиков освещённости и другие. Если выбрать пункт меню View - Optional rendering - Show Distance Sensor Rays можно увидеть рендер лучей датчиков расстояния.

Датчик расстояния здесь работает достаточно интересно, зависимость между числом, которое он выдаёт и расстоянием до объекта указана в справочной информации о роботе. Чем большее значение показывает датчик, тем ближе робот к препятствию.

https://cyberbotics.com/doc/guide/epuck

from controller import Robot, Motor

TIME\_STEP = 64

# create the Robot instance.

robot = Robot()

# get the motor devices

leftMotor = robot.getMotor('left wheel motor')

rightMotor = robot.getMotor('right wheel motor')

# set the target position of the motors

leftMotor.setPosition(10.0)

rightMotor.setPosition(10.0)

while robot.step(TIME\_STEP) != -1:

pass

Небольшие пояснения кода программы. В самом начале импортируются функции. После этого из виртуального пространства считывается значение параметра BasicTimeStep.

В цикле while вызывается функция robot.step(timestep). Эта функция Синхронизирует данные контроллера и симулятора. Она должна быть в каждом контроллере и должна вызываться через одинаковые промежутки времени. Параметр функции - время, которое она выполняется, то есть после указанного промежутка времени функция возвращает управление в контроллер, функция вычисляет указанное количество миллисекунд симуляции и возвращает управление. И это время симулятора, а не реальное время. То есть выполнение этой функции может продлиться и одну секунду и две, может и меньше, в зависимости от сложности симуляции.

Условие выхода из цикла - это когда функция step вернёт -1. Эта функция может вернуть -1 когда Webots остановит контроллер. Обычно контроллер останавливается, когда симуляция сбрасывается, выполняется выход из webots, изменяется имя контроллера или загружается новое виртуальное пространство.

Pass - Оператор-заглушка, равноценный отсутствию операции.

Теперь нужно код сохранить, и запустить симуляцию. Если всё хорошо, робот поедет вперёд и остановится когда колёса повернутся на угол в 10 радиан.

Рассмотрим способ управления скоростью колёс. Для этого перед циклом while нужно изменить две строки и добавить три новых. Первые две строки указывают, что колёса робота будут вращаться бесконечно, а вторые две - устанавливают скорость на левом и правом колесе. Запускаю симуляцию и видно, что робот едет со скоростью, меньшей чем в предыдущем примере.

MAX\_SPEED = 6.28

leftMotor.setPosition(float('inf'))

rightMotor.setPosition(float('inf'))

# set up the motor speeds at 10% of the MAX\_SPEED.

leftMotor.setVelocity(0.1 \* MAX\_SPEED)

rightMotor.setVelocity(0.1 \* MAX\_SPEED)

from controller import Robot, Motor

TIME\_STEP = 64

MAX\_SPEED = 6.28

# create the Robot instance.

robot = Robot()

# get a handler to the motors and set target position to infinity (speed control)

leftMotor = robot.getMotor('left wheel motor')

rightMotor = robot.getMotor('right wheel motor')

leftMotor.setPosition(float('inf'))

rightMotor.setPosition(float('inf'))

# set up the motor speeds at 10% of the MAX\_SPEED.

leftMotor.setVelocity(0.1 \* MAX\_SPEED)

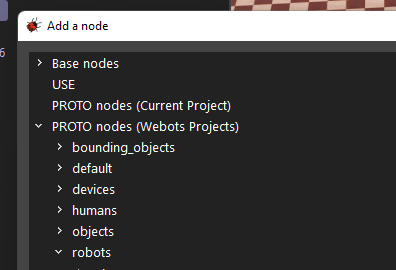
rightMotor.setVelocity(0.1 \* MAX\_SPEED)

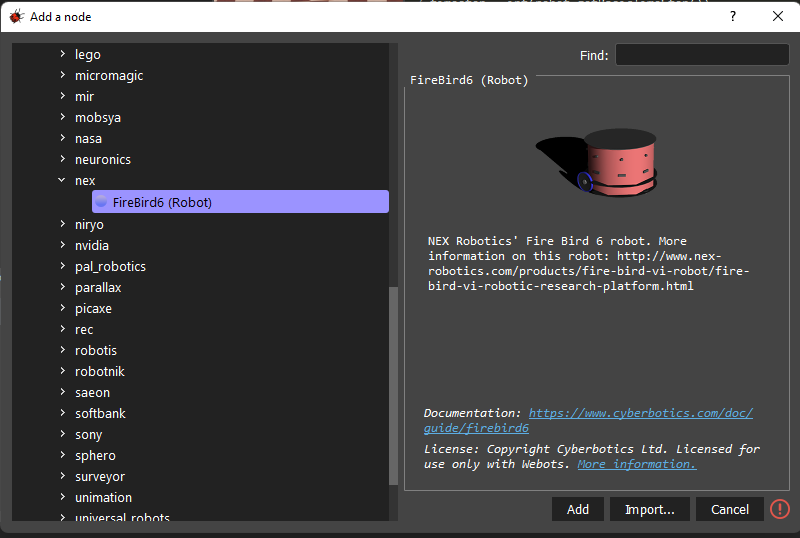
while robot.step(TIME\_STEP) != -1:

pass

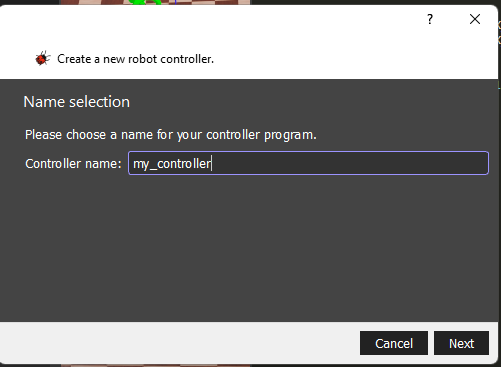
**Выполнение:**

Создание робота:





Создание нового контроллера:



Код контроллера:

**from controller import Robot, Motor**

**robot = Robot()**

**timestep = int(robot.getBasicTimeStep())**

**leftMotor = robot.getMotor('left wheel motor')**

**rightMotor = robot.getMotor('right wheel motor')**

**leftMotor.setPosition(float('inf'))**

**rightMotor.setPosition(float('inf'))**

**MAX\_SPEED = 15**

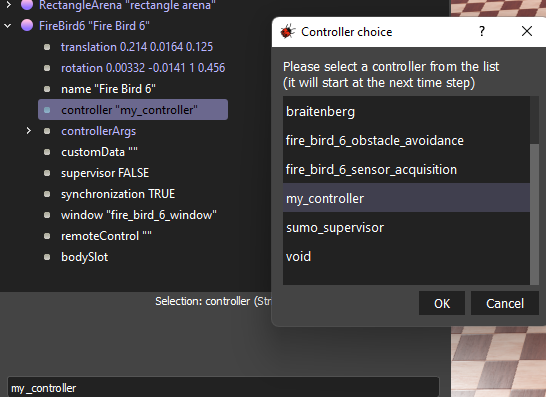
**leftMotor.setVelocity(0.1 \* (MAX\_SPEED/2))**

**rightMotor.setVelocity(0.1 \* MAX\_SPEED)**

**while robot.step(timestep) != -1:**

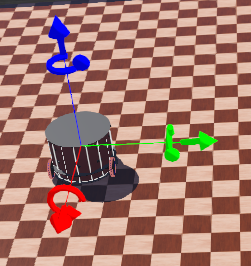
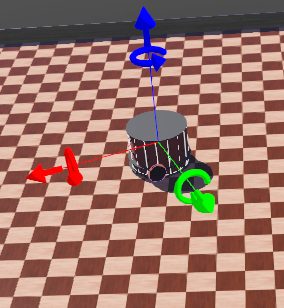
**pass**

подключение контроллера:



**Результат:**

Робот бесконечно едет по кругу



Вывод: Мы ознакомились с программой WeBots и создали простейшего робота