Столыпинский вестник

Научная статья

Original article

УДК 004.921

РАЗРАБОТКА СРЕДЫ ПРОГРАММИРОВАНИЯ РОБОТОВ С РЕЖИМОМ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

DESIGN OF INTEGRATED DEVELOPMENT ENVIRONMENT FOR ROBOTS WITH SIMULATION MODE

Орехова София Михайловна, студент-бакалавр, Кубанский государственных технологический университет, г. Краснодар

Янаева Марина Викторовна, кандидат технических наук, доцент, Кубанский государственный технологический университет, г Краснодар

Orekhova Sofiia Mikhailovna, B.S. in Engineering, Kuban State Technology University

Yanaeva Marina Viktorovna, Candidate of Technical Sciences, Associate Professor, Kuban State Technological University, Krasnodar

Аннотация

В ходе развития информационных технологий и промышленной инженерии в 21 веке в общее пользование начали появляться такие помощники, как роботы. Само понимание «робот» пришло определенно недавно и происходит от чешского языка. Что же представляет из себя робот? Робот – это некоторое автоматическое устройство, которое предназначено для выполнения определённой операции и действующее по заранее заложенной программе. В текущее время это может быть, как и физическое воплощение полноценного

робота, так и автономно действующая программа. Так же, для обучения такой дисциплины как «робототехника» были применены такие программы, как среда программирования виртуального робота, где основным средством при разработке логики проекта является визуальное программирование.

Подобная среда программирования направлена на введение в развивающуюся робототехнику и начальным обучением программирования. Разработка программы для виртуального робота одна из важнейших частей, которая помогает развивать аналитическое мышление.

Определённо, после введения такого понятия, следует следующий вопрос – что из себя представляет подобная среда? В данной статье будет раскрыто само понимание виртуального робота и каким образом его можно представить в проекте. Так же, отдельно будет рассматриваться развитие визуального языка в данной разработке на примере самого старшего и наиболее развитого графического языка G, иначе LabView.

Annotation

In the course of development information technology and industrial engineering in the 21st century, helpers such as robots began to appear in the public domain. The very understanding of the «robot» came definitely recently and comes from the Czech language. What is a robot? A robot is some kind of automatic device that is designed to perform a specific operation and acts on a predetermined program. In the current time, this can be both the physical implementation of a full-fledged robot and an autonomous program. Also, for the training of such discipline as «robotics» such programs as virtual robot programming environment were applied, where the main tool in the development of project logic is visual programming.

A similar programming environment is aimed at introducing robotics and initial programming training to the developing world. Developing a virtual robot program is one of the most important parts that helps to develop analytical thinking.

Surely, after the introduction of such a concept, the next question is, what is such an environment? This article will reveal the very understanding of the virtual robot and how it can be presented in the project. Also, the development of visual

language in this development will be considered separately on the example of the oldest and most developed graphical language G, otherwise LabView.

Ключевые слова: робототехника, графический язык программирования, разработка виртуальной среды, разработка графического движка, применение среды программирования в обучении

Keywords: robotics, graphical programming language, development of a virtual environment, development of a graphics engine, integrated development environment in education

Каждое обучение начинается с определенных «введений» в данную область. Такая популярная на данных момент дисциплина «робототехника», которая полноценно развивается в промышленных масштабах в наше время, тоже имеет курсы по введению. В данных курсах принято работать с облегченными конструкторами, и соответственной средой программирования робота на основе диаграмм. Для более старшего поколения были введены такие платы, как Arduino с собственным C-подобным языком, но любая из перечисленных моделей обучения не имеет среды программирования с имитационным режимом. Если на просторах интернета созданные трехмерные сцены для программирования роботов и можно найти, то они не все имеют полноценную интеграцию с каждой развивающейся прошивкой. Таким образом, поддерживать долгосрочность данных продуктов не представляется возможным, что влияет как на процесс понимания данной системы.

Для чего же нужна имитационная среда? Подобная среда поможет в обучении дистанционно или может использоваться в школьных курсах информатики для обучения «азам» программирования. Введение в понимания циклов, потоков и параллельного программирования.

Для полноценного понимания дальнейшей темы данной статьи, следует углубиться в тему понимания виртуального робота, и обозначить его отличие от «программной» версии. Сама тема состоит из нескольких ответвлений: это роль робота в имитационной среде и каким образом его можно представить в

проекте, так сама среда программирования – как она устроена, и какие функции она будет выполнять.

Виртуальный робот представляет модель в имитационной среде, то есть при обучении программированию команды передаются не физическому образцу, а его виртуальному «брату». Из подобной среды можно выделить основные плюсы – менее затратное введение в программирование моделей для выполнения команд, возможность дистанционного обучения малых групп дисциплины «робототехника». Имитационная среда представляет из себя двумерное пространство. Это обусловлено многими факторами, начиная от менее ресурсозатратного программного обеспечения до легкости в представлении возможных препятствий для виртуального робота. Каким же образом можно интегрировать робота в программу – на этот вопрос уже следует обратиться к разработке самого программного обеспечения.

Первое с чего необходимо начинать любой симулятор – это выбор физики или выбор физического ядра симулятора. Поскольку программа реализуется в наиболее детализированном языке C++, выбор пал на две альтернативы: разработка собственного ядра или использование в доработке наиболее детализированного. В выбранном игровом движке – Box2D не поддерживается следующая особенность, которую было необходимо добавить собственные доработки, так как виртуализация робота и проверка запрограммированных команд заключается в виде сверху, так как только при таком расположении камеры возможно расставить на карте препятствия, но и видеть каким образом робот будет обходить их. В доработке обсуждается следующий вопрос: каким образом можно создать физику модели с видом сверху? Обычно, модели с видом сверху моделируются в мире невесомости, представленном одним корпусом для шасси и четырьмя отдельными корпусами для колес. В зависимости от того, насколько реалистичной требуется симуляция, может быть достаточно просто использовать один корпус для шасси и не беспокоиться о наличии отдельных колес. Проблема заключается в следующем – необходимости предотвращения движения тела

по локальной оси. Основная процедура состоит в том, чтобы найти текущую боковую скорость тела и применить импульс, который компенсирует эту скорость. В проекте легче начинать с одного тела.

В начале данной работы необходимо компенсировать боковую скорость. Компенсация боковой скорости необходима для приведения вектора текущей скорости к нормали для направления самой шины (под шиной понимается направление модели колес) – рисунок 1.

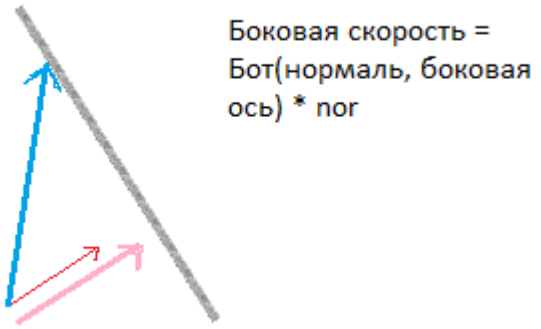

Рисунок 1 – Распределение боковой скорости

При движении шины, которая повернута, проецирование синего вектора к красному, который перпендикулярно движется к нормали, необходимо смещением. Для того, чтобы шина далее не продолжала крутиться вокруг своей оси необходимо обратиться к теме о движении с постоянной скоростью, где необходимо ввести расчёт импульса:

```
void updateFriction() {
    b2Vec2 impulse = m_body->GetMass() * -getLateralVelocity();
    m_body->ApplyLinearImpulse( impulse, m_body->GetWorldCenter() );
 }
```

После настройки передвижения объекта, необходимо продумать условие поверхности, по которой должен перемещаться объект. Несмотря на то, что ранее упоминалось, что подобное моделирование работает в мире невесомости, наложение объектов друг на друга не отменяется.

В данном проекте, установка гравитации к нулю уменьшит эффект реальной симуляции модели, поэтому к данной проблеме так же стоит

подключить имитацию кулоновского трения или закон Амонтона-Кулона, который определяется в следующем: $F = \mu N$, что интерпретируется в следующем: сила трения скольжения одного тела по поверхности другого тела (опоры) направлена тангенциально к общей границе между двумя телами в сторону, противоположную перемещению. После данных настроек, до формирования графики – поле выглядит следующим образом – рисунок 2:

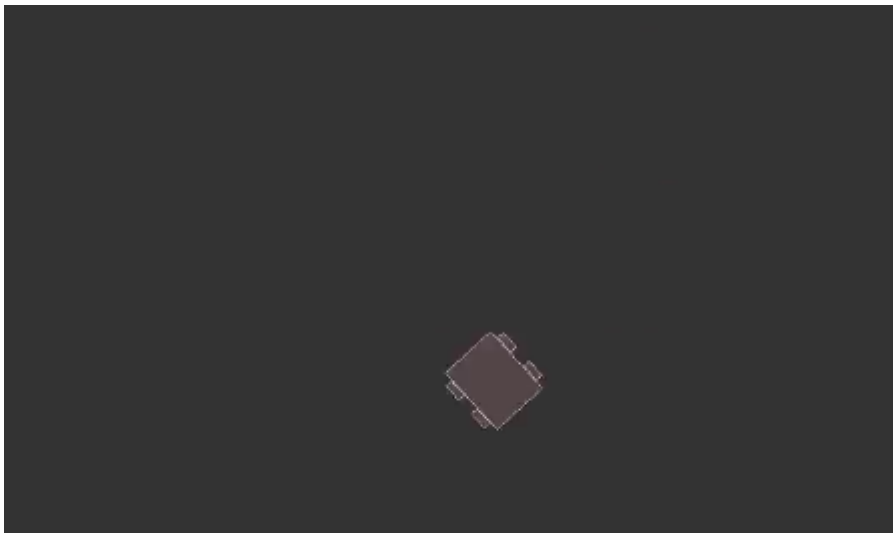

Рисунок 2 – Модель на обозреваемом поле

Данный бот не оснащен потенциальными датчиками, которые будут использоваться в последствии.

В малой версии проекта необходимо определить основные датчики, которые будут определять расстояние до объекта, это так же моделируется на основе Box2D, так как данная библиотека имеет встроенные классы: b2RayCastInput и b2RayCastOuput, которые проводят линию между двумя точками и возвращают диапазон для любых объектов, которые пересекает линия – рисунок 3.

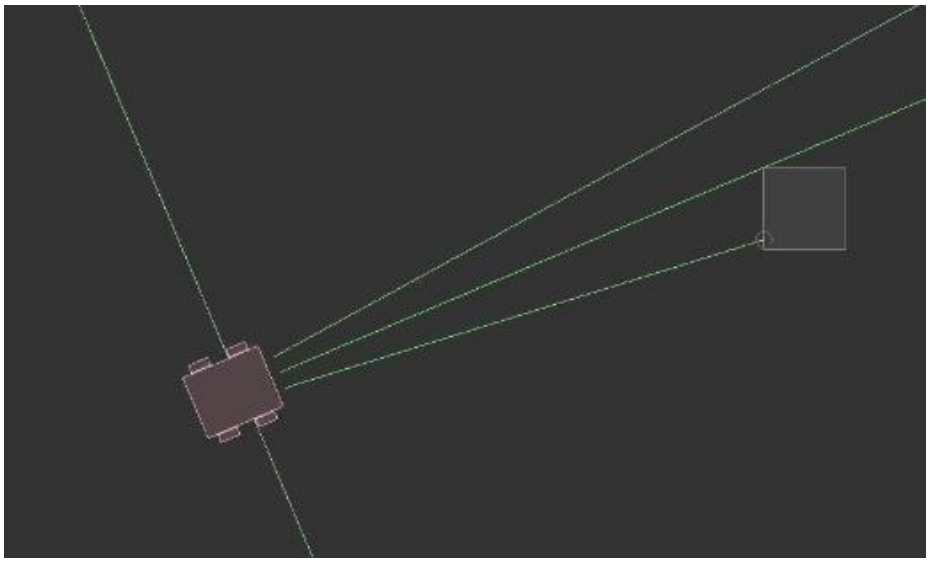

Рисунок 3 – Внедрение датчиков к боту

Имитация линейного детектора менее проста. В реальном мире линейный детектор обычно представляет собой ИК-датчик, который обнаруживает излучение света от объектов, но Box2D не может моделировать на этом уровне, потому что в его библиотеке данный вопрос не предусмотрен. Для решения возникшей проблемы в isSensor (булево значение класса b2FixtureDef), который собирает контактные данные, значение true для линии, что делает ее «физически невидимой», но позволяет Box2D регистрировать столкновение. Затем в коде используется b2ContactListener для получения обратного вызова, когда происходит столкновение.

После моделирования основной цели – объект с видом сверху, необходимо ввести возможность программно отдавать команды для этого бота.

Был использован метод, который используется с 1986 года средой разработки виртуальных приборов – графический язык программирования. Плюсы такого языка заключаются в следующем, подобные языки, как и применяемый LabView, основан на архитектуре потоков, что это означает? Программа при таком подходе моделируется в виде орграфа потока, что напоминает диаграмму потоков данных. Что дает во время программирования естественное визуальное представление о будущей программе.

Визуальное программирование или же графическое, помогает лучше

понять основы разработки программы для системы без навыков программирования, таким образом, любой начинающий сможет с легкостью понять основы задачи программы и передачу потоков между моделью и кодом.

Поскольку при моделировании объектов ранее был использован Box2D, который ориентирован на архитектуру физических плат Arduino, то при разработке самой системы визуального программирования, библиотеки будут использоваться стандартные. Так, для примера можно ввести библиотеку сервомоторов, которые и применялись при создании модели ранее.

Визуально программа представляет следующее: на выбор выдается несколько действий, иначе подпрограмм, в которые уже подгружены библиотеки. Как и любая диаграмма потоков данных, так и в данном языке, необходимо обозначить старт, то есть подготовка робота к загрузке программы. Для этого в коде создается отдельный интерфейс, оповещающий о начале выполнения команд:

```
public interface IRobotAction
   {
      string GetActionType();
   }
```

Для каждого дальнейшего потока, подключается своя часть интерфейса, которая отвечает за любое выбранное действие робота. В данном случае – выполнение поворотов при определении препятствия, что рассматривалось ранее при разработке физики объекта.

Подведя итоги, можно выделить следующее: для того чтобы данный проект можно было использовать при обучении робототехники, помимо реалистичной симуляции, которая поможет объяснить обучающимся основные законы физики, которым подчиняется любой объект, необходима дружелюбная среда, которая поможет развить представление о передаваемых потоках в системе, и без труда задавать программы. Использование двумерного пространства при имитации среды робота позволяет наглядно выстроить препятствия и уже при создании программы – использовать данную

информацию для облегченного построения обхода объектов.

Литература

1. Box2D 2D physics engine for games URL: https://box2d.org/documentation/index.html
2. Никеров, В. А. Физика. Современный курс : учебник / В. А. Никеров. — 4-е изд. — Москва : Издательско-торговая корпорация «Дашков и К°», 2019. - 452 с. - ISBN 978-5-394-03392-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/1093441
3. Методическое пособие СВФУ по Трик Студио URL: https://yagu.s-vfu.ru/pluginfile.php/1075958/mod_resource/content/1/TRIKStudio_%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%B8%D1%87%D0%BA%D0%B0.pdf
4. Комков Н.И., Бондарева Н.Н. Перспективы и условия развития робототехники в России // МИР (Модернизация. Инновации. Развитие), 2016
5. Adobe Edge: April 2010 – Разработка физических игр с Adobe Flash Professional. Adobe. Оригинальный архив, Август 11, 2011. Проверка July 19, 2016.

Literature

1. Box2D 2D physics engine for games URL: https://box2d.org/documentation/index.html
2. V. A. Nikerov, Physics. Modern course: textbook / V. A. Nikerov. - 4th ed. - Moscow: Publishing and Trade Corporation «Dashkov and Co», 2019. - 452 p. - ISBN 978-5-394-03392-6. - Text : electronic. - URL: https://znanium.com/catalog/product/1093441
3. NEFU Methodological Manual for Trick Studio URL: https://yagu.s-vfu.ru/pluginfile.php/1075958/mod_resource/content/1/TRIKStudio_%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%B8%D1%87%D0%BA%D0%B0.pdf
4. Komkov N.I., Bondareva N.N. Prospects and conditions of development of

robotics in Russia // WORLD (Modernization. Innovations. Development), 2016

5. Adobe Edge: April 2010 – Developing physics-based games with Adobe Flash Professional. Adobe. Archived from the original on August 11, 2011. Retrieved July 19, 2016.

Для цитирования: Орехова С.М., Янаева М.В. РАЗРАБОТКА СРЕДЫ ПРОГРАММИРОВАНИЯ РОБОТОВ С РЕЖИМОМ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ // // Научный сетевой журнал «Столыпинский вестник», №4/2022.