RUDN Journal of Informatization in Education
2018 Том 15 № 1 18—28
Вестник РУДН. Серия: Информатизация образования
http://journals.rudn.ru/informatization-education

DOI 10.22363/2312-8631-2018-15-1-18-28
УДК 373

ВОЗМОЖНОСТИ И ПЕРСПЕКТИВЫ ПРИМЕНЕНИЯ ТЕХНОЛОГИЙ И СРЕДСТВ ВИЗУАЛЬНОГО ПРОГРАММИРОВАНИЯ ПРИ ОБУЧЕНИИ ШКОЛЬНИКОВ

Э.М. Каган

Московский городской педагогический университет
Шереметьевская ул., 29, Москва, Россия, 127521

В статье рассматриваются типичные проблемы, возникающие у учащихся при освоении программирования. Для каждой проблемы приводится ряд существующих специализированных сред программирования, предлагающих решение. Проводится анализ элементов сред программирования и выделение тех, которые непосредственно участвуют в решении проблем. В завершении проводится агрегация лучших решений, и делается предположение о возможности комбинирования лучших элементов проанализированных сред. Изначально школьный курс информатики был ориентирован на формирование навыков программирования и управления компьютером. Сейчас основная часть курса отводится изучению прикладного программного обеспечения и информационным технологиям. При этом для обучения в большей степени используются классические языки программирования, разработанные еще в прошлом веке. Первой успешной попыткой создания альтернативного языка программирования, который мог бы выступать в роли образовательного средства, является язык Logo. Аналогичный способ отображения можно найти во многих средах программирования, когда пользователь не должен иметь навыков программирования, но быть в состоянии составить работоспособный алгоритм. Это позволяет не отвлекаться на язык программирования, а конструировать программу из блоков. Каждая из упомянутых в статье сред визуального программирования не лишена ряда недостатков. Однако даже при таком положении дел просматривается тенденция расширения использования в обучении визуальных языков программирования.

Ключевые слова: визуальное программирование, среда программирования, проблемы обучения программированию, неклассические языки программирования, анализ применимости

Вопрос о методике и средствах обучения программированию — ровесник доступных вычислительных машин. И если в первое время основным контингентом пользователей были в основном научные сотрудники, то с миниатюризацией и снижением стоимости они стали доступны широкому кругу профессий, что потребовало определенного уровня компьютерной грамотности, а, как следствие, и коррекции образовательных программ в целях приведения их в соответствие с требованиями рынка труда. Учитывая специфику компьютеров того времени, основным методом взаимодействия было программирование, и, как следствие, большинство образовательных программ включили в себя именно этот элемент, как базовый.

Однако с течением времени возможности компьютеров росли, и охватить все возможные сценарии его использования в рамках сколь угодно большой про-

граммы становилось не возможно. Параллельно с этим в сферу компьютерных систем пришла бизнес составляющая[1], что создало отдельный рынок программистов, который требовал специфических навыков от работника. Чтобы соответствовать требованиям, тогда и появилось отдельное направление подготовки выпускников, которые обладали фундаментальными знаниями компьютерной науки, а не навыками использования прикладного программного обеспечения.

Несмотря на то, что историческая основа современных программ в большей степени относится к уровню высшего образования, нечто подобное можно проследить, рассматривая развитие курса информатики в средней школе. Если изначально курс был ориентирован на формирование навыков программирования и управления компьютером, как машиной, то сейчас все большая доля курса отводится изучению прикладного программного обеспечения и информационным технологиям в целом [3; 6]. Отдельно необходимо остановиться на вопросе времени, уделяемого изучению алгоритмов и практическому программированию, в связи со смещением акцента на прикладное программное обеспечение [7]. Согласно примерным программам Л.Л. Босовой на изучение тем, связанных с алгоритмизацией и программированием, отводится всего 40 ч. В 6-м классе выделяется 10 ч [1], 21 ч в 8-м классе и всего 8 ч в 9-м классе [2]. Интересно отметить, насколько сильно изменилась программа — в первом варианте из 68 ч 48 отводилось на изучение именно темы «Алгоритмизация и программирование» при условии, что занятия проводились только в старшей школе, а техническое обеспечение школ было недостаточным [5].

Изучение базовых алгоритмических конструкций происходит в рамках 6-го класса с помощью создания презентаций, а практикум проводится с помощью среды КуМир. Данная среда также используется в 8 классе при изучении темы «Основы алгоритмизации». Далее происходит переход на язык Pascal в рамках темы «Начала программирования», этот язык также используется и в 9-м классе при изучении темы «Алгоритмы и программирование».

Таким образом, можно видеть, что обучение в большей степени использует классические языки программирования, разработанные еще в 70—80-х годах прошлого века. При этом более новая среда для начального обучения основам алгоритмизации от авторов среды КуМир[2] под названием Пиктомиры по какой-то причине не используется, несмотря на то, что она разрабатывалась именно для обучения в начальной школе и первых классах средней.

Еще в 80-х годах прошлого века Смит, Финцер и Голд и Сазерленд [10; 20; 22] начали говорить о необходимости альтернативных языков и сред программирования, которые были более просты в освоении. Их исследования являются результатом острой необходимости рынка в новых специалистах с хорошим уровнем знаний в области компьютерных наук, сильными навыками программирования и высокой мотивацией работы именно в этом направлении. Однако сложность использовавшихся тогда языков программирования, недостаток информации и большая вариативность самих платформ обучения приводили к тому, что даже

[1] Появление первых компьютерных корпораций, таких как IBM, Microsoft, Apple.

[2] URL: https://www.niisi.ru/kumir/ (дата обращения: 02.09.2017).

студент, проходящий специальное обучение, по его окончании далеко не всегда соответствовал предъявляемым требованиям. Важно заметить, что, несмотря на большое количество практических и теоретических разработок в области сред и языков программирования, острота проблемы обучения столь специфическому навыку сохраняется и по сей день [11].

Наличие заинтересованности со стороны рынка привело к тому, что на сегодняшний день существует ряд решений разработанных исключительно с образовательной целью и ориентированных на формирование универсальных навыков, необходимых программисту, таких как: декомпозиция, анализ, синтез, а также общее развитие алгоритмического, логического и системного мышления. Согласно текущему стандарту образования многие их этих компетенций должны быть сформированы в рамках школьного курса. Применение специализированных инструментов с большой вероятностью может повысить качество формируемых навыков, а также сократить время овладения ими. Исторически первой успешной попыткой создания альтернативного языка программирования, который мог бы выступать в роли образовательного средства, стал язык Logo, разработанный Сеймуром Пейпертом в конце 1960-х годов. Основной идеей автора являлось создание инструмента, с помощью которого ученики смогли бы научиться размышлять — преодолеть языковой барьер и начать общаться с машиной на ее языке [16]. Пейперт С., как и наш соотечественник А.П. Ершов, считал, что практикум программирования позволяет учащимся через выражение собственных мыслей в строгой нотации языка программирования постепенно привить им системное мышление, тем самым, повысив эффективность дальнейшего обучения.

Наиболее интересным с точки зрения образовательного потенциала в языке Logo является «черепашья» графика, которая позволяет наглядно продемонстрировать исполнение программы. Это небольшое нововведение в практику программирования позволило языку Logo стать на несколько десятилетий главным образовательным языком, а также продемонстрировать, что программирование может быть применимо не только на уроках информатики. Так, например, книга “Turtle Geometry: The Computer as a Medium for Exploring Mathematics” использует среду Logo, как инструмент интерактивной демонстрации математической теории [8].

Черепашка стала символом языка, однако изначальная проблема, которую хотели решить авторы, была совершенно иной. Разработка была начата, исходя из предположения, что наибольшей трудностью для учащихся представляется сложность синтаксиса существовавших языков программирования. Logo старался решить эту проблему путем исключения таких элементов, как точка с запятой в конце строки, обязательное прописывание скобок и требования относительно форматирования кода. Узнаваемая черепашка была добавлена в язык практически перед самым релизом среды.

Язык был также усечен с точки зрения возможностей, чтобы сделать задачи максимально простыми, а команды языка максимально близкими к естественному языку. По этой причине программы на нем могут казаться похожими на псевдо-код. По причине ограниченности предметной области Logo не может быть универсальным языком, что не является проблемой в случае обучения, тем более,

что можно применять для разных типов задач различные его диалекты, которых на данный момент существует более трех сотен [9]. Интересно отметить, что, несмотря на различия в предметных областях, большая часть из них сохраняет базовую механику взаимодействия пользователя с программой, а в большинстве случае в язык вносятся некоторые новые команды, изменяется отображение, или же корректируется сценарий взаимодействия с обстановкой. Logo и его диалекты, в большинстве своем, — языки, ориентированные на обучение, и при этом предоставляют решение для ряда часто возникающих у обучающихся проблем, таких как синтаксическая сложность, избыточная вариативность решения и отсутствие наглядного отображения хода исполнения программы.

Заложенные в язык Logo решения стали фундаментом для многих последующих разработок, а отдельные элементы используются и по сей день в неизменном виде. Здесь можно упомянуть среду ViMAP, которая позволяет работать с агентурными сетями в различных предметных областях [19]. При задании правил поведения для агентов используется визуальный язык, полностью ликвидирующий проблему набора текста программы. Полученная модель незаметно для учащегося транслируется в код языка NetLogo, который является диалектом Logo. При исполнении программы для отображения результатов работы используется сходная с черепашкой аллегория исполнителя, но в связи со сменой предметной области это может быть муравей, бабочка или что-то еще. При этом агенты отображаются в едином пространстве, а их программы могут взаимодействовать. Важно отметить, что количество предметных областей для среды может быть легко расширено с помощью написания новых команд на NetLogo [18]. При этом решается проблема замкнутости среды в одной предметной области, и появляется еще больше возможностей для применения программирования в рамках других курсов.

Как уже было указано, ViMAP использует визуальный язык программирования, что позволяет избежать ошибок при программировании и более наглядно отобразить сам алгоритм. Однако эта идея получила еще большее развитие в среде Boxer, разработанной знаменитым исследователем и программистом Андреа ди Сессой. Она использует концепцию наивного реализма, т.е., любой элемент системы отображаем (ничто не скрыто). Любая переменная будет отображена на экране, как и любая команда языка. При этом при отладке программы значения и ход выполнения также будут отображаться прямо поверх исходного «кода».

Такой подход мог бы стать популярным еще в начале 80-х годов прошлого века, однако, из-за привязанности среды к специфическому манипулятору (световое перо) его применимость была ограничена. На данный момент похожий способ отображения можно найти во многих средах программирования, ориентированных на профессиональное использование при производстве аудио- и видеоконтента, где пользователь не должен иметь навыков программирования, но при этом должен быть в состоянии составить работающий алгоритм. Такая особенность, очевидно, может быть использована в образовательном процессе, так как позволяет не отвлекаться на язык программирования в целом, а конструировать программу из блоков и проверять себя одновременно. Boxer, тем самым, решает проблемы сложности отладки, наглядности алгоритма программы и его исполнения одновременно, не ограничивая предметную область.

За более чем 30 лет изысканий многие исследователи также пришли к выводу, что главная проблема при обучении состоит в отсутствии у учащихся достаточно развитого навыка декомпозиции [21]. Одной из групп исследователей в Университете Карнеги-Меллон был разработан ряд сред, призванных решить и эту проблему. Примером одной из них может служить редактор Genie [14]. Данный редактор исходного кода является структурным, так как позволяет с помощью визуальных блоков выделять структурные элементы программы и запрещает произвольное написание кода. Важно заметить, что язык, используемый в Genie, не визуальный сам по себе, однако среда программирования делает его таковым.

Более того, сам редактор предполагает поэтапное продумывание программы, для чего содержит необходимый механизм документирования. Изначально пользователь задает структуру программы, добавляя в рабочую область необходимые элементы, и по необходимости комментирует их, чтобы дать самому себе и читателям кода пояснения о причинах выбранных решений, цели каждого блока, его аргументах и результатах. После чего, когда уже вся структура программы сформирована, можно приступать к программированию. Такой подход позволяет оказать поддержку начинающим программистам за счет исключения необходимости написания большого количества кода. Например, блок цикла с счетчиком содержит ограничения по умолчанию, заявление и зануление переменной счетчика и аккумулятор, при этом все, что остается программисту, — написать тело цикла. Данная техника получила название **скаффолдинг**[1] и применяется сейчас во многих профессиональных средах программирования и фреймворках.

Интересно, что одна из прародительниц современных профессиональных интегрированных сред разработки — среда программирования для языка Smalltalk, разработанная в Xerox PARC в рамках проекта Dynabook, также уделяла большое внимание документации и структурированности программ. Каждая команда языка может содержать подробное описание того, как именно она выполняется, какие атрибуты имеет, какое поведение порождает и какими будут результаты ее работы. Более того, комплект поставки включал в себя руководство программиста, интегрированное прямо в среду, к необходимому разделу которого можно было обратиться в любой момент.

Это руководство давало описание языка, его базовых концепций, структуры программ, а также набор примеров, который позволял быстро изучить сам язык. Важна здесь возможность, не покидая среды, выполнить код примера из руководства, скорректировать его и сразу же скопировать в код собственной программы. Вероятно, именно наличие столь кропотливо проработанной документации и простота ее доступности позволили данному языку завоевать определенную популярность и изменить историю развития языков программирования, сделав объектно-ориентированное программирование доминирующим на следующие несколько десятилетий.

Именно Smalltalk ввел такие важные понятия, как документация кода, интегрированная среда разработки, объектно-ориентированное программирование

[1] Скаффолдинг (scaffolding) — один из способов задания структуры программы с помощью набора мета-шаблонов. В переводе с английского scaffolding означает «строительные леса».

и среда с непосредственным взаимодействием. С момента ее появления все большее количество языков стали неразрывно связаны со своими средами программирования, так как последние позволяли избежать большого количества ошибок, тем самым, повышая качество программ и сокращая время разработки.

В рамках среды Smalltalk программа всегда будет «живой», что перекликается с идеями наивного реализма, но, кроме того, что все составляющие могут иметь визуальное представления, Smalltalk позволяет корректировать программу прямо в момент ее исполнения. Такой подход в значительной степени схож с механикой ручного управления черепашкой в среде Logo, что в некоторой степени закономерно, так как проект Dynabook изначально был ориентирован на создание компьютера, применимого в качестве средства обучения [13]. К несчастью, в момент выпуска язык не смог закрепиться в образовательном сегменте школ, некоторые исследователи связывают это с нехваткой подготовленных к преподаванию программирования с использованием объектно-ориентированного языка педагогов в тот момент.

Последним типом проблем, возникающих у многих школьников, о котором необходимо упомянуть, — проблема мотивации. Современные ученики воспринимают компьютерные технологии как нечто естественное, а навык использования прикладного программного обеспечения не позволяет им зачастую воспринимать компьютер как нечто большее, чем устройство, позволяющее выполнять набор различных действий. По этой причине заинтересованность в обучении не может появиться самостоятельно. Дополнительным сдерживающим фактором служит сама организация обучения, при которой программирование не позволяет увидеть результаты собственных действий достаточно ярко. Для ученика программа, которая производит сортировку массива, не будет интересной, так как она не имеет для него никакого практического смысла. Тем более, если эта программа должна быть написана на настольном персональном компьютере, и может быть запущена только на нем.

Решением данной проблемы могут стать более современные инструменты, которые позволяют работать с интерфейсом и быстро создавать программы, способные исполняться на мобильных платформах. Примером здесь может послужить среда AppInventor, которая была успешно применена в ряде тестовых групп [23]. Она позволяет конструировать небольшие приложения для платформы Android с помощью минимального количества строчек кода. Программирование интерфейса полностью исключает написание текста и производится простым перетаскиванием элементов из палитры. Чтобы программа заработала, достаточно написать несколько строчек кода, которые должны быть выполнены после некоторого действия над элементом интерфейса.

Еще более мотивирующее решение предложила компания LEGO. Представленный этой компанией робототехнический конструктор позволяет создавать в физическом мире несложные роботы, которые, однако, содержат сенсоры и актуаторы. После сборки робота желаемой конфигурации учащийся может запрограммировать его действия согласно внутреннему таймеру, значениям сенсоров, положению в пространстве и др. Важно заметить, что в последней версии конструктора авторы значительно переработали среду программирования — теперь

она доступна для устройств на базе Android и iOS, что позволяет вести программирование и управлять роботом без использования полноразмерного персонального компьютера [12].

Отдельного упоминания достоин социальный аспект. В качестве примера можно привести среду Scratch, разработанную Митчеллом Резником и коллегами. Кроме того, что среда использует блочный подход к оформлению алгоритма, обладает визуализацией исполнителя и хода выполнения программы, доступна на многих платформах, она также имеет встроенную поддержку сообщества. Это означает, что любой пользователь среды может получить доступ к большому количеству программ, выложенных другими такими же программистами, а также может сам опубликовать свою программу. Этот небольшой элемент среды позволяет быстрее обучаться за счет изучения кода других пользователей, с одной стороны, и развивает информационную культуру, с другой. Важно заметить, что на данный момент в репозитории сообщества Scratch находятся тысячи программ, изучение которых может стать хорошим подспорьем при изучении программирования. Сам автор говорит, что одной из задач при проектировании было формирование среды, способствующей «обучению без учителя» [17].

Подводя итог, можно сделать вывод, что каждая из упомянутых сред программирования не лишена ряда недостатков, и, как следствие, на данный момент не существует эталонного средства, как в период создания Logo. Нерешенным остается и ряд проблем визуального программирования таких, как преодоление порога предельного количества элементов на экране, создание емкого отображения для императивных конструкций, разработка методики обучения с использованием визуального программирования и многие др. Однако даже при таком положении дел явно угадывается тенденция все большего использования элементов визуальных языков [4]. Некоторые исследователи считают, что переход к полностью визуальному программированию является закономерным [10; 15]. На данный момент еще нет точно сформированного видения «следующего Logo», однако можно выделить ряд его характерных черт.

Двойное представление алгоритма. Программирование должно быть возможно, как с помощью полностью визуального инструментария (ориентированного на начинающих), так и с помощью классического текстового языка программирования.

Мультипарадигмальность. Универсальный учебный язык программирования также должен обеспечивать возможность вести разработку в нескольких парадигмах (например, в императивной и декларативной).

Расширяемость. В целях увеличения областей применения язык должен позволять создавать новые команды и формировать наборы команд, ориентированных на решение задач в различных предметных областях.

Интерактивное исполнение. Среда программирования для такого языка должна позволять пользователю вмешиваться в ход выполнения программы в произвольный момент времени без необходимости перезапуска всей программы.

Наглядное отображение алгоритма и хода исполнения программы. Алгоритм программы должен иметь аналогическое представление и быть легкочитаемым для пользователя без дополнительной подготовки.

Доступность. Среда программирования должна быть доступна, как на полноразмерных персональных компьютерах, так и на устройствах под управлением Android и iOS.

Документируемость. Язык программирования должен обладать встроенной подсистемой документирования программы, которая позволяла бы вести необходимое для понимания неподготовленным пользователем описание любого блока программы на естественном языке.

Возможность самостоятельного обучения. Пользователи должны иметь возможность начать программировать, взяв за основу пример одной из конструкций языка. Такой подход позволит сделать обучение самостоятельным за счет того, что программа может быть получена на основе использования образца и доведения его до необходимого состояния с помощью метода «проб и ошибок» с минимальной вовлеченностью учителя.

Социальность. Язык программирования может стать популярным только в случае, если вокруг него собирается достаточно большое сообщество энтузиастов. По этой причине описываемый язык должен иметь механизмы для обмена исходными «кодами» программ.

В завершении необходимо сказать, что рассмотренные примеры сред демонстрируют возможность решения для каждого из указанных здесь требований, однако совмещение лучших практик каждой из них является нетривиальной задачей и, вероятно, по этой причине нельзя наблюдать столь же прорывного языка, как Logo в свое время. Однако уже сейчас можно применять более современные среды программирования при обучении школьников программированию, повышая тем самым их заинтересованность, качество знаний и эффективность курса в целом.

СПИСОК ЛИТЕРАТУРЫ

[1] *Босова Л.Л., Босова А.Ю. Информатика.* 5—6 классы: метод. пособие. М.: БИНОМ. Лаборатория знаний, 2017. 384 с.

[2] *Босова Л.Л., Босова А.Ю.* Информатика. 7—9 классы: метод. пособие. М.: БИНОМ. Лаборатория знаний, 2015. 472 с.

[3] *Гриншкун В.В., Левченко И.В.* Особенности фундаментализации образования на современном этапе его развития // Вестник Российского университета дружбы народов. Серия: Информатизация образования. 2011. № 1. С. 5—11.

[4] *Гриншкун В.В.* Теория и методика использования иерархических структур в информатизации образования // Информатика и образование. 2003. № 12. С. 117—119.

[5] *Кузнецов А.А., Захарова Т.Б., Захаров А.С.* Общая методика обучения информатике: учеб. пособие для студентов педвузов. М.: Прометей, 2016. 300 с.

[6] *Лапчик М.П., Семакин И.Г., Хеннер Е.К.* Методика преподавания информатики: учеб. пособие для студентов педвузов. М.: Академия, 2001. 624 с.

[7] Федеральный государственный образовательный стандарт среднего (полного) общего образования. 2012. URL: https://infourok.ru/federalnyy_gosudarstvennyy_obrazovatelnyy_standart_srednego_polnogo_obschego_obrazovaniya_2012-414883.htm (дата обращения: 07.09.2017).

[8] *Abelson H., diSessa A. A.* Turtle Geometry: The Computer as a Medium for Exploring Mathematics: Turtle Geometry: The Computer as a Medium for Exploring Mathematics. MIT Press, 1986. С. 6—9.

[9] *Boytchev P.* Logo tree project. 2007. URL: http://elica.net/download/papers/LogoTreeProject.pdf (дата обращения: 07.09.2017).
[10] *Finzer W., Gould L.* Programming by Rehearsal // BYTE. 1984. Vol. 9. No. 6. Pp. 187—210.
[11] *Guzdial M.* Programming environments for novices: Computer Science Education Research. USA: CRC Press, 2004. — Pp. 127—154.
[12] *Gindling J.* LEGOsheets: A Rule-Based Programming, Simulation and Manipulation Environment for the LEGO Programmable Brick // IEEE Computer Society Press. Germany, 1995. Pp. 172—179.
[13] *Kay A., Goldberg A.* Personal Dynamic Media // Computer. 1977. Vol. 10. No. 3. Pp. 31—41.
[14] *Miller P.* Evolution of novice programming environments: The structure editors of Carnegie Mellon University // Interactive Learning Environments. 1994. Vol. 4. No. 2. Pp. 140—158.
[15] *Myers B. A.* Visual Programming, Programming by Example, and Program Visualization: A Taxonomy // Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Boston, Massachusetts, USA: ACM, 1986. Pp. 59—66.
[16] *Papert S.* Mindstorms: children, computers, and powerful ideas. New York, USA. Basic Books, 1980. 230 p.
[17] *Resnick M.* Scratch: programming for all // Communications of the ACM. 2009. Vol. 52. No. 11. Pp. 60—67.
[18] *Sengupta P.* Programming in K-12 science classrooms // Communications of the ACM. 2015. Vol. 58. No. 11. Pp. 33—35.
[19] *Sengupta P., Farris A. V., Wright M.* From agents to continuous change via aesthetics: learning mechanics with visual agent-based computational modeling // Technology, Knowledge and Learning. 2012. Vol. 17. No. 1/2. Pp. 23—42.
[20] *Smith D.C.* Pygmalion: a computer program to model and stimulate creative thought. Birkhauser, 1977. Vol. 40. Pp. 77—87.
[21] *Soloway E.* Learning to Program = Learning to Construct Mechanisms and Explanations // Commun. ACM. New York, NY, USA, 1986. Vol. 29. No. 9. Pp. 850—858.
[22] *Sutherland I.E.* Sketchpad: A Man-machine Graphical Communication System // Proceedings of the May 21-23, 1963, Spring Joint Computer Conference. Detroit, Michigan: ACM, 1963. Pp. 329—346.
[23] *Wagner A.* Using app inventor in a K-12 summer camp // Proceeding of the 44th ACM technical symposium on Computer science education. ACM. 2013. Pp. 621—626.

История статьи:
Дата поступления в редакцию: 13 сентября 2017
Дата принятия к печати: 30 октября 2017

Для цитирования:
Каган Э.М Возможности и перспективы применения технологий и средств визуального программирования при обучении школьников // *Вестник Российского университета дружбы народов. Серия «Информатизация образования»*. 2018. Т. 15. № 1. С. 18—28. DOI 10.22363/2312-8631-2018-15-1-18-28

Сведения об авторе:
Каган Эдуард Михайлович, аспирант кафедры информатизации образования Московского городского педагогического университета. *Контактная информация*: e-mail: eduard.kagan@yandex.ru

PROSPECTS AND OPPORTUNITIES FOR VISUAL PROGRAMMING TECHNOLOGIES AND TOOLS APPLICATION IN EDUCATING AT SCHOOL

E.M. Kagan

Moscow city pedagogical university
Sheremet'evskaja str., 29, Moscow, Russia, 127521

Article depicts typical problems that students encounter when mastering programming. For each problem, a number of existing specialized software environments offering a solution is displayed. Analyses of programming environments elements and the selection of those who are directly involved in solving problems are conducted. In the end, aggregation of the best solution is carried out and an assumption about the possibility of combining the best elements is made. Initially, the school computer science course was focused on the formation of computer programming and computer skills. Now the main part of the course is devoted to the study of applied software and information technologies. At the same time, classical programming languages developed in the last century are used for learning more. The first successful attempt to create an alternative programming language that could act as an educational tool is the Logo language. A similar display method can be found in many programming environments, where the user does not need to have programming skills, but be able to make a workable algorithm. This allows you not to be distracted by programming language, but to design a program from blocks. Each of the visual programming environments mentioned in the article is not without a number of drawbacks. However, even with this state of affairs, there is a tendency to expand the use of visual programming languages in teaching.

Key words: visual programming, programming environment, learning programming problems, non-classical programming languages, applicability analysis

REFERENCES

[1] Bosova L.L., Bosova A.Ju. *Informatika. 5—6 klassy* [Informatika. Classes 5—6]: metodicheskoe posobie. BINOM. Laboratorija znanij, 2017. 384 p.

[2] Bosova L.L., Bosova A.Ju. *Informatika. 7—9 klassy* [Classes 7—9]: metodicheskoe posobie. BINOM. Laboratorija znanij, 2015. 472 p.

[3] Grinshkun V.V., Levchenko I.V. *Osobennosti fundamentalizacii obrazovanija na sovremennom jetape ego razvitija* [Features of fundamentalization of education on the modern stage of its development]. *Vestnik Rossijskogo universiteta druzhby narodov. Serija «Informatizacija obrazovanija»* [Bulletin of the Russian university of friendship of the people. "Education Informatization" series]. 2011. No. 1. Pp. 5—11.

[4] Grinshkun V.V. *Teorija i metodika ispol'zovanija ierarhicheskih struktur v informatizacii obrazovanija* [Theory and methodology of using hierarchical structures in Informatization of education]. *Informatika i obrazovanie* [Informatics and education]. 2003. No. 12. Pp. 117—119.

[5] Kuznecov A.A., Zaharova T.B., Zaharov A.S. *Obshhaja metodika obuchenija informatike* [General methods of teaching computer science]: uchebnoe posobie dlja studentov pedvuzov. M.: Prometej, 2016. 300 p.

[6] Lapchik M.P., Semakin I.G., Henner E.K. *Metodika prepodavanija informatiki* [Methods of teaching computer science]: uchebnoe posobie dlja studentov pedvuzov. M.: Akademija, 2001. 624 p.

[7] *Federal'nyj gosudarstvennyj obrazovatel'nyj standart srednego (polnogo) obshhego obrazovanija* [Federal state educational standard of secondary (complete) General education]. 2012. URL: https://infourok.ru/federalnyy_gosudarstvennyy_obrazovatelnyy_standart_srednego_polnogo_obschego_obrazovaniya_2012-414883.htm

[8] *Abelson H., diSessa A.A.* Turtle Geometry: The Computer as a Medium for Exploring Mathematics: Turtle Geometry: The Computer as a Medium for Exploring Mathematics. MIT Press, 1986. Pp. 6—9.
[9] *Boytchev P.* Logo tree project. 2007. URL: http://elica.net/download/papers/LogoTreeProject.pdf
[10] *Finzer W., Gould L.* Programming by Rehearsal // BYTE. 1984. Vol. 9. No. 6. Pp. 187—210.
[11] *Guzdial M.* Programming environments for novices: Computer Science Education Research. USA: CRC Press, 2004. — Pp. 127—154.
[12] *Gindling J.* LEGOsheets: A Rule-Based Programming, Simulation and Manipulation Environment for the LEGO Programmable Brick // IEEE Computer Society Press. Germany, 1995. Pp. 172—179.
[13] *Kay A., Goldberg A.* Personal Dynamic Media // Computer. 1977. Vol. 10. No. 3. Pp. 31—41.
[14] *Miller P.* Evolution of novice programming environments: The structure editors of Carnegie Mellon University // Interactive Learning Environments. 1994. Vol. 4. No. 2. Pp. 140—158.
[15] *Myers B.A.* Visual Programming, Programming by Example, and Program Visualization: A Taxonomy // Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Boston, Massachusetts, USA: ACM, 1986. Pp. 59—66.
[16] *Papert S.* Mindstorms: children, computers, and powerful ideas. New York, USA. Basic Books, 1980. 230 p.
[17] *Resnick M.* Scratch: programming for all // Communications of the ACM. 2009. Vol. 52. No. 11. Pp. 60—67.
[18] *Sengupta P.* Programming in K-12 science classrooms // Communications of the ACM. 2015. Vol. 58. No. 11. Pp. 33—35.
[19] *Sengupta P., Farris A.V., Wright M.* From agents to continuous change via aesthetics: learning mechanics with visual agent-based computational modeling // Technology, Knowledge and Learning. 2012. Vol. 17. No. 1/2. Pp. 23—42.
[20] *Smith D.C.* Pygmalion: a computer program to model and stimulate creative thought. Birkhauser, 1977. Vol. 40. Pp. 77—87.
[21] *Soloway E.* Learning to Program = Learning to Construct Mechanisms and Explanations // Commun. ACM. New York, NY, USA, 1986. Vol. 29. No. 9. Pp. 850—858.
[22] *Sutherland I.E.* Sketchpad: A Man-machine Graphical Communication System // Proceedings of the May 21-23, 1963, Spring Joint Computer Conference. Detroit, Michigan: ACM, 1963. Pp. 329—346.
[23] *Wagner A.* Using app inventor in a K-12 summer camp // Proceeding of the 44th ACM technical symposium on Computer science education. ACM. 2013. Pp. 621—626.

Article history:
Received: 13 September, 2017
Accepted: 30 October, 2017

For citation:
Kagan E.M. (2018) Prospects and opportunities for visual programming technologies and tools application in educating at school. *RUDN Journal of Informatization of Education*, 15 (1), 18—28. DOI 10.22363/2312-8631-2018-15-1-18-28

Bio Note:
Kagan Eduard Mikhailovich, postgraduate student of department of informatization of education Moscow city pedagogical University. *Contact information*: e-mail: eduard.kagan@yandex.ru