Робототехническое онлайн-состязание: опыт и ресурсы

Robotic Online Competition: Experience and Resources

Получено 15.04.2022 Одобрено 08.06.2022 Опубликовано 24.06.2022 УДК 004.8 DOI: 10.12737/1998-0744-2022-10-3-13-22

ОВСЯННИКОВ А.Ю.,
методист по олимпиадной робототехнике центра подготовки, приема и развития студентов Университа Иннополис, с 2013 года тренер сборной Российской Федерации по спортивной робототехнике Всемирной олимпиады роботов, член национального экспертного совета Федерации спортивной и образовательной робототехники

e-mail: a.ovsyannikov@innopolis.ru

OVSYANNIKOV A.YU.,
Methodist in Olympiad Robotics at the Center for Training, Admission and Development of Innopolis University Students, since 2013 Coach of the Russian Federation Team in Sports Robotics at the World Robot Olympiad, Member of the National Expert Council of the Federation of Sports and Educational Robotics

e-mail: a.ovsyannikov@innopolis.ru

Аннотация

В статье представлен опыт подготовки и проведения робототехнического онлайн-состязания в виртуальной среде, который позволяет охватывать удаленных участников, в том числе из разных стран, на основе использования симулятора с управлением моделью робота на основе программирования. Участники состязания получают возможность творчески применить навык программирования в прикладной робототехническое среде для разных моделей виртуальных роботов и полигонов управления ими.

Ключевые слова: Олимпиада по робототехнике, виртуальная среда, программирование, моделирование.

Abstract

The article presents the experience of preparing and conducting an online robotic competition in a virtual environment. This approach allows you to cover remote participants in the competition, including those in different countries, based on the use of a robot model control simulator based on programming. Competitors get the opportunity to creatively apply programming skills in an applied robotic environment for various models of virtual robots and their control areas.

Keywords: robotics Olympiad, virtual environment, programming, modeling.

Для робототехнических конкурсов с акцентом на программирование стало возможным проводить онлайн-состязания с использованием специализированных программных сред – симуляторов, имитирующих и визуализирующих поведение робота и его окружение и позволяющих с помощью программирования управлять виртуальным роботом на компьютере. Примерами подобного подхода может быть профиль «Летающая робототехника» Национальной технологической олимпиады (использовался симулятор Gazebo, в котором имитировалось поведение коптера) или профиль «Манипуляционные ИРС» олимпиады Innopolis Open Robotics 2020 и 2021 сезонов (использовался симулятор CoppeliaSim). Например, для онлайн-тренировок и выполнения заданий организаторы предоставили участникам RoboCupJunior Rescue для состязания RoboCupJunior Rescue Simulation (Demonstration) в 2021 году и организаторы Brazilian Robotics Olympics (OBR) [1].

Проблематика использования симуляторов в робототехническом состязании

Несомненным преимуществом использования симулятора в робототехнических соревнованиях является возможность онлайн-участия команд и простота масштабирования на значительное количество участников. Онлайн-формат даёт возможность участвовать в состязании и тренироваться почти в любых условиях: дома, в образовательной организации, на площадке мероприятия, присутствуя в любой точке мира, – без очередей и помех извне. Команда соперника не сможет повлиять на полигон состязания (поле управления Роботом), не включит помеху (например, мощный радиоканал, влияющий на работу датчиков), не создаст отвлекающие эффекты, как это может произойти в условиях очного участия. Для онлайн-участия достаточно иметь компьютер с описанными в правилах состязания системными требованиями и доступ

к подготовленному организаторами инструментарию состязания (модель робота и окружения, библиотеки для написания программ). Простота масштабирования заключается в легкой установке компьютерного симулятора на большем количестве компьютеров.

Большинство симуляторов имеют бесплатные для использования в образовательных целях версии, которые подходят для решения соревновательных задач.

Главная сложность, с которой сталкиваются организаторы состязаний по робототехнике при переходе в симулятор, заключается в *корректном переносе модели робота и его окружения в симуляционную среду*. Робот должен предсказуемо и максимально правдоподобно вести себя в симуляторе:

- реагировать на команды управления и программу;
- обрабатывать столкновения с препятствиями;
- выдавать приближенные к действительным показания сенсоров (в том числе помехи, которые участникам необходимо обрабатывать с помощью программы).

За всеми этими параметрами подчас скрываются неочевидные свойства реального мира, поэтому создателям виртуальной модели приходится анализировать и указывать параметры материала модели, например, резинового или пластикового колеса, ее физические свойства (максимальный крутящий момент и допустимую скорость мотора), учитывать или имитировать возможные помехи датчиков и разряд батареи робота.

В некоторых соревнованиях можно пренебречь точностью и подробностями описания некоторых физических характеристик модели. Окончательное решение о точности предоставленной участникам состязания модели симулятора принимают организаторы, исходя из постановки задачи. Чем подробнее предоставляется описание модели робота и виртуального полигона в компьютерном симуляторе, тем проще и эффективнее будет переход участника состязания от реального робота к среде симулятора и обратно.

Естественно, при использовании симуляторов роботов общего назначения (вместо создания собственного «соревновательного» решения) существует необходимость ограничить свободу действий участника, ведь в большинстве из них доступны «волшебные» функции, недоступные в реальном мире: мгновенный перенос или создание нужных объектов из программы управления роботом, считывание параметров окружающего мира без датчиков, выход за заданные параметры скоростей и сил / моментов двигателей. Ограничить использование подобных функций можно путем предварительной подготовки промежуточных классов / библиотек, реализующих необходимый функционал и отсекающих «волшебные» команды.

Рассмотрим примеры организации состязания для виртуального финала профиля «Манипуляционные ИРС» олимпиады Innopolis Open Robotics в 2020 и 2021 годах и подготовки онлайн проектного Робот-тура международного Кубка информатики ЮНИОР ISIJ, используя авторские разработки – вебинары [2] и уроки [3] по работе с симулятором [4].

Выбор симулятора для онлайн-состязания

В качестве основы для проведения виртуальных туров был выбран симулятор CoppeliaSim, доступный бесплатно для использования в образовательных некоммерческих целях. Симулятор является наследником V-REP и развивается уже много лет, предлагает несколько физических движков (physics engines) с разной степенью детализации тех или иных физических явлений (физику столкновений и реакций, потоков и частиц, пружин и приводов и подобных), множество интерфейсов и развитую систему команд для написания управляющих программ на разных языках программирования.

Внешний вид окна симулятора с открытой сценой представлен на рис. 1.

При организации онлайн проектного Робот-тура международного Кубка информатики ЮНИОР ISIJ используется язык программирован C++, в остальных проводимых соревнованиях выбор языка осуществляется участниками, организаторы лишь должны обеспечить наибольший их охват и подготовить для них инструментарий.

Языки программирования C/C++ поддерживаются в CoppeliaSim на нескольких уров-

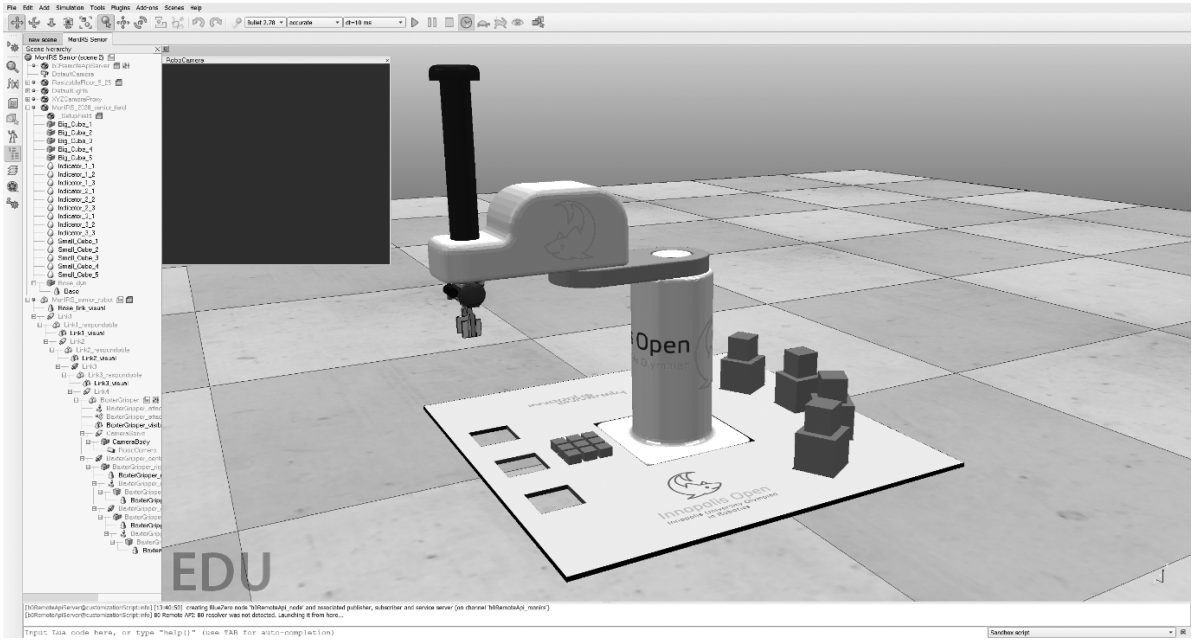

Рис. 1. Пример интерфейса симулятора с загруженной моделью робота-манипулятора

	Embedded script	Add-on / sandbox script	Plugin	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	Yes	Yes	Yes
Difficulty to implement	Easiest	Easiest	Relatively easy	Easy	Relatively easy	Easy
Supported programming language	Lua, Python	Lua, Python	C/C++	C/C++, Python, Java, JavaScript, Matlab, Octave	Any [1]	Any

Рис. 2. Поддерживаемые симулятором языки программирования

нях: написание плагинов для симулятора, использование в пользовательской программе Legacy remote API, использование ROS/ROS2 нод и использование ZeroMQ remote API (рис. 2).

Можно считать симулятор CoppeliaSim оптимальным выбором, обеспечивающим простое бесплатное решение для каждого участника, гибкий функционал и широкие возможности для организаторов. Широкий выбор языков программирования и библиотек подключения внешних программ позволяет проводить симуляцию на локальном компьютере, удаленно по сети, а кроссплатформенность и инструменты запуска из командной строки дают возможность использовать инструменты виртуализации и запускать симуляцию в контейнерах Docker.

Этапы подготовки сцены в симуляторе организаторами состязания

Описываемые робототехнические соревнования и конкурсы предполагают создание модели и программы робота для выполнения какого-либо набора действий с физическими объектами: перемещение робота, поиск игровых объектов, определение их параметров и сортировка или перемещение нужных. В редких случаях объекты требуется не перемещать, а оставлять в исходном положении (например, объезжать препятствия, в поисках выхода

из лабиринта или кратчайшего пути следования робота из одной зоны в другую).

Как правило, в заданиях соревнований авторы закладывают необходимый для изучения участниками теоретический аппарат, такой как построение карты мобильным роботом во время его движения [5], решение прямой и обратной задач кинематики манипулятора, описание карты с помощью графа и поиск в нем кратчайшего пути. Именно навыки решения этих задач организаторы соревнований стремятся проверить у участников, но, зачастую, участники не могут их проявить в должной мере, так как не обладают инженерными навыками создания роботов. Их устройства не выдерживают длительных нагрузок или перевозки в багаже от места подготовки к месту проведения соревнований, различные технические проблемы с питанием или датчиками делают робота беспомощным при решении основной задачи соревнований. Именно симулятор позволяет отсечь подобные проблемы, избавляя участников от необходимости решать инженерные задачи и сконцентрироваться на алгоритмах поведения роботов [6].

Стоит отдельно отметить, что некоторые конкурсы ориентированы на изготовление робота и комплексное решение задачи, предполагают командное участие и распределение ролей в них. Так, кто-то из участников прорабатывает механику и конструкцию робота, другой участник – электронику, третий – алгоритмы и программную часть. В этом случае идеология применения симуляторов может быть иной. Но в индивидуальных соревнованиях и олимпиадах по информатике участникам сложно учесть все эти аспекты, и организаторы могут заранее подготовить базовую модель робота, единую для всех.

Учитывая описанные ранее особенности конкурсов, получаем **список элементов** сцены, которые необходимо заранее подготовить в симуляторе, для организации их виртуального этапа:

- базовая модель робота;
- игровые объекты;
- окружение робота (полигон);
- система жеребьевки для подготовки полигона перед запуском;
- система контроля выполнения задания и подсчета результатов (программа-судья, «чекер»).

Первые четыре пункта входят в инструментарий участников и выдаются им на начальных этапах подготовки. Последний пункт относится к сцене, но используется организаторами уже во время финала. Кроме программы-судьи есть еще скрипт для автоматического запуска программ (решений) участников, но он не относится к сцене и симулятору, а лишь запускает его с нужными параметрами.

Базовую модель робота, игровые элементы и полигон удобнее создавать во внешних редакторах – системах автоматического проектирования (САПР), таких как KOMPAS 3D, Fusion 360, AutoCAD, SolidWorks и подобных. Переносить в симулятор можно как базовую геометрию объектов (формат файлов .obj, .dxf, .stl), так и геометрию с описание кинематических связей (формат файла .urdf). Отличия этих способов в том, что при использовании URDF (Unified Robot Description Format) автоматически создаются кинематические пары – связи двух или нескольких звеньев робота (links). Такими кинематическими парами могут быть любые приводы робота: ведущие колеса робота, схват, подвес камеры.

Первый этап подготовки объектов сцены, выполняемый сразу после переноса геометрии, заключается в оптимизации динамических моделей каждого звена робота. Симулятор позволяет визуально приятно отображать различные объекты, но при расчетах перемещений, столкновений, взаимодействий используются упрощенные динамические поверхности (dynamic shapes). Чем проще эти модели, тем быстрее происходит расчет динамики на каждом шаге симуляции (рис. 3).

В иерархии модели робота динамическая и визуальная поверхности (объекты) отделяются, им задаются отличающиеся параметры. Визуальной части отключается любой расчет динами и столкновений, она жестко привязывается к динамической. У динамической же настраиваются как параметры столкновений и динамики (модель может участвовать в столкновениях объектов на сцене, но быть зафиксированной неподвижно, жестко закрепленной на полигоне), так и масса, момент

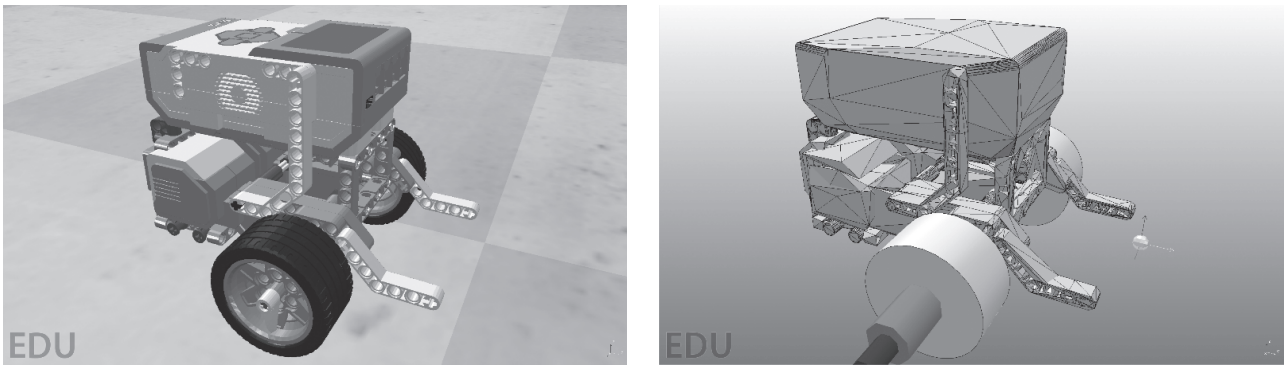

Рис. 3. Сравнение визуальной и динамической поверхностей мобильного робота Lego Mindstorms EV3

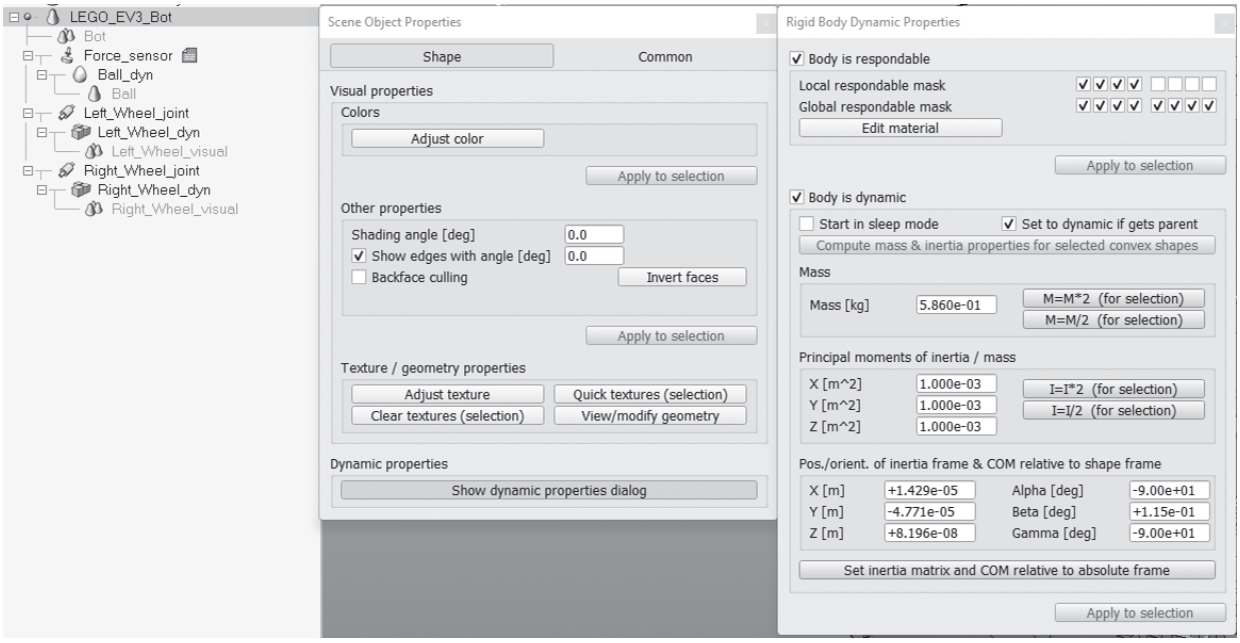

Рис. 4. Окна настройки параметров и динамических свойств модели робота Lego Mindstorms EV3

инерции и другие физические параметры. На рис. 4 приведены параметры динамической поверхности робота Lego Mindstorms EV3. Тело участвует в столкновения и обсчитывается динамическим движком симулятора.

На этом этапе необходимо соблюдать баланс в количестве вершин динамической поверхности. Чем их больше, тем точнее будет совпадать визуальная и динамическая модели, тем точнее будут расчеты столкновений и поведение объектов; но тем медленнее будут выполнять эти расчеты и каждый шаг симуляции соответственно. Существуют правила и приемы создания и совершенствования динамических моделей, улучшающие и уко-

ряющие работу симулятора. Они описаны в официальной справке [7] и используют некоторые хитрости симулятора.

На втором этапе устанавливаются все приводы (joint'ы), подвижные связи между частями робота или полигона. Если перенос робота из САПР в симулятор происходил с использованием URDF, то приводы могут быть уже созданы самим симулятором. Остается проверить их параметры и, при необходимости, откорректировать. Настройка заключается в указании пределов вращения / перемещения привода (или выборе непрерывного вращения), нулевой, а также стар-

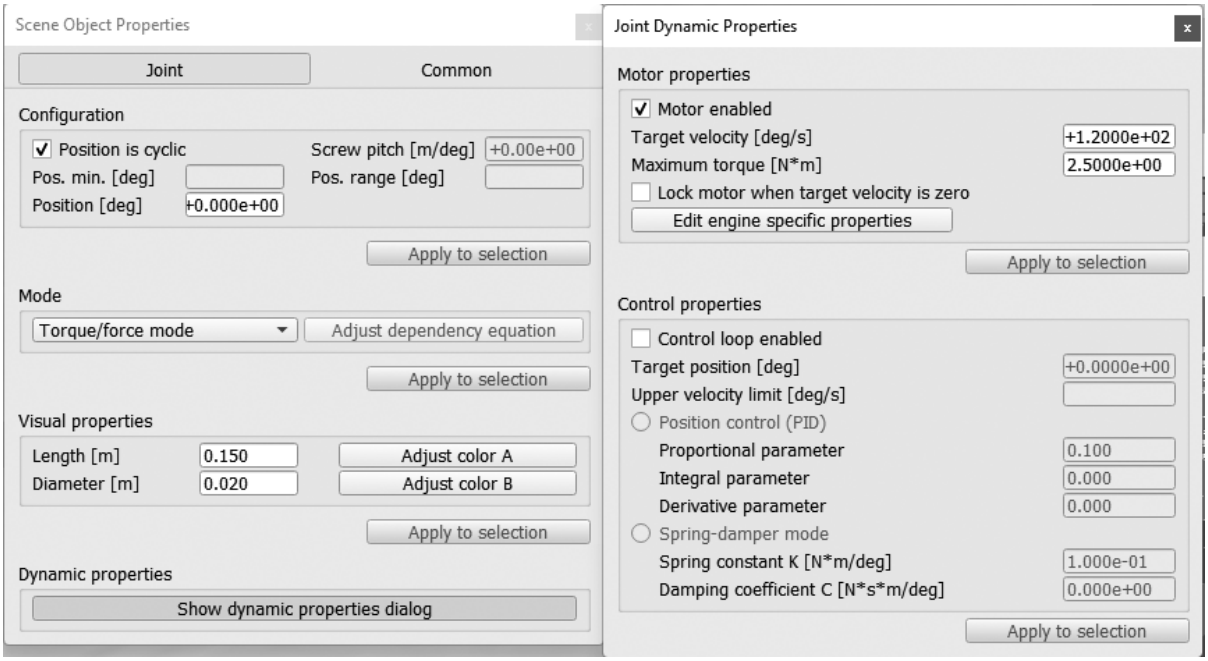

Рис. 5. Окна настройки параметров привода робота

товой позиции привода. Каждый привод может работать в нескольких режимах:

- **Passive mode** – привод свободно перемещается в заданных пределах угла поворота или линейного смещения; может использоваться для имитации пассивных шарнирных или скользящих соединений.
- **Torque / force mode** – управление усилием/крутящим моментом привода из программы. При этом усилие/момент может задаваться пользователем (участником соревнований) из программы напрямую или из встроенного ПИД-регулятора, а от пользователя потребуется указать коэффициенты регулятора и желаемое положение привода, после чего внутренний алгоритм попытается установить его в это положение. Перемещение происходит не мгновенно, а занимает несколько тактов симуляции, как в реальном мире. Неверно подобранные коэффициенты ПИД-регулятора могут вызвать слишком медленное перемещение звеньев робота или, наоборот, перерегулирование и колебания.
- **Dependent mode** – зависимый режим, в котором привод явно зависит от положения другого привода (через математическую зависимость, например, делает поворот в 3 раза меньше или в обратную сторону). Может применяться для имитации синхронной работы нескольких приводов, складывающих свои усилия.

Подробнее о настройках приводов можно почитать в официальной справке к симулятору [8]. Рисунок 5 иллюстрирует окна параметров приводов и выбор режимов их работы.

На следующем этапе у робота создаются датчики и настраиваются их параметры. CoppeliaSim позволяет имитировать поведение датчиков расстояния (proximity sensor), камер и оптических датчиков (vision sensor), сенсора усилий (force sensor) и датчиков положения приводов. При этом всем сенсорам могут быть заданы отдельные скрипты для отдельного шага симуляции (callback function), имитирующие обработку сигналов во внешних модулях реального робота, передающих в главный управляющий контроллер обработанный итоговый сигнал: координаты найденного на кадре камеры объекта, отфильтрованное расстояние до ближайшего препятствия, скорость вращения мотора вместо его текущего положения и т.д. Показания датчиков обновляются на каждом шаге симуляции, проходят обработку в этих функциях и могут использоваться в основной программе участника.

На четвертом этапе подготовки объектов сцены всем динамическим поверхностям уста-

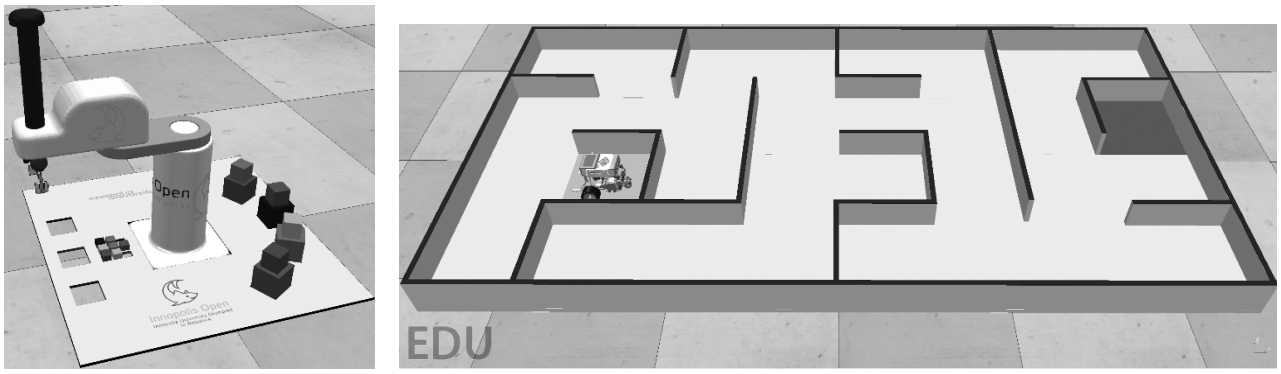

Рис. 6. Примеры готовых сцен робототехнических соревнований в симуляторе. **Слева:** старшая категория олимпиады Innopolis Open Robotics 2020. **Справа:** категория «Лабиринт: туда и обратно».

навливают реалистичные вспомогательные физические параметры: трение, отражение, свечение, параметры взаимодействия с датчиками. Для простых объектов, например, игровых элементов, динамическая и визуальная поверхность могут быть представлены одним объектом иерархии, но важно на этом этапе корректно его настроить. Визуально при подготовке и запуске робота модель может отображаться, но датчик-камера ее может не видеть, или датчик расстояния не учитывать как препятствие.

В завершении процесса подготовки объектов сцены (робота, игровых элементов, полигона) задаются визуально приятные и соответствующие действительным цвета и текстуры. Они накладываются на визуальные поверхности. Могут быть добавлены логотипы, разметка, имитация текстуры выбранного материала (дерева, металла, пластика). Стоит отметить, что в файле с описанием сцены симулятора изображения хранятся в несжатом виде и добавление текстур большого разрешения значительно увеличивает размер файла, вплоть до нескольких сотен мегабайт, и негативно влияет на скорость запуска сцены. Это необходимо учитывать при автоматическом тестировании множества решений участников, при неверно настроенной сцене запуск занимает большее время, чем ее проверка.

На рис. 6 приведены примеры моделей готовых сцен симулятора для соревнований «Лабиринт: туда и обратно» и олимпиады Innopolis Open Robotics. Цвета объектов, освещение и параметры отражения подбирались идентично реальным объектам и их свойствам.

Система жеребьевки и предварительной настройки полигона должна позволять быстро и просто указать стартовую конфигурацию поля перед запуском робота. Это могут быть и цвета объектов, и их расположение. Подобные элементы рандомизации определяются регламентом соревнований, например, это может быть конфигурация лабиринта, из которого роботу необходимо найти выход. Кроме того, этот скрипт должен позволять загрузить стартовую конфигурацию, предложенную организаторами, для соблюдения равных условий всех участников соревнований.

Ранее, при проведении виртуальных этапов соревнований с использования, система жеребьевки описывалась в управляющем скрипте всего полигона и использовала встроенный инструмент создания интерфейсов. Интерфейс (рис. 7) давал необходимые возможности, но был полностью открыт для пользователя и мог быть случайно изменен.

В будущем для предварительной настройки полигона, контроля и проверки выполнения задания роботом предполагается разработать отдельный плагин, так как это позволит разделить скрипт сцены, управляющую программу участника и код плагина, более гибко настроив их разрешения.

Подготовка среды программирования

Для каждого языка программирования необходимо готовить отдельную среду, состоящую из библиотек и классов робота, которые

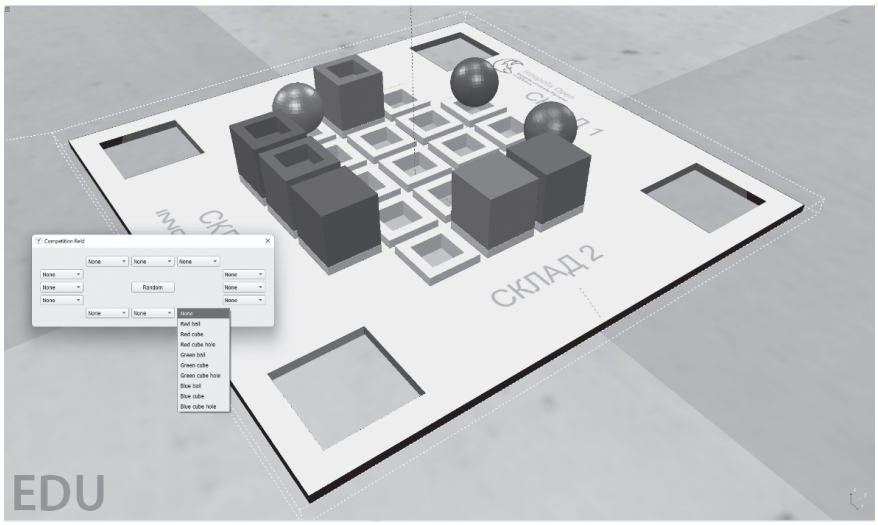

Рис. 7. Пример интерфейса установки объектов на полигоне

может вызывать участник в своей программе. Далее рассматривается пример конкретно для C/C++, хотя подобный функционал создается и для встроенных скриптов на Lua, и для Python, Java.

До версии 4.2 вместо библиотеки ZeroMQ [9] поддерживалась библиотека BlueZero [10]. В последней версии симулятора разработчики не рекомендуют ее использовать, считая ее устаревшей, но именно она, вместе с использованием Legacy remote API, была самым простым и понятным для детей и неподготовленных пользователей способом написания управляющих программ для робота на C/C++. Из-за более активного развития и поддержки полного спектра команд и функций (рис. 8), именно она была выбрана как инструмент написания программ участниками. Теперь ее заменила библиотека ZeroMQ, поддерживаемая и развиваемая в настоящий момент.

Инструкции по подключению библиотек и примеры простейших управляющих программ приведены в официальной справке к симулятору [11]. Для еще большего упрощения, унификации решений участников и автоматизации проверки организаторами соревнований заранее подготовлены несколько файлов:

- Robot.h / Robot.cpp – файлы, описывающие класс робота, модель которого необходимо запрограммировать участнику. Не предполагается, что участники соревнований будут их изменять; в проверяющей системе используются оригинальные файлы. Класс Robot предоставляет необходимые параметры и функции управления роботом, в том числе считывание датчиков, управления приводами,

	Easy to use	Directly available functions	Languages
ZeroMQ remote API	++	All	C++, Python (Java & Matlab coming soon)
WebSocket remote API	++	All	JavaScript
Legacy remote API	+	Subset	C++, Python, Java, Matlab, Octave, Lua

Рис. 8. Сравнение набора команд и доступных языков программирования внешних API

изменение режимов их работы и всего робота в целом. Подобный подход имитирует использование реального робота: участник может программно повлиять только на него, через него получает данные об окружающем мире, но не может использовать внешние данные или «волшебство» по перемещению или удалению объектов.

■ *Competition_name*.cpp – файл для написания программы конкретным участником соревнования (*Competition_name* – название соревнований). В нем происходит подключение всех необходимых библиотек, приведены примеры обращения к роботу и его функциям: считывания показаний датчиков, управления приводами, изменения режимов работы. В качестве решения участнику требуется прислать этот файл с описанной в нем программой или скомпилированный файл *Competition_name*.exe. Ниже приведен пример кода из соответствующего файла старшей возрастной категории олимпиады Innopolis Open Robotics 2020:

```
float kp = 0.1;          //пропорциональный
                                  коэффициент
float ki = 0.01;         //интегральный
                                  коэффициент
float kd = 0.1;          //дифференциальный
                                  коэффициент
robot.setPID
(robot.linkX, kp, ki, kd);     //обновление
                                  коэффициентов
robot.resetPID
(robot.linkX);                 9//применение
                                  коэффициентов
```

■ ReadMe.pdf – краткие инструкции по использованию симулятора, сцены и написанию базовой управляющей программы.

Все эти файлы, а также необходимые для подключения к симулятору библиотеки, представляют собой базовый инструментарий участников для быстрого старта.

На рис. 9 представлен кадр выполнения роботом задания и консоль программы участника. В левом верхнем углу видно изображение с камеры робота, обрабатываемое участниками.

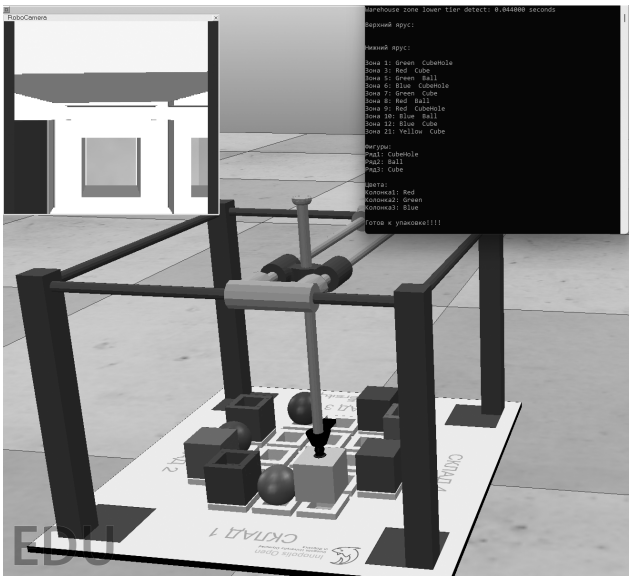

Рис. 9. Модель робота-манипулятора и консоль программы участника в процессе выполнения олимпиадного задания

Таким образом, подготовленный организаторами инструментарий позволяет участникам сконцентрироваться на алгоритмическом решении задачи соревнования, а не на подготовке и настройке симулятора или изготовлении механики и электроники робота. В качестве решения участникам достаточно отправить один файл *Competition_name*.cpp с исходным кодом управляющей программы, который будет запущен и проверен в тестирующей системе. Причем использование инструментов виртуализации позволяет легко масштабировать количество участников от единиц и десятков до сотен и тысяч, изолированно проверяя их решения. А фокус на робототехнику позволяет под новым углом взглянуть и отработать на практике знакомые детям алгоритмы регулирования, автоматов состояний, фильтрации показаний датчиков, компьютерного зрения и подобных, широко применяемых на реальных роботах. Симулятор же позволяет делать переход в робототехнику более плавным, абстрагируясь от некоторых проблем на более простых уровнях и углубляясь в них по мере освоения материала и усложнения задач. Для этого достаточно уточнять модели робота и соревновательного полигона в симуляторе.

Список литературы

1. Carlos Fernando Joventino. Application of a ROS / CoppeliaSim Integration in a Practical "OBR" Competition Scenario. In: 2020 Latin American Robotics Symposium (LARS).
2. YouTube. Канал «Довуз Университета Иннополис», плейлист «Робототехника». – URL: https://www.youtube.com/watch?v=p2xstUY8pr8&list=PLn4CTt5ibU6e8ko0Ut3FbsisBd5KbTAXx
3. Занимательная робототехника. Курсы по робототехнике Университета Иннополис. – URL: http://edurobots.ru/innopolis-university-courses
4. CoppeliaRobotics. The robotics simulator SoppeliaSim. – URL: https://coppeliarobotics.com
5. Benjamin Balaguer, Stefano Carpin. Towards quantitative comparisons of robot algorithms: Experiences with SLAM in simulation and real world systems. USA, 2007.
6. I. Mas´ar, P. Bahn´ık. Online Robot Simulation Contest. Germany, 2009.
7. CoppeliaSim User Manual. Designing dynamic simulation. – URL: https://coppeliarobotics.com/helpFiles/en/designingDynamicSimulations.htm
8. CoppeliaSim User Manual. Joint types and operation. – URL: https://coppeliarobotics.com/helpFiles/en/jointDescription.htm
9. ZeroMQ. An open-source universal messaging library. – URL: https://zeromq.org
10. GitHub. BlueZero: Project Page. URL . – URL: https://github.com/CoppeliaRobotics/bluezero
11. CoppeliaSim User Manual. ZeroMQ remote API. – URL: https://coppeliarobotics.com/helpFiles/en/zmqRemoteApiOverview.htm

References

1. Carlos Fernando Joventino. Application of a ROS / CoppeliaSim Integration in a Practical "OBR" Competition Scenario. In: 2020 Latin American Robotics Symposium (LARS).
2. YouTube. Kanal «Dovuz Universiteta Innopolis», pleylist «Robototekhnika» Available at: https://www.youtube.com/watch?v=p2xstUY8pr8&list=PLn4CTt5ibU6e8ko0Ut3FbsisBd5KbTAXx
3. *Zanimatel'naya robototekhnika. Kursy po robototekhnike Universiteta Innopolis* [Entertaining robotics. Robotics courses at Innopolis University]. Available at: http://edurobots.ru/innopolis-university-courses
4. CoppeliaRobotics. The robotics simulator SoppeliaSim. Available at: https://coppeliarobotics.com
5. Benjamin Balaguer, Stefano Carpin. Towards quantitative comparisons of robot algorithms: Experiences with SLAM in simulation and real world systems. USA, 2007.
6. I. Mas´ar, P. Bahn´ık. Online Robot Simulation Contest. Germany, 2009.
7. CoppeliaSim User Manual. Designing dynamic simulations. Available at: https://coppeliarobotics.com/helpFiles/en/designingDynamicSimulations.htm
8. CoppeliaSim User Manual. Joint types and operation. Available at: https://coppeliarobotics.com/helpFiles/en/jointDescription.htm
9. ZeroMQ. An open-source universal messaging library. Available at: https://zeromq.org
10. GitHub. BlueZero: Project Page. Available at: https://github.com/CoppeliaRobotics/bluezero
11. CoppeliaSim User Manual. ZeroMQ remote API. Available at: https://coppeliarobotics.com/helpFiles/en/zmqRemoteApiOverview.htm