Сравнение образовательных сред визуального программирования роботов

Д.А. Мордвинов
Санкт-Петербургский государственный университет
Кафедра системного программирования
Email: mordvinov.dmitry@gmail.com

Ю.В. Литвинов
Санкт-Петербургский государственный университет
Кафедра системного программирования
Email: y.litvinov@spbu.ru

Введение

Понятие «Исполнитель» давно используется в преподавании информатики для того, чтобы объяснять школьникам и студентам основные принципы программирования. И это не случайно — программы незримы, поэтому обучающемуся сложно представить их себе и переиспользовать предыдущий опыт из реального мира, особенно при первоначальном изучении программирования. До сих пор одним из самых популярных исполнителей является «черепашка» LOGO[], предложенная Сеймуром Пейпертом ещё в 1967 году. Черепашка при исполнении программы перемещается по экрану и рисует линии, тем самым визуализируя ход исполнения. Помимо черепашки существует несколько других известных исполнителей, основанных на похожих принципах, например, учебная система «Робик»[], разработанная группой под руководством А.П. Ершова.

Однако исполнитель, перемещающийся по экрану компьютера, всё же недостаточно нагляден. Сеймур Пейперт в своих экспериментах использовал механического робота-черепашку[], управляемого с компьютера, что делало исполнение программы ещё более наглядным, а процесс программирования увлекательным. В те времена создание подобных устройств было сложным, поэтому механическая черепашка не получила широкого распространения.

Современная электроника позволяет создавать недорогие устройства управления, что дало возможность вернуться к идеям Пейперта и начать массовое внедрение реальных исполнителей в образовательный процесс. При этом чаще всего используются робототехнические конструкторы, такие как Lego Mindstorms NXT[], Lego EV3[], ТРИК[] и т.д.

Робот, собранный из конструктора, сложнее в программировании, чем «черепашка», потому что конструктор позволяет создавать произвольные конструкции, которые приходится программировать в терминах мощностей моторов на портах, а не конкретных движений и поворотов. Поэтому при использовании таких конструкторов в образовательных целях большое внимание уделяется средствам программирования. В образовательной робототехнике очень популярны среды, использующие визуальные языки, поскольку визуальные языки проще в использовании и нагляднее, чем текстовые — иногда на визуальной среде программировать робота могут даже дошкольники, не умеющие ещё даже читать[].

Таких сред сейчас существует довольно много, все они обладают своими достоинствами и недостатками. Эта статья задумывалась как насколько возможно полный обзор образовательных визуальных сред программирования роботов, где имеющиеся среды будут проанализированы в соответствии с критериями, важными для образования, и будет также приведён обзор среды программирования TRIK Studio[], разрабатываемой авторами данной работы. В этот обзор не включены профессиональные среды, если по ним не удалось обнаружить информации об их использовании в образовательном процессе, однако включены среды, где работа ведётся с роботом, симулируемым на экране. Статья может быть интересна как тем, кто занимается преподаванием информатики или робототехники, так и тем, кто разрабатывает или планирует разработку подобных сред.

1. Критерии обзора

Критерии сравнения сред программирования были выбраны исходя из опыта общения со школьными учителями и преподавателями вузов, преподающими робототехнику и информатику[], общения со школьниками и студентами[], а также личного педагогического опыта авторов. Критерии выбирались так, чтобы, где это возможно, оценка среды была объективной и легко проверяемой. Критерии разбиты на несколько групп для удобства представления результатов и сравнения сред.

1. Группа критериев, связанных с выразительной силой средств программирования.
   (a) Наличие визуального языка программирования робота (да/нет).

(b) Наличие текстового языка программирования робота (список поддерживаемых языков).

(c) Поддержка записи математических выражений в «натуральной» форме (да/нет). Не все среды программирования позволяют записывать выражения в том виде, в котором они знакомы из курса математики.

(d) Используемая модель вычислений (поток данных/поток управления). Среды, использующие поток данных как модель вычислений, позволяют элегантно описывать сложные модели и больше подходят для опытных программистов, тогда как среды, использующие поток управления, более полезны при первоначальном обучении информатике, поскольку позволяют легко перейти к «обычному» текстовому программированию в императивном стиле.

(e) Наличие поддержки обычных алгоритмических конструкций (численная шкала от 0 до 3). Каждая поддерживаемая конструкция из списка «ветвление-цикл», «подпрограмма», «параллельные потоки» даёт 1 балл, оценка по этому критерию равна сумме баллов, набранных средой. Этот критерий косвенно выражает полезность среды при изучении информатики.

2. Группа критериев, связанных с наличием инструментальных средств.

(a) Наличие средств дистанционного управления роботом при интерпретации программы на компьютере (да/нет). Интерпретация делает исполнение программы более наглядным и позволяет предоставить обратную связь от робота.

(b) Наличие средств автономного исполнения программы на роботе (да/нет). Автономное исполнение важно в соревновательной робототехнике, где важна скорость работы программы и управление с компьютера не обеспечивает достаточного быстродействия, и в ситуациях, где связь с компьютером поддерживать технически сложно (например, автономные авиамодели).

(c) Наличие возможности по визуальному представлению генерировать пригодный для просмотра код на текстовом языке (да/нет). При обучении информатике важно показать, как диаграммы связаны с текстом программы.

(d) Наличие возможности исполнить программу на симуляторе робота на компьютере (да/нет). Это очень важно для преподавания, поскольку позволяет учащимся делать часть работы без доступа к роботу, например, дома. Кроме того, это упрощает отладку, поскольь-

ку позволяет сконцентрироваться на алгоритме, избежав проблем, возникающих при работе с физическими устройствами.

(e) Наличие средств отладки и контроля состояния исполняемой программы (да/нет). Отображение текущего исполняемого оператора, переменных программы и показаний датчиков делает исполнение программы более понятным и наглядным.

3. Группа критериев, связанных с прагматикой использования среды в образовании.

   (a) Простота, современность и визуальная привлекательность интерфейса (численная шкала от 1 до 5). Оценка ставится на основании мнения авторов данной работы, при этом будут даны пояснения оценке в кратком описании среды.

   (b) Наличие методических пособий (да/нет). Готовые курсы и методические материалы очень важны при внедрении в образовательный процесс.

   (c) Наличие встроенных средств проверки задач (да/нет). Такие средства позволяют эффективно учиться даже без участия учителя.

   (d) Русификация (да/нет). Не все обучающиеся владеют иностранными языками, поэтому отсутствие русификации может существенно повысить когнитивную нагрузку.

   (e) Стоимость (бесплатная/платная). Не все образовательные учреждения могут себе позволить дорогие средства программирования.

4. Группа критериев, связанных с технологическими аспектами среды.

   (a) Кроссплатформенность (список из «Windows», «Linux», «Mac OS», «Android», «Браузер»).

   (b) Поддержка популярных робототехнических конструкторов (список из «Lego NXT», «Lego EV3», «ТРИК», «другие»).

   (c) Лицензия (проприетарная/open source). Наличие открытых исходных кодов позволяет дорабатывать среду силами энтузиастов и избежать ситуации «vendor lock-in».

   (d) Наличие поддержки и развития среды (да/нет).

Разумеется, по предложенным критериям невозможно ввести метрику, позволившую бы определить «лучшую» среду. Каждая среда программирования, если она используется, в чём-то превосходит аналоги и лучше подходит

для той задачи, которую решает, но решаемые задачи для всех существующих сред разные. Этот обзор имеет своей целью предоставить фактическую информацию, необходимую для того, чтобы сделать осознанный выбор среды программирования, которая подходила бы для конкретной ситуации, и указать на важные элементы функциональности подобных сред.

Дальнейшее изложение будет построено следующим образом: будут представлены все рассмотренные среды, каждая из них с кратким описанием, после чего будут приведены таблицы с результатами оценки сред по приведённым выше критериям.

2. Среда LabView и ее адаптации

LabVIEW[] — кроссплатформенная среда моделирования и разработки общего назначения. Среда является проприетарной, создана американской компанией National Instruments в 1986 году и поддерживается до сих пор. Программирование в LabVIEW осуществляется на визуальном языке программирования потоков данных G. Язык G моделирует процесс вычислений, ориентированный на данные, в котором явно задаётся не последовательность выполнения операторов, а связи между блоками по данным. Блок программы может предоставлять некоторые выходные данные, которые могут служить входными данными для другого блока. Блоки начинают исполняться, когда имеют данные на всех входах. Если сразу несколько блоков имеют данные на всех входах, они исполняются параллельно. Такой подход довольно сильно отличается от подхода, принятого в императивном программировании, но тем не менее, он широко распространён среди инженеров и учёных. Например, на тех же принципах основана другая известная визуальная среда программирования научных вычислений и моделирования Matlab/Simulink.

LabVIEW применяется исследователями и инженерами по всему миру для автоматизации лабораторных и технологических процессов. Среда поддерживает огромное количество аппаратных платформ и предоставляет десятки библиотек программных компонентов от сложных математических методов обработки информации и алгоритмов компьютерного зрения до средств взаимодействия с системами управления базами данных, создания пользовательских интерфейсов и внешнего вида приборных панелей. В частности, имеются модули, обеспечивающие поддержку в LabVIEW робототехнических конструкторов Lego Mindstorms NXT и Lego Mindstorms EV3.

По сравнению с другими платформами, представленными в данной работе, система LabVIEW в «оригинальном» виде достаточно сложна для изучения, однако нередко применяется в образовании. Образовательные эксперименты с применением LabVIEW в средних школах были особо популярны в 1990-х годах[] (что дало начало различным образовательным адаптациям LabVIEW,

таким как Robolab), значительно чаще в настоящее время среда используется в ВУЗах, тематических кружках и лагерях[].

Среда обладает возможностями генерации кода для автономного его исполнения на роботе, а также отладки программ на компьютере с посылкой команд на робота. Алгоритмические конструкции поддержаны в полном виде, однако модель исполнения является потоковой, что может быть весьма удобным для программирования многих алгоритмических конструкций, но сложным для изучения и понимания.

Пользовательский интерфейс приложения и создаваемые диаграммы не выглядят современно (что объясняется возрастом системы). Создаваемые в среде программы довольно часто получаются громоздкими и трудно читаемыми, особенно если рисуются пользователями-новичками. Система предоставляет тысячи блоков, пиктограммы которых зачастую различаются очень мелкими деталями. Эти факторы с учетом стоимости лицензии среды и ее сложности делают довольно редкими применения LabVIEW в школьном образовании, гораздо чаще для этого используются различные ее адаптации, о которых будет рассказано ниже.

2.1. Среда NXT-G

Среда NXT-G[] — средство программирования, которое поставляется в комплекте с конструктором Lego Mindstorms NXT. Среда базируется на системе LabVIEW, поэтому NXT-G «наследует» язык потоков данных G. Блоки автоматически размещаются на диаграмме, свойства могут быть отредактированы прямо на графическом представлении блока. Средствами LabVIEW возможно добавление сторонних блоков, кроме того, сам NXT-G позволяет выделить набор блоков в подпрограмму и использовать её как новый блок.

Основная проблема NXT-G состоит в «слабой» поддержке математических выражений. Математические формулы здесь, как и вся программа, строятся визуально, из блоков. Существуют блоки чтения и записи значения в переменную, блок, считывающий значение константы, блоки, считывающие показания с сенсоров и блоки элементарных арифметических операций, не предоставляющих, впрочем, даже операции извлечения арифметического квадратного корня. Таким образом, даже чтобы запрограммировать несложную формулу, требуется изображать блоками дерево разбора выражения, которое эту формулу задаёт. Серьезность данной проблемы иллюстрируется программой, реализующей пропорционально-дифференциальный регулятор для движения робота вдоль линии или вокруг препятствия. На языке C такая программа занимает порядка десятка строк, тогда как на NXT-G не помещается на одном экране, содержит множество потоковых связей и весьма сложна для понимания. Таким образом, одному из предложенных критериев — пригодности для иллюстрации содержательного материала информатики и

кибернетики — NXT-G не соответствует. В основном, поэтому NXT-G и не получил широкого распространения в школах.

Среда NXT-G специально создавалась для начинающих, поэтому довольно проста и удобна в работе. По мнению некоторых пользователей, она даже слишком эргономична, поскольку не даёт произвольно размещать блоки на диаграмме, автоматически (и не всегда удачно) прокладывает соединительные линии между блоками и т.д. Существуют научные статьи, представляющие отрицательные результаты юзабилити-экспериментов с NXT-G[]. Кроме того, для применения NXT-G в школьных классах оказалась важна такая особенность: большинство свойств элементов не отображается на диаграмме, а доступна только через редактор свойств, что делает невозможным отображение всей программы на проекторе.

Никаких средств отладки NXT-G не имеет, текстовая форма программы не порождается, русификация существует, но неофициальная. К плюсам продукта следует отнести то, что он распространяется вместе с конструктором и доступен для загрузки с сайта производителя бесплатно.

2.2. Robolab

Еще одна среда, базирующаяся на LabVIEW — Robolab — специально создавалась как адаптация LabVIEW для школьного образования и с самого начала своего развития учитывала пожелания школьных учителей и специфику преподавания в школах[]. Пример специфичного для школ решения, реализованного в Robolab — наличие двух уровней сложности языка среды. Каждый из двух уровней сложности разбивается еще на несколько, на каждом подуровне пользователю «открывается» новая функциональность. На самом простом, пилотном уровне доступны только некоторые возможности визуального языка, и программа строится заполнением пустых мест в шаблоне посредством выбора блоков из всплывающего меню. Это позволяет создавать только самые простые программы, имеющие стандартную структуру: команды управления моторами, за которыми следует блок, ожидающий наступления какого-либо события. Идея такого разделения в том, чтобы дать возможность детям в начальной школе или даже детском саду пользоваться программой — в столь раннем возрасте они вполне могут не уметь читать. На втором уровне сложности пользователи могут рисовать диаграммы, размещая произвольным образом блоки из палитры и соединяя их линиями, определяющими поток управления. Разбиение на уровни и подуровни организовано так, чтобы дети могли осваивать среду программирования практически без помощи учителя, руководствуясь лишь интуицией. Отзывы педагогов, приведённые в [], показывают, что этой цели удалось достигнуть.

В отличие от NXT-G, Robolab позволяет описывать произвольные математические выражения в текстовом виде. Для задания формул могут исполь-

зоваться тригонометрические функции и обращения напрямую к значениям показаний сенсоров. Циклы в Robolab описываются при помощи блоков «метка» и «переход к метке» — передача управления никак больше не визуализируется. Имеются условные операторы, возможность запуска параллельных процессов, блоки для управления этими процессами, а также средства работы с подпрограммами. На Robolab возможно реализовать достаточно сложные программы, и Robolab вполне подходит для иллюстрации материала из кибернетики вплоть до младших курсов вузов.

По другим критериям Robolab показывает несколько худшие характеристики. Приложение было создано в конце 90-х годов, с тех пор его интерфейс практически не менялся, поэтому сейчас он выглядит несовременно, диаграммы, хоть и наглядно визуализируют поток управления, на больших размерах становятся трудно читаемыми. Специализированных средств отладки в Robolab нет, хотя есть возможность снятия показаний устройств ввода с робота и отображения их на экране компьютера. Возможности генерации диаграммы в текстовое представление не поддержано. Русификация присутствует, но лишь частично, некоторые элементы управления не переведены. Robolab не бесплатен, одна лицензия по стоимости сравнима с робототехническим набором, что для школ довольно дорого. Развитие Robolab идёт в основном путём добавления новых блоков, сама среда давно не изменялась. В частности, в среде отсутствует поддержка Lego EV3.

Несмотря на указанные недостатки, Robolab на данный момент является одной из наиболее широко используемой в школах средой программирования роботов. По отзывам преподавателей, у многих имеется желание от него отказаться и заменить на более современную систему.

2.3. Среда EV3-G

Среда EV3-G — программное обеспечение, поставляемое в комплекте с конструктором Lego Mindstorms EV3. EV3-G также создана на основе LabVIEW и позволяет программировать контроллеры NXT и EV3 на языке G. Среда во многом аналогична NXT-G: имеет современный пользовательский интерфейс, набор примеров, исполнения диаграмм на роботе. Блоки, из которых создается программа, автоматически «сцепляются» друг с другом (аналогично NXT-G), имеют понятное и удобное для редактирования графическое представление. В среде также исправлен один значительный недостаток NXT-G — математические выражения могут задаваться значительно удобнее, в том числе, текстом.

Тем не менее, некоторым критериям EV3-G не удовлетворяет. В среде отсутствуют возможности отладить программу на симуляторе, а также не поддерживается ОС Linux. Имеются известные проблемы совместимости с NXT. В робототехническом сообществе распространено мнение, что среда плохо

подходит для создания больших и сложных программ. Система является бесплатной для индивидуальной эксплуатации, однако версия для образовательных учреждений является платной.

3. Microsoft Robotics Developer Studio

Среда Microsoft Robotics Developer Studio[] — разработка компании Microsoft, предназначенная для программирования сложных многопоточных приложений с реактивной моделью поведения, используемых для управления робототехническими системами. Необходимость создания таких приложений есть не только в робототехнике, поэтому Robotics Developer Studio используется и для создания приложений, к робототехнике не относящихся (например, социальная сеть MySpace использует MRDS как составную часть серверного ПО[]). Программы в Robotics Developer Studio рисуются в виде диаграмм на языке VPL (Visual Programming Language), визуализирующим связи между отдельными параллельно исполняемыми компонентами (веб-сервисами), из которых состоит программа.

Необходимо отметить, что в сфере школьного образования Microsoft Robotics Developer Studio используется очень редко. Главная причина этого заключается в том, что среда рассчитана в основном на симуляцию и не может эффективно взаимодействовать с реальным роботом. Для LEGO Mindstorms NXT есть возможность управления по каналу Bluetooth, но нет возможности загрузки программы на робота для автономного ее исполнения — на роботе нет возможности запустить виртуальную машину .NET. Реальные роботы, управляемые MRDS, обычно гораздо сложнее и дороже того, что можно использовать в школах (стандартная платформа, например, имеет в своём составе ноутбук). Управления по Bluetooth недостаточно для решения задач, требующих малого времени реакции робота, из-за больших задержек посылки-приёма Bluetooth-пакетов, что делает MRDS неприменимой для большой области решаемых в школе задач. Симуляции же тоже оказывается недостаточно, потому что даже с хорошей физической моделью мира MRDS создаёт модель некоторого идеального мира, в котором большого количества проблем, решаемых алгоритмами кибернетики, просто не возникает. Даже простая задача, решаемая на реальном роботе, может оказаться нагляднее и полезнее школьникам, чем сложная программа, исполняемая на модели в симуляторе.

Вторая важная причина очень узкого распространения MRDS в школах — используемая в ней модель вычислений. Представление программы в виде набора взаимосвязанных распределённых веб-сервисов может быть удобным для опытных программистов, но начинающим тяжело понять принципы, лежащие в основе такой модели. Сложные механизмы взаимодействия веб-сервисов во многом «спрятаны» с помощью визуального языка VPL, но всё же требуется

некоторое понимание происходящих в системе процессов для того, чтобы рисовать содержательные диаграммы. В целом можно сказать, что MRDS больше подходит для студентов или профессиональных программистов, чем для школьников. Среда хоть и позволяет писать сколь угодно сложные распределенные программы, но делать это можно довольно нетривиальным и специфическим образом, что сильно снижает её ценность как иллюстративного материала.

Что касается других критериев, система довольно удобна в работе, имеет средства отладки и генерации кода, однако, в силу своей специфики, эти средства довольно сложны для использования школьниками. Русификация системы отсутствует. С 2014 года MSDS официально не поддерживается компанией Microsoft.

4. Scratch и Scratch-подобные среды

Scratch[] — кроссплатформенная визуальная среда программирования с открытым исходным кодом, разрабатываемая в Массачусетском Технологическом Институте для обучения школьников основам информатики. Программирование осуществляется посредством соединения блоков, напоминающих элементы мозаики. Scratch позволяет нарисовать и запрограммировать простые графические объекты, называемые спрайтами.

В «чистом» виде Scratch не позволяет программировать роботов, однако существует большое количество расширений и сред на базе Scratch, позволяющих программировать роботы Lego WeDo, Lego NXT, Lego EV3 и Arduino. Среди таких «самостоятельных» проектов, созданных на базе Scratch, упомянем S4A и mBlock для программирования Arduino и Enchanting для программирования NXT, а также российский проект ScratchDuino. Общими для Scratch-подобных сред плюсами являются легкость в изучении, привлекательный пользовательский интерфейс, открытость и бесплатность, возможность отладки удаленного управления роботом с компьютера и загрузки кода для автономного исполнения (последнее доступно не во всех Scratch-системах). Существует также возможность исполнения программы на виртуальном спрайте, что может рассматриваться по нашим критериям как отладка на симуляторе (однако о приближенности такой симуляции к реальности речи не идет). Scratch русифицирован, в Интернете имеется множество примеров и обсуждений на тематических форумах.

К отрицательным сторонам отнесем отсутствие «продвинутых» средств обучения программированию. К примеру, отсутствует возможность генерации читаемого кода по визуальной модели, что могло бы значительно облегчить переход обучающихся на текстовые языки. Алгоритмические аспекты поддержаны не полностью, к примеру, отсутствует поддержка массивов раз-

мерности больше 1. Средства автоматической проверки корректности решения заданий также отсутствуют. Таким образом, Scratch хорошо подходит для изучения информатики и робототехники в младших и средних классах и хуже для старшего возраста и «продвинутых» занятий.

4.1. Blockly

Среда Blockly — визуальная Scratch-подобная среда, разрабатываемая компанией Google Inc. для обучения детей программированию. Blockly может быть встроен в сторонние веб-приложения, что активно используется разработчиками по всему миру (к примеру, в проектах AppInventor и Wonder Workshop, см. секции ниже). Интерфейс среды и визуальный язык практически не отличаются от Scratch.

Существует набор виртуальных исполнителей и заданий для этих исполнителей, для решения которых используется Blockly. Для каждого задания проверяется корректность его решения. При этом для каждого решения можно просмотреть код на языке JavaScript, соответствующий визуальной диаграмме.

4.2. AppInventor

AppInventor[] — среда визуального программирования приложений для платформы Android. AppInventor использует ядро Blockly в качестве редактора диаграмм, а также позволяет нарисовать макет пользовательского интерфейса Android-приложения. Приложение далее генерируется в код и может быть запущено на Android-устройстве. Одна из особенностей AppInventor, интересная в контексте данной статьи — возможность взаимодействия программируемых приложений с устройствами Lego NXT по протоколу Bluetooth. Это довольно часто используется для программирования пультов удаленного управления Lego-роботом. Остальные особенности AppInventor не представляют интереса для данной работы и здесь обсуждаться не будут.

4.3. 12Blocks

12Blocks — еще один Scratch-подобный инструмент для программирования Lego Mindstorms NXT и Arduino-роботов, доступны и другие, менее популярные платформы (например, Scribbler). Среда кроссплатформенная, доступна для ОС Windows, Linux и Mac OS X. Имеется возможность исполнения программ на трехмерном симуляторе Cogmation, генерации кода по диаграмме, интеграции с ROS. В языке поддержаны все основные алгоритмические конструкции и типы данных, существует возможность выделения кода в

подпрограмму. Существуют возможности автономного исполнения программы роботом, отладки программы на компьютере с посылкой команд роботу, а также построения графиков с сенсоров в реальном времени. Разработчики позиционируют систему как пригодную на всех этапах образования — от начальной школы до университета.

К отрицательным сторонам отнесем следующие. 12Blocks имеет слабую методическую поддержку, русификация отсутствует, в Интернете практически нет сообществ вокруг среды (однако есть набор видео-инструкций по пользованию основными возможностями среды). Также в системе отсутствуют какие-либо средства автоматической проверки заданий. 12Blocks распространяется по коммерческой лицензии, однако какая-либо активность на официальном сайте с 2014 года отсутствует.

5. Другие среды

К данной категории относятся официальные среды визуального программирования менее распространенных образовательных робототехнических платформ. Здесь будут упомянуты две таких среды: ROBO Pro и Scribbler Program Maker.

ROBO Pro — официальная среда программирования контроллера ROBO TX, управляющего моделями, собранными из конструктора fischertechnik. Программирование в ней ведется на языке блок-схем. Среда русифицирована, имеет возможности интерпретации программ на компьютере и загрузки на робота для автономного их исполнения. Язык поддерживает все основные алгоритмические конструкции и типы данных. Остальным, «продвинутым» критериям среда не удовлетворяет. Стоит отметить, что ROBO Pro — практически единственная среда, программирующая реальных роботов в терминах блок-схем, однако система не поддерживает программирование каких-либо других устройств, кроме контроллера ROBO TX.

Scribbler Program Maker — среда, предлагаемая производителями для программирования образовательных роботов Scribbler. Среда позволяет отладить программу на компьютере или запустить программу на автономное исполнение. Язык среды спроектирован таким образом, чтобы в нем нельзя было допустить синтаксические ошибки. В нем существует возможность задания всех элементарных алгоритмических конструкций, создания своих подпрограмм. Однако даже простейшие программы, такие как «следование вдоль черной линии», выглядят громоздкими и плохо читаемыми. Интерфейс среды довольно прост, однако официальной русификации для него не существует. Возможности среды ограничиваются описанными, более специализированной функциональности (как переход к текстовому программированию, отладка на симуляторе или проверка заданий) в системе не поддержано.

6. Исполнители

Среды программирования для виртуальных исполнителей, работающих на экране компьютера, заслуживают отдельного рассмотрения. Хотя их и нельзя назвать средами программирования роботов в буквальном смысле, в образовании они служат тем же целям. Самой старой и самой известной такой средой является «Logo» Сеймура Пейперта и её исполнитель «черепашка», разработка которых началась в 1967 году. В книге [], изданной в 1980 году, приводится подробная аргументация в пользу использования компьютеров и программ-исполнителей для обучения детей, и именно работы Пейперта можно считать заложившими основание концепции образовательных исполнителей. Несколько менее известно, что Пейперт предлагал использовать помимо «черепашки» на экране реального робота-черепашку, перемещающегося по полу и рисующего маркером.

Существует много современных реализаций черепашки и черепашьей графики, в том числе, браузерные Turtle Academy, Online Turtle Graphics, Turtle Graphics.org и т.д., и требующие установки: ЛогоМиры, KTurtle, даже интерпретатор, встроенный в известный текстовый редактор LibreOffice. Однако фокус этой статьи — визуальное программирование, поэтому следует упомянуть среду Turtle Art и её браузерную версию. Среда представляет собой объединение идей Scratch и Logo, позволяя программировать черепашку не текстовыми командами, а блоками, стыкующимися друг с другом как в Scratch. Среда не позволяет записывать математические выражения в естественной форме и не локализована, однако обладает развитыми средствами отладки.

Следуя работам Пэйперта были созданы и другие исполнители, такие как КуМир, среда «Исполнители», «Паркетчик» и т.д. КуМир имеет упрощённую версию для дошкольников и младших школьников, ПиктоМир, распространяемую отдельно и более интересную нам с точки зрения этого исследования, поскольку она, в отличие от вышеперечисленных сред, использует простой графический язык программирования. Также среда имеет браузерную версию.

7. Среды для дошкольников и начальных классов

Данная глава содержит описание сред программирования роботов и игр, где требуется программировать виртуальных роботов, которые используются в дошкольном и начальном образовании. Очевидно, что введенная в секции [] система критериев не годится для оценки сред для дошкольников, так как содержит слишком много требований, которым такие среды заведомо не удовлетворяют. Гораздо более важными для такой среды критериями являют-

ся ее «интуитивность», дружественность к пользователю и наличие игрового элемента. Первые два критерия не являются объективными, поэтому оценки, приведенные здесь, основаны на педагогическом опыте авторов и отзывах педагогов школьного образования, работавших с данными средами.

7.1. Lego WeDo Software

Lego WeDo — довольно популярное в России образовательное решение от Lego, ориентированное на дошкольников и учащихся первого-четвертого классов. С помощью Lego WeDo можно собирать простейшие подвижные конструкции, которые, при желании, могут оборудоваться датчиками расстояния и наклона, входящими в комплект.

Программирование осуществляется в визуальной среде, очень похожей на NXT-G и EV3-G (среда также основана на LabVIEW), однако с некоторыми упрощениями. Среда содержит множество примеров в виде картинок с моделями, собранными из конструктора, дружественна к пользователю, русифицирована. Однако язык G может оказаться сложным для детей дошкольного возраста. Кроме того, существуют отрицательные результаты юзабилити-экспериментов[], которые, хоть и проведены с NXT-G, касаются WeDo Software (к примеру, назначение не всех пиктограмм элементов в WeDo интуитивно понятно, зачастую они отличаются друг от друга лишь мелкими значками). Ко всему прочему, Lego WeDo Software не является бесплатной, цена сравнима со стоимостью самого конструктора.

7.2. Create (Arts & Bots)

Arts & Bots[] — проект, развиваемый в США и не столь популярный на данный момент в российском образовании. Идея Arts & Bots состоит в использовании канцелярских материалов (бумага, картон, краски и т.д.) совместно с аппаратными компонентами, входящими в набор (силовые моторы, сервомоторы, вибро-моторы, светодиодные ленты, набор датчиков и контроллер) для изготовления роботов и подвижных стендов.

Контроллер Arts & Bots программируется визуальной средой Create. Целью разработчиков Create было максимальное снижение порога вхождения в программную среду. По этой причине, к примеру, в системе отсутствует необходимость ввода чисел с клавиатуры, вместо этого скорость моторов регулируется ползунком. Визуальный язык специально спроектирован таким образом, чтобы в нем нельзя было допустить синтаксическую ошибку.

Программирование модели в Create осуществляется в два шага: определение элементарных модулей поведения (выражений) и определение последовательности их исполнения. Модули выражений определяются в редакторе, где

к схеме контроллера требуется подключить устройства и определить количественные параметры их входных и выходных импульсов. Полученные модули выражений могут быть затем использованы в модуле последовательностей для задания порядка их выполнения. Модуль последовательности имеет возможность задания развилки в зависимости от значения датчика (значения большие или равные пороговому передают управление по одной ветке, меньшие — по другой), определения циклов со счетчиком, а также выделения набора блоков в отдельный модуль. Более сложные поведения (например, с нетривиальным условием в развилке или с параллельным исполнением задач) не могут быть определены, однако, учитывая целевую аудиторию конструктора, этого и не должно потребоваться.

По утверждению авторов Create, среда подходит для обучения дошкольников и учеников начальной школы. Язык более прост, но менее выразителен, чем предлагаемый, к примеру, в Lego WeDo Software. Однако систему нельзя назвать интуитивной. К примеру, элементарные блоки, из которых выстраивается программа, вообще не имеют картинок в их графическом представлении, только текстовые подписи, что ставит под вопрос применимость в дошкольном образовании, где дети могут читать с затруднениями или даже не уметь читать вовсе. В Create отсутствуют игровые моменты. На данный момент проект не пользуется большой популярностью, поэтому, в частности, русскоязычная локализация и методические материалы отсутствуют.

7.3. Wonder Workshop

Еще одно образовательное решение, довольно популярное в США и странах Европы, однако практически не применяемое в России — роботы Dash и Dot от американской компании Wonder Workshop. Dash и Dot — готовые роботы, которые могут быть запрограммированы одной из двух визуальных сред. Первая из них основана на Blockly, о котором уже было рассказано в разделе []. Разработчики предлагают использовать Blockly с возраста 8 лет, так как создаваемые в нем программы содержат много текста.

Другая среда, Wonder, ориентирована на любой возраст. Среда практически не содержит текста, программирование ведется в терминах конечных автоматов: диаграмма составляется из красочно выглядящих блоков, соединяемых связями. Среда содержит набор заданий, оформленных в виде игры для прохождения: корректность каждого задания проверяется автоматически, при выполнении очередного задания пользователь допускается к следующему. Wonder содержит большое количество визуальных эффектов, прост в изучении, назначение всех элементов понятно на интуитивном уровне. На официальном сайте проекта доступны методические рекомендации, однако ни они, ни интерфейс программ не переведены на русский язык (на момент написания данной работы). Таким образом, среда Wonder удовлетворяет всем критери-

ям, предлагаемым здесь для оценки детских сред программирования, однако требует адаптации в России.

7.4. Развивающие игры, Lightbot

Существует целый класс развивающих игр для детей, где нужно так или иначе задавать программы поведения роботов. Одна из таких игр, Lightbot, применяется в дошкольном образовании в России и за рубежом. По представленной в работе классификации Lightbot скорее следует отнести к к классу исполнителей, цель игры — посетить все светящиеся точки на карте виртуальным мобильным роботом. Программа для исполнителя задается визуально, в виде цепочки простых действий (как, например, пройти вперед, повернуться, запрыгнуть на препятствие и т.д.), при этом нельзя превышать фиксированное количество используемых блоков. По мере прохождения игры задания усложняются, например, становится необходимым группировать блоки в подпрограммы и т.д. Игра не русифицирована, однако пояснительный текст зачастую и не нужен — требуемые от игроков действия ясны на интуитивном уровне.

8. Выводы

Из приведенного обзора образовательных визуальных сред программирования роботов можно сделать следующие выводы. Для дошкольного и начального образования существует не столь большое количество средств программирования роботов, большинство из существующих появились в 2010-х годах и постоянно появляются новые, к примеру, проект Root от Гарвардского Университета, упоминания о котором начали появляться в 2016 году (на момент написания статьи проект не столь популярен и поэтому подробно не рассмотрен). Однако даже несмотря на немногочисленность существующих решений, среди них уже существуют инструменты, качество которых, по мнению авторов, находится на удовлетворительном уровне.

Совсем другая ситуация с инструментами, применяющимися для среднего образования. Разброс довольно велик — от больших и сложных коммерческих инструментов, таких как LabVIEW и MRDS, до сред программирования конкретных робототехнических платформ, предоставляющих «минимальную» функциональность (редактор диаграмм + возможность запуска на роботе). «Продвинутые» функции, такие как генерация читаемого текстового кода по диаграмме, возможность симуляции на виртуальном устройстве или встроенная автоматическая проверка ограничений, практически не встречаются в популярных на сегодняшний день средах, а если такие функции и реализованы, то не все сразу. К тому же продукты, предлагающие такую функциональность, являются коммерческими, и многие учебные заведения попросту

не могут себе их позволить (не говоря об индивидуальном использовании). Более того, не существует единого решения, которое охватывало бы все популярные робототехнические образовательные платформы. Все это говорит о необходимости создания нового решения, которое бы учитывало все эти неудобства.

9. TRIK Studio

Среда TRIK Studio является попыткой решения вышеописанной проблемы. TRIK Studio позволяет визуально программировать различные робототехнические платформы. Программа в TRIK Studio составляется из блоков и стрелок, вместе описывающих поток управления программы. Исполнение начинается со специального начального блока и далее передается по стрелкам. Условие рисуется как развилка (две стрелки, отходящие от блока), бесконечный цикл — как связь назад, арифметический цикл — как набор блоков со связью назад и выходной стрелкой. Таким образом, поток управления программы наглядно визуализируется создаваемой диаграммой. Часть диаграммы может быть вынесена в подпрограмму для ее последующего переиспользования. Математические выражения, условия на развилках, значения свойств описываются на встроенном текстовом языке — Lua (точнее, его статически типизируемом диалекте).

Для облегчения рисования диаграммы среда может распознавать жесты мышью. К примеру, если пользователь правой кнопкой мыши в произвольном месте окна редактора нарисует стрелку вперед, появится блок включения моторов, если нарисует пиктограмму часов — появится блок ожидания, если провести линию от одного блока к другому, появится стрелка, их соединяющая, и т.д. Жесты распознаются довольно сложным алгоритмом[], поэтому могут восприниматься системой и не будучи в точности соответствующими идеальным (которые показываются средой во всплывающих подсказках к блоку вместе со всей необходимой информацией о самом блоке).

На данный момент среда поддерживает программирование конструкторов Lego Mindstorms NXT, Lego Mindstorms EV3 и конструктора ТРИК. Для каждого конструктора среда предоставляет три режима работы с ним: режим интерпретации, режим автономного исполнения и режим отладки на симуляторе. В режиме интерпретации программа исполняется на компьютере с отправкой команд роботу по какому-либо низкоуровневому протоколу (USB и Bluetooth для NXT и EV3, Wi-Fi для ТРИК). Значения всех переменных во время интерпретации могут быть просмотрены в соответствующем окне, а также можно отслеживать графики показаний датчиков, строящиеся в реальном времени. В режиме автономного исполнения среда генерирует код, компилирует его, если целевой язык не скриптовый, загружает по низкоуровневому

протоколу на робота и запускает его на исполнение, показывает его во встроенном текстовом редакторе. Код генерируется в читаемом виде, он может быть открыт и отредактирован во встроенном текстовом редакторе с подсветкой синтаксиса и автоматическим дополнением. Для одного конструктора TRIK Studio может поддерживать больше одного текстового языка. К примеру, в режиме ТРИК возможна генерация в JavaScript, F# и Pascal ABC.NET, в режиме NXT программа может быть сгенерирована в NXT OSEK C или русскоязычный школьный алгоритмический язык (ШАЯ).

В третьем режиме, доступном для каждого из поддерживаемых конструкторов, режиме симуляции, программа будет выполнена на двумерной модели робота, открываемой внутри окна среды. Двумерный симулятор позволяет пользователю нарисовать произвольную модель мира, состоящую из стенок, регионов и цветных элементов, нарисованных на полу. К примеру, могут быть нарисованы все стандартные поля и полосы препятствий, используемые в спортивной робототехнике. Далее указывается, какие датчики подключены к роботу, их пространственное положение и ориентация. Программа затем может быть исполнена на нарисованной модели мира, при этом, так же как и в режиме интерпретации на реальном устройстве, можно отслеживать значения переменных и графики значений сенсоров. Для удобства отладки скорость течения времени в модельном мире может быть уменьшена или увеличена.

Наличие режима симуляции полезно не только для отладки. Возможность программирования виртуального робота может быть полезна образовательным учреждениям и индивидуальным пользователям, у которых по тем или иным причинам отсутствует реальный робот. К примеру, детям, у которых дома нет роботов, преподаватели могут выдавать домашнее задание, которое нужно решить для виртуального робота. Двумерный симулятор робота может рассматриваться как исполнитель. В частности, робот может рисовать на полу след траектории его перемещения (аналогично исполнителю «Чертежник»). В среде имеется возможность автоматической проверки заданий. Задание описывается на внутреннем языке ограничений и может быть сохранено особым образом для последующего его распространения между учениками. Авторы данной работы участвовали в разработке онлайн-курса на платформе Stepic, состоящего из более, чем двадцати задач спортивной и образовательной робототехники. Каждая задача может быть решена на TRIK Studio или онлайн-среде, и затем проверена средствами проверки ограничений TRIK Studio на локальном компьютере и удаленном сервере.

TRIK Studio бесплатна, исходный код открыт для всех желающих. Среда переведена на 3 языка (русский, английский, французский), имеется справочная система. Проект находится на стадии активного развития, версии выходят часто, в каждой из них появляется функциональность, добавляемая по пожеланиям педагогов. Таким образом, среда удовлетворяет практически всем

предложенным в работе критериям.

Существуют три критерия, которым, однако, среда на данный момент не удовлетворяет или удовлетворяет не полностью. Один из самых главных недостатков — «бедность» методических материалов на русском и иностранных языках. Документация, на данный момент, ограничивается справкой, входящей в дистрибутив среды и набором видео-уроков из упомянутого онлайн-курса. Другие два недостатка связаны с языком среды. Во-первых, несмотря на то, что основная часть работы по написанию программы выполняется мышкой в визуальном редакторе, детям все же необходимо учить элементы текстового языка. Это может быть рассмотрено и как преимущество (язык становится «ближе» к текстовым, что облегчает переход на них), так и как недостаток (значительно повышается порог вхождения в среду). Наконец, на данный момент, язык недостаточно выразителен для описания всех основных алгоритмических конструкций. Речь идет о рекурсии с параметрами — последняя на данный момент версия среды (3.1.3) не имеет возможности передачи параметров в подпрограмму. Тем не менее, проект быстро развивается, поэтому недостатки носят скорее временный характер. Авторы также надеются, что проблема с методическими материалами будет решена в результате тесного сотрудничества с педагогами.

Заключение

В работе был представлен сравнительный анализ большого количества популярных на сегодняшний день образовательных сред визуального программирования роботов. Была предложена система критериев, по которой оцениваются среды из различных классов (среды для дошкольников и учеников начальных классов, и среды для среднего и высшего образования, в том числе виртуальные исполнители). Результаты сравнения сред из второго класса по такой системе критериев приведены в табл. 1 (исполнители не включены).

ТАБЛИЦА