Санкт-Петербургский Государственный Университет
Математико-механический факультет

Кафедра системного программирования

Трёхмерное имитационное моделирование в QReal:Robots

курсовая работа студента 371 группы
Копытова Дмитрия Сергеевича

Научный руководитель Мордвинов Д.А.

Санкт-Петербург
2014

Оглавление

- Введение
  - Постановка задачи
- Обзор существующих решений
  - Breve
  - Gazebo
  - Simbad 3D Robot Simulator
  - LpzRobots
  - MORSE (the Modular OpenRobots Simulation Engine)
  - Выбор симулятора
- Реализация
  - Портирование симулятора Morse
  - Интеграция Morse с QReal:Robots
  - Генерация скриптов
  - Документация
- Заключение
  - Результаты
  - Дальнейшее развитие
- Список литературы

Введение

На кафедре системного программирования СПбГУ в течение нескольких лет разрабатывается среда разработки предметно-ориентированных визуальных решений QReal [17]. На базе неё создана система визуального программирования роботов QReal:Robots [18]. Среда предназначена для обучения основам программирования и кибернетики. Среда позволяет создавать графические программы для роботов и исполнять эти программы прямо на компьютере, посылая команды роботу через Bluetooth или USB-интерфейс, а так же генерировать по диаграмм код на Си-образном языке и загружать его для исполнения на робота. Для людей, не имеющих собственного робототехнического конструктора, но желающих так же наглядно представлять результаты работы своих программ, присутствует возможность двухмерного имитационного моделирования. Однако она не может полностью отобразить взаимодействие робота с окружающим миром. Поэтому было решено осуществить поддержку трёхмерного имитационного моделирования.

Постановка задачи

Целью данной курсовой работы было осуществить поддержку трёхмерного имитационного моделирования в среде QReal:Robots. В связи с этим были поставлены следующие задачи:

- исследовать существующие роботические 3D симуляторы;
- на основе полученных данных выбрать наиболее подходящий симулятор;
- выбранный симулятор интегрировать с QReal:Robots, если же не удастся найти подходящий симулятор, то разработать свой;
- запустить уже имеющиеся примеры программ на трёхмерной имитационной модели для проверки корректности её работы;
- разработать пользовательскую документацию и документацию для разработчиков.

Обзор существующих решений

На данный момент существует большое количество роботических симуляторов. Среди них имеют как коммерческие, так и бесплатные с открытым исходным кодом. Выбор же осуществлялся из множества бесплатных с открытым исходным кодом. Критерия для выбора два: совместимость API симулятора с нашими потребностями (некоторые симуляторы могут не подойти из-за своей бедной функциональности – не умеют делать всё, что нам нужно, либо их проблемно интегрировать с QReal:Robots) и поддержка платформы Windows. Ниже представлен список симуляторов, которые были рассмотрены в рамках данной работы:

- Breve [3]
- Gazebo [6]
- Simbad 3D Robot Simulator [21]
- LpzRobots [7]
- OpenSim [15]
- Morse [8]
- Moby [9]
- SimRobot [23]
- OpenHPR3 [13]

Далее рассмотрим поподробнее некоторые из этих симуляторов.

Breve

Бесплатный программный пакет, позволяющий строить 3D симуляции децентрализованных систем и искусственной жизни. Пользователь может задавать поведение мультиагентных систем в трехмерном мире и наблюдать, как они взаимодействуют. Breve включает в себя физическую симуляцию и обнаружение столкновений. Для визуализации используется собственный движок, использующий OpenGL. Breve доступен для Mac OS, Linux и Windows.

Данный симулятор имеет всего одно достоинство - он поддерживает платформу Windows. Из недостатков можно выделить следующие:

- отсутствие готовых сенсоров (имеется разве что обнаружение столкновений);
- нет готовых роботов;
- слишком примитивное графическое исполнение;
- добавление новых компонент, будь то робот или сенсор, затруднено;
- проект не находится в активной разработке, либо вообще заброшен (последняя активность в проекте датируется 2008 годом).

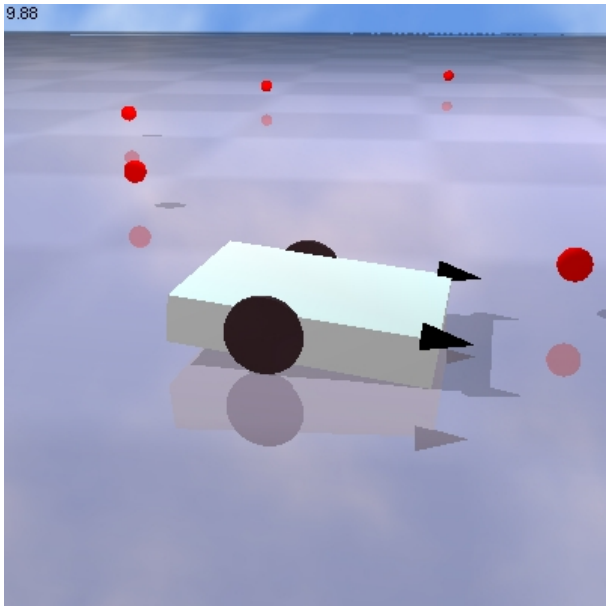

рис. 1. Breve.Физическая симуляция тележки

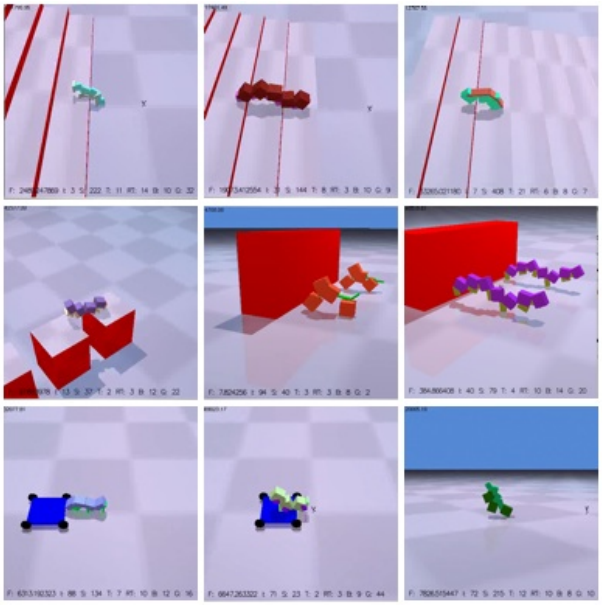

рис. 2. Breve. Эволюция морфологии и поведения.

Gazebo

Симулятор поддерживаемый Unix-подобными системами. Для визуализации использует графический движок OGRE [12]. Для эмуляции физики использует следующие физические движки: ODE [11], Bullet [4], Simbody [22] и DART [5]. Gazebo позволяет симулировать как популяцию роботов, так и одного конкретного робота, но с более высокой точностью. Имеет в наличии множество стандартных сенсоров, таких как сонар, контактные сенсоры, 2D/3D камеры, Kinect и др. Так же имеются уже встроенные модели роботов: Pioneer2DX, Pioneer2AT, SegwayRMP и др. Кроме того, есть возможность самостоятельно сделать модель нужного робота с помощью SDF (The Robot Modeling Language) [20]. Имеется графический интерфейс, позволяющий управлять миром и объектами в нём. Так же для взаимодействия с миром и для управления роботом можно использовать фреймворк ROS [19].

К достоинствам рассмотренного симулятора можно отнести следующие:

- большой набор готовых компонент (роботы, сенсоры);
- имеется возможность добавления своих компонент;
- хорошая документированность.

Недостаток всего один - официально не поддерживается платформа Windows.

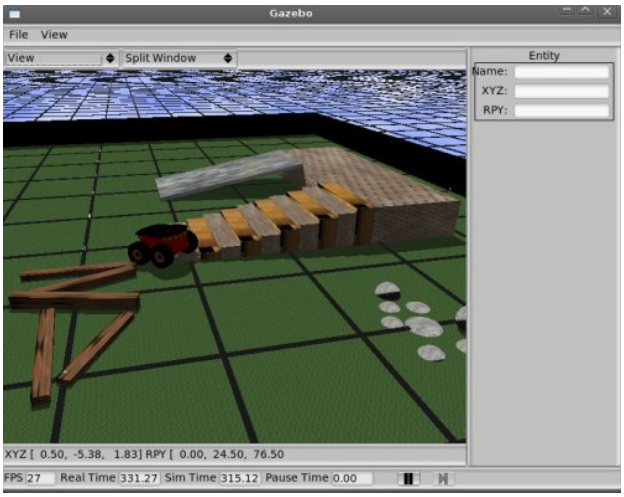

рис. 3. Gazebo. Пример среды исполнения.

рис. 4. Gazebo. Робот Atlas и внедорожник.

Simbad 3D Robot Simulator

Роботический симулятор для научных и учебных целей. Позволяет создавать собственный роботический контроллер, изменять среду и использовать доступные сенсоры. Доступны следующие сенсоры: сонар, инфракрасный сенсор, сенсор касания, цветная моноскопическая камера. Симулятор доступен для Mac OS, Windows и некоторых дистрибутивов Linux.

Из плюсов данного симулятора только поддержка платформы Windows. Недостатки следующее:

- симулятор написан полностью на языке Java, поставляется как библиотека, поэтому для использования необходим проект написанный на языке Java;
- мало готовых сенсоров и отсутствие готовых роботов;
- примитивное графическое представление;
- проект не находится в активной разработке, либо вообще заброшен.

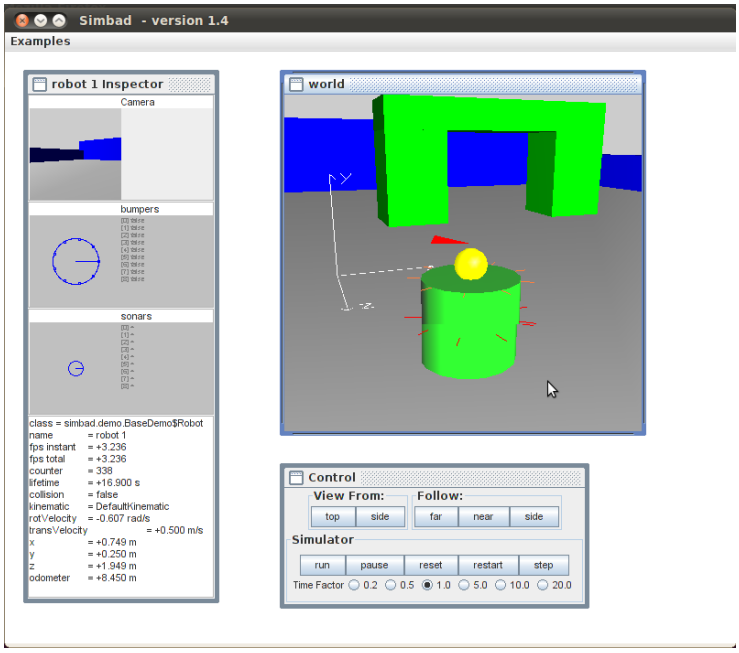

рис. 5. Symbad 3D. Интерфейс симулятора.

LpzRobots

LpzRobots - коллекция алгоритмов, симуляторов и инструментов. Включает в себя физический симулятор ode_robots, базирующийся на физическом движке ODE (Open Dynamic Engine). Для визуализации используется библиотека OpenSceneGraph [14]. Симулятор идёт вместе с готовыми компонентами, одними из которых являются готовые роботы (двух и четырёхколёсные тележки и др.) и сенсоры (сенсор касания, лазерный сенсор, инфракрасный сенсор, камера и др.). Поддерживаются платформы Mac OS и Linux.

К достоинствам данного симулятора можно отнести:

- большое количество готовых компонент (роботы, сенсоры)
- возможность построения собственного робота из готовых компонент;

Недостатки:

- нет поддержки платформы Windows;
- недостаточно полная документированность.

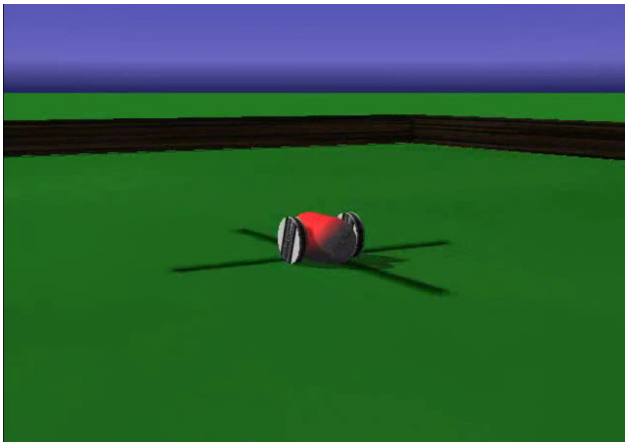

рис. 6. LpzRobots. Двухколёсный робот.

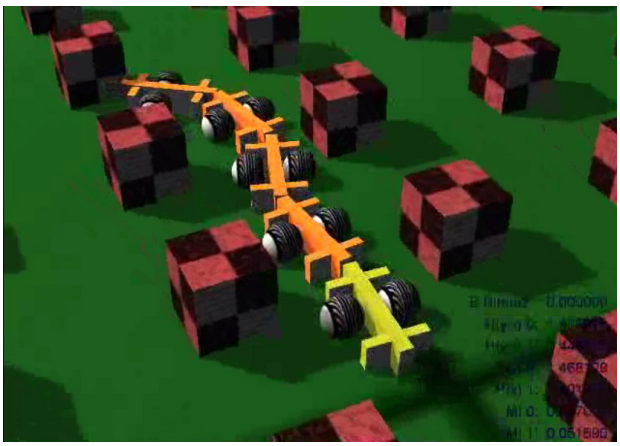

рис. 7. LpzRobots.

MORSE (the Modular OpenRobots Simulation Engine)

Morse - общий симулятор для академической робототехники. Сосредоточен на реалистичной 3D симуляции малых и больших сцен (в помещении или на открытом воздухе) с одним или десятками роботов. Процесс может полностью контролироваться через командную строку, либо с использованием сторонних фрейворков. Morse поддерживает 4 таких: ROS, YARP [24], Pocolibs [16] и MOOS [10]. Сцены генерируются из скриптов на Python. Базируется на игровом движке Blender Game Engine [2], который содержит в себе уже средства для визуализации, эмулирования физики, а также ряд уже готовых сенсоров.

рис. 8. Morse. ATRV робот в замкнутом помещении.

рис.9. Morse. Segway RMP 400 в открытом пространстве.

Morse идёт вместе с готовыми компонентами, такими как:

- роботы и роботические платформы
  - Segway RMP 400 platform
  - Quadrotor
  - Quadrotor with dynamics
  - Pioneer 3-DX platform
  - Yamaha RMAX platform
  - …
- стандартные сенсоры
  - Accelerometer
  - Collision
  - Depth camera
  - GPS
  - Gyroscope
  - Kinect
  - Laser Scanner Sensors
  - Semantic camera

    - Video camera
    - …
- другие компоненты
    - освещение
    - силовые приводы
    - ...

Кроме этого он позволяет легко добавлять собственные компоненты.
Morse официально поддерживает только Linux.
К достоинства данного симулятора можно отнести:

- большое кол-во встроенных компонент (роботы, сенсоры);
- возможность контролировать процесс симуляции посредством командной строки, либо с использованием сторонних фреймворков;
- возможность добавления собственных компонент;
- хорошая документированность.

Недостаток только один - официально не поддерживается платформа Windows.

Выбор симулятора

Среди представленных симуляторов наиболее подходящими по своей функциональности и документированности оказались Gazebo и Morse. Оставшиеся из рассмотренных симуляторов не подошли из-за своей бедной функциональности, плохой документированности (или полного её отсутствия) или общей заброшенности.

Раннее уже были предприняты попытки [1] добавления трёхмерной имитационной модели в QReal:Robots. Тогда в качестве симулятора был выбран Gazebo и интеграция с QReal:Robots была неудачна.

Поэтому, несмотря на то, что симулятор Morse официально не поддерживает платформу Windows, было решено выбрать его. В связи с этим появляется новая задача - портирование симулятора Morse на платформу Windows.

Реализация

Портирование симулятора Morse

Симулятор Morse официально не поддерживает платформу Windows, поэтому перед интеграцией с QReal:Robots он был портирован. Чтобы облегчить процесс обновления симулятора до новой версии, был написан batch-файл для автоматизации процесса исправления. Он загружает исходный код нужной версии Morse из репозитория проекта, настраивает и собирает симулятор, после делает ряд изменений необходимых для его работы на платформе Windows.

Интеграция Morse с QReal:Robots

Для начала рассмотрим архитектуру плагина для роботов. В нём содержится несколько подпроектов, но вся работа будет содержаться в подпроекте robotsInterpreter. Он же в свою очередь состоит из следующих компонент:

- *Interpreter*: хранит в себе контекст вычислений, отвечает за работы с потоками интерпретируемой программы.
- *Thread*: представляет ветвь исполнения многопоточной программы, запоминает и подсвечивает текущий исполняемый блок.
- *Block*: представляет высокоуровневое действие над роботом (например, включить мотор по такому-то порту на такую-то мощность, подать звуковой сигнал), определяет поведение элемента на диаграмме. Каждому типу элемента на диаграмме соответствует свой подкласс класса Block.
- *RobotModel*: логическая модель робота, представляет сам робот и набор его частей. Отвечает за соединение с роботом, его инициализацию, предоставляет всем компонентам доступ к логическим частям робота.
- *RobotPart*: базовый класс, его подклассы представляют логические части робота (мотор, сенсор, блок управления). Отвечает за формирование команд для интерфейса связи и разбор ответов.
- *RobotCommunicationInterface*: интерфейс связи с роботом, отвечает за низкоуровневое общение с устройством (или с эмулятором), передаёт ему команды и получает ответ.

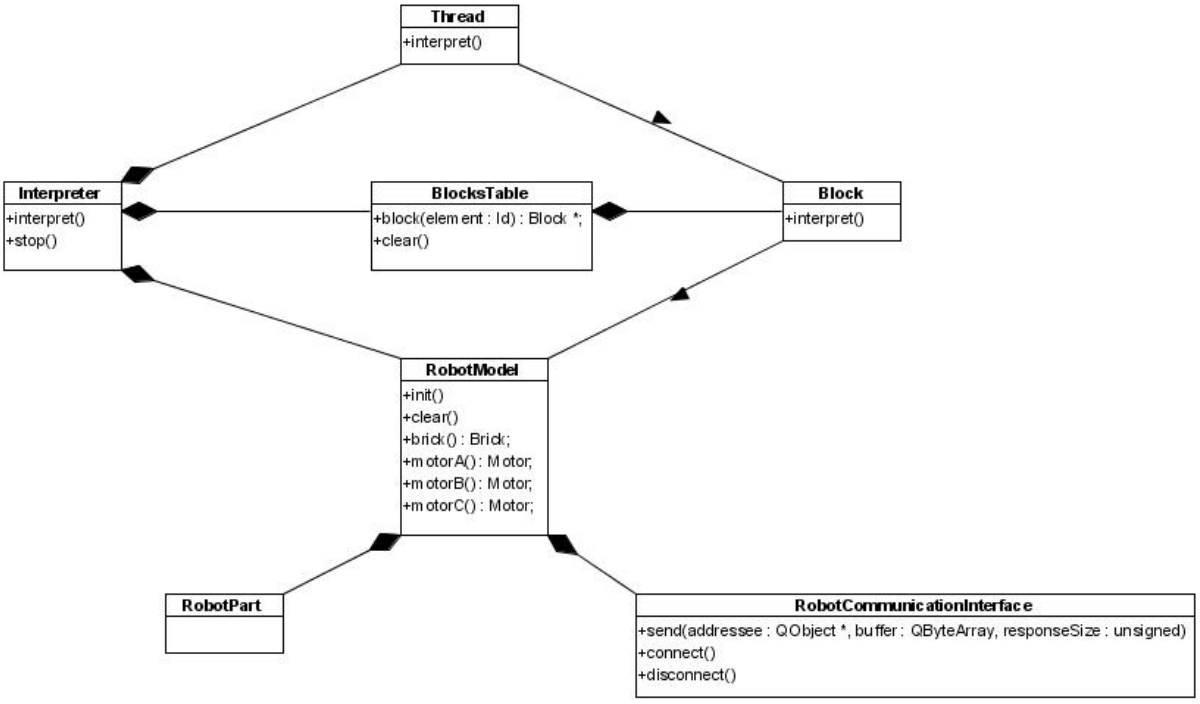

рис. 10. Общая структура robotsInterpreter.

Конкретные реализации логической модели выделены в отдельные классы, которые описывают различные виды исполнения программ. Для каждой части модели так же описываются подробные реализации. Все классы этих конкретных реализаций наследуются от соответствующего общего абстрактного класса, предоставляющий для логической модели интерфейс работы с ними или её частям, которые содержат экземпляр этого класса в качестве поля. Общая схема представлена на рис. 11.

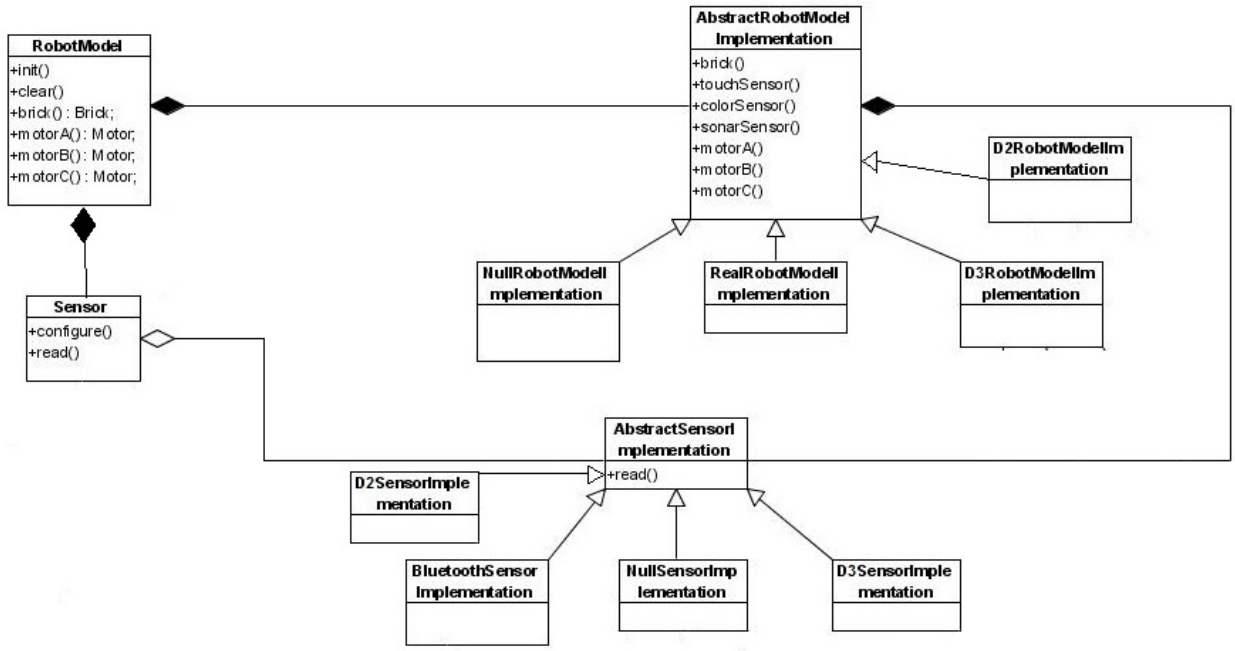

рис. 11.

Таким образом, для поддержки трёхмерного имитационного моделирования в QReal:Robots необходимо реализовать класс D3RobotModelImplementation, от которого в свою очередь будут наследоваться конкретные классы, соответствующие определённому типу робота. Данный класс был реализован, кроме этого был реализован класс, соответствующий роботу Lego Mindstorms NXT 2.0.

Генерация скриптов

На основе готового фреймворка для создания генераторов был реализован генератор скриптов на языке Python, необходимых для запуска и работы симулятора. Их всего два: build-скрипт и исполняемый скрипт. Build-скрипт нужен для инициализации робота, сенсоров и окружения, в котором будет происходить симуляция. Исполняемый же скрипт генерируется по диаграмме поведения робота и контролирует процесс симуляции.

Так как в симуляторе информация, поступающая от сенсоров, имеет другой формат по сравнению с тем, который предполагается в графических программах среды QReal:Robots, были реализованы дополнительные функции для конвертирования поступающей информации с сенсоров.

На данный момент подержано 4 сенсора: сенсор касания, сонар, сенсор цвета и сенсор света.

Сенсор касания реализован на базе сенсора коллизии. В отличии от обычного сенсора касания, который определяет столкновение с чем угодно, сенсор коллизии может так же определять столкновение только с теми объектами, которые имеют некоторое заранее оговорённое свойство. Какой-то особого конвертирования информации здесь не нужно (в обоих случаях получаемые значения это true/false - есть или нет касания), но для удобства при генерации кода, была выделена отдельная функция.

В качестве основы для сонара взят сканирующий лазерный сенсор. Он представляет собой дугу с множеством лучей, угол между соседними лучами одинаковый. В среде QReal:Robots предполагается, что сонар возвращает расстояние до ближайшего объекта, но лазерный сенсор в симуляторе Morse, возвращает два списка: список расстояний и список точек. Первый содержит расстояние до ближайшего объекта, достигнутого каждым лучом, если такового не имеется, то в качестве значения стоит максимальная дальность действия лазера. Второй список содержит координаты точки на ближайшем объекте, найденном каждым лучом, в противном случае возвращается (0, 0, 0). В связи с этим, была реализована функция, которая используя первый список (список расстояний) и возвращает расстояние до ближайшего объекта.

Последние сенсоры - сенсор цвета и сенсор света были реализованы на основе сенсора “камера”. Это сенсор при каждом запросе выдаёт RGBA изображение. Сенсор же цвета выдаёт либо распознанный цвет, либо его интенсивность, а сенсор света яркость. Для сенсора света, по аналогии с тем как это было реализовано в двумерной имитационной модели, был реализована функция, которая возвращает эту самую яркость. С сенсором цвета всё немного сложнее: в отличие от двумерной

имитационной модели, где есть только “чистые” цвета (белый, чёрный, зелёный, красный синий), в трёхмерной имитационной модели появляются различные оттенки этих цветов (из-за освещения, теней, текстур и прочего). Поэтому способ, реализованный в двумерной имитационной модели, не совсем подходит. Несмотря на это, на первое время реализован был всё-таки он. В дальнейшем его необходимо заменить на более корректный.

Документация

Так как симулятор Morse использует пакет Blender для визуализации, создания сред симуляции, моделей роботов и прочего, то необходима документация по тому, как пользоваться пакетом Blender. Для этого был создан небольшой вводный курс, который, на примере уже имеющихся готовых графических программ, показывает как строить под них среду исполнения и заодно основные особенности работы с пакетом Blender.

Кроме пользовательской была создана документация для разработчиков, кратко описывающая сам симулятор Morse, его компоненты и особенности работы с ним.

Заключение

Результаты

В рамках курсовой работы были рассмотрены различные роботические симуляторы. Выбранный симулятор был интегрирован с QReal:Robots. Была реализована архитектура поддержки трёхмерной симуляции в QReal:Robots. Были поддержаны уже имеющиеся четыре сенсора: сенсор касания, сонар, сенсор цвета и сенсор света. Разработана пользовательская документация и документация по поддержке симулятора Morse в среде QReal:Robots.

На данный момент по имеющемуся набору сенсоров строится робот, в качестве модели робота взята стандартная модель iRobot ATRV, в качестве среды для симуляции пока что берётся тестовая среда. По диаграмме поведения робота генерируется управляющий скрипт и запускается игровым движком симулятора.

Дальнейшее развитие

Текущие цели:

- добавление более корректной функции распознавания цвета;
- добавление модели Lego Mindstorms NXT 2.0, вместо стандартной iRobot ATRV;
- добавление возможность выбора среды исполнения;
- добавление поддержки интерпретации графических программ на трёхмерной имитационной модели;
- добавление поддержки других роботов и сенсоров среды QReal:Robots.

Список литературы

1. Павлов С.Н. , курсовая работа 3 курса “Трехмерная модель робота в QReal:Robots”, 2012 год
2. Blender Game Engine, http://www.blender.org/
3. Breve, http://spiderland.org/
4. Bullet, http://bulletphysics.org/
5. DART, http://dartsim.github.io/
6. Gazebo, http://gazebosim.org/
7. LpzRobots, http://robot.informatik.uni-leipzig.de/software/
8. Morse, http://www.openrobots.org/morse/doc/stable/morse.html
9. Moby, http://physsim.sourceforge.net/
10. MOOS, http://www.robots.ox.ac.uk/~mobile/MOOS/wiki/pmwiki.php
11. ODE, http://www.ode.org/
12. OGRE, http://www.ogre3d.org/
13. OpenHPR3, http://www.openrtp.jp/openhrp3/en/index.html
14. OpenSceneGraph, http://www.openscenegraph.org/
15. OpenSim, http://opensimulator.sourceforge.net/
16. Pocolibs, https://www.openrobots.org/wiki/pocolibs
17. QReal, http://qreal.ru/
18. QReal:Robots, http://robots.qreal.ru/
19. ROS, http://www.ros.org/
20. SDF, http://gazebosim.org/sdf.html
21. Simbad 3D Robot Simulator, http://simbad.sourceforge.net/
22. Simbody, https://simtk.org/home/simbody/
23. SimRobot, http://www.informatik.uni-bremen.de/simrobot/index_e.htm
24. YARP, http://eris.liralab.it/yarp/