Информационно-коммуникационные технологии в педагогическом образовании, 2025. № 1 (94). *infed.ru*

УДК 378

С. Д. Коткин

S. D. Kotkin

Коткин Сергей Дмитриевич, к. п. н., доцент кафедры ИОТД, КГПИ ФГБОУ ВО «КемГУ», г. Новокузнецк, Россия.

Kotkin Sergey Dmitrievich, Ph.D., Associate Professor, Kuzbass Humanitarian Pedagogical Institute of Kemerovo State University, Novokuznetsk, Russia.

ИСПОЛЬЗОВАНИЕ «ЧЕРЕПАШЬЕЙ ГРАФИКИ» ДЛЯ ОРГАНИЗАЦИИ ИЗУЧЕНИЯ ШКОЛЬНИКАМИ СРЕДНЕГО ЗВЕНА В УЧРЕЖДЕНИЯХ ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ PYTHON

USING «TURTLE GRAPHICS» TO ORGANIZE THE STUDY OF PYTHON PROGRAMMING BY MIDDLE SCHOOL STUDENTS IN INSTITUTIONS OF ADDITIONAL EDUCATION

Аннотация. *В статье рассматривается организация вводного курса программирования на языке Python. Сравнивается использование для этих целей языка Scratch, игровых движков начального уровня и библиотеки черепашьей графики. Даются рекомендации по преодолению проблем, возникающих в ходе использования в учебном процессе библиотеки черепашьей графики.*

С. Д. Коткин 2025-02-25

Annotation. *The article examines the design and implementation of an introductory programming course using Python. It compares the effectiveness of Scratch, entry-level game engines, and the turtle graphics library as tools for teaching programming. The paper also provides recommendations for addressing challenges that arise when integrating the turtle graphics library into educational settings.*

Ключевые слова: *обучение программированию, дополнительное образование, среднее звено школы, черепашья графика.*

Keywords: *programming training, further education, secondary school, Python, turtle graphics.*

В начале пути обучения программированию перед учащимися возникает множество проблем: они должны выучить новый язык, сформировать абстрактные понятия, научиться осуществлять декомпозицию задач, освоить методы поиска и исправления ошибок в программе. Всё это учащиеся вынуждены постигать одновременно. Когда же они натыкаются на примеры программ, показывающие те или иные возможности языка, то иногда начинают сомневаться уже в своих способностях. Поэтому даже ученики, хорошо мотивированные в начале занятий, могут через какое-то время разочароваться в выбранном предмете, так как созданный ими образ изучения программирования в связи с практически полным отсутствием адекватных представлений о деятельности программиста (к сожалению, свою роль в этом играет и современная поп-культура, демонстрирующая фантасмагорические образы программиста как некоего хакера, способного очень быстро решать практически любые «компьютерные» задачи) не соответствует тому, что происходит на занятиях и тем задачам, которые нужно решать дома.

Для того чтобы облегчить ученику начальный этап изучения программированию, применяются различные дидактические методики обучения, различающиеся, в том числе, используемыми в качестве дидактических средств программными инструментами. Одним из таких дидактических средств является язык программирования Scratch [1]. Это визуальный язык программирования, позволяющий описывать поведение и взаимодействие графических персонажей, спрайтов, с помощью программ, составляемых из графических блоков. Данный язык доступен как в виде веб-приложения, так и в виде автономной программы для персонального компьютера. В [2] указывается, что Scratch позволяет осваивать базовые понятия программ, такие как математические и строковые операции, последовательность команд, цикл, ветвление, обработку событий, параллелизм, скалярные данные и списки. Также авторы указывают, что в ходе работы над проектом учащиеся знакомятся с такими «практиками вычислительного мышления» (Computational thinking practices) как итеративность и инкрементность, тестирование и отладка, повторное использование и ремиксинг (remixing), абстрагирование и модульность. Помимо классической реализации Scratch существуют и предметно-ориентированные реализации этого языка, в частности, для программирования микроконтроллеров [3] и для создания мобильных приложений [4]. Также имеется среда, позволяющая транслировать программы, написанные на Scratch, в код на других языках программирования, таких, как JavaScript, Python и т.п. [5].

Ещё один подход заключается в использовании для обучения программированию сред разработки, предназначенных для создания игр. К таким средам можно отнести как проприетарные [6, 7], так и открытые [8] среды. В них языки Scratch, Lua или собственный язык, похожий по синтаксису на Python, используются для написания двумерных и трехмерных игр.

Третий подход заключается в использовании обучению программированию обычных языков программирования, но с акцентом на создание графики, используемой для организации обратной связи от компьютерной программы к человеку и для поддержания мотивации к обучению. При этом могут использоваться различные языки, хотя в последние годы чаще всего применяется язык Python, а также разные библиотеки, такие как встроенная в Python библиотека черепашьей графики а также библиотека Pygame или её модификация для начинающих, Pygame Zero [9].

Использование языка Scratch хорошо показало себя в обучении школьников младшего звена. Визуальный интерфейс позволяет сосредоточиться на структуре программы, а не на синтаксисе языка. Этот же визуальный интерфейс используется для предотвращения некоторых ошибок, заключающихся в использовании несочетаемых языковых конструкций, например, использовании циклов как части оператора присваивания и т.п. Школьники могу разрабатывать проекты инкрементно, тут же получая обратную связь. Кроме того, создание динамичных визуальных сцен способствует повышению мотивации.

Также Scratch удобен для создания прототипов при программировании микроконтроллеров устройств [3, 10] или для создания так называемых минимально жизнеспособных продуктов для мобильных устройств [4].

Однако уже в среднем звене достоинства Scratch начинают оборачиваться его недостатками. Этот инструмент начинает восприниматься школьниками как детский, что не лучшим образом сказывается на мотивации. Проекты на Scratch весьма быстро становятся очень громоздкими при их усложнении. Отсутствие многих структур данных, имеющихся в современных языках, затрудняет реализацию сложных алгоритмов. А отсутствие отладчиков сильно затрудняет анализ программ. Сама же концепция конструктора программ из блоков хоть и позволяет быстро начать программировать, в дальнейшем приводит к более низкой производительности в написании кода по сравнению с современными средами разработки, используемыми обычными языками.

Использование в обучении программированию игровых движков, основанных на обычных, не визуальных, языках программирования, дает хорошие результаты тогда, когда учащиеся уже овладели основами программирования и им необходимо подобрать интересный проект, где они бы смогли познакомиться с некоторыми современными концепциями программирования. Однако, поскольку даже относительно простые игровые движки, такие, как Godot, обладают инструментарием, сходным с тем, что применяется в игровых движках, применяемых профессиональными командами разработчиков игр, накладные расходы на знакомство с ними делают их практически непригодными в начале процесса обучения программированию. Поэтому для обучения программированию школьников среднего звена более подходящим является применение графических библиотек, в частности, черепашьей графики.

Черепашья графика была разработана в 1967 году как часть языка программирования Logo, предназначенного для обучения детей программированию. Идея черепашьей графики заключается в использовании исполнителя, «черепашки», для построения геометрических фигур, для чего её отдаются команды, такие как движение вперед или назад на определенное расстояние, поворот на заданный угол, поднятие и опускание пера и прочее. Черепашья графика вместе с языками, в которых программа представляется в виде текста, а не графических блоков обладает перед Scratch определенными преимуществами. Учащиеся осваивают язык, имеющий практическое применение не только в качестве языка для обучения программированию, но и в таких сферах как веб-разработка, анализ данных, машинное обучение, искусственный интеллект, научных исследованиях, анализе финансовых рынков, системном администрировании, разработке игр, разработке мобильных приложений. При этом в зависимости от сложности задач они могут использовать различные по возможностям и по сложности освоения IDE, начиная с обладающей лишь базовыми возможностями Wing 101 [11] и заканчивая PyCharm [12], имеющей встроенную поддержку различных фреймворков, систем контроля версий, баз данных и прочие возможности для профессиональной разработки. Библиотека turtle, реализующая в языке Python черепашью графику, обеспечивает традиционные возможности черепашьей графики: результаты последовательно выполняемых черепашкой команд тут же отображаются на экране как в виде следа, оставляемого черепашкой, так и в виде состояния самой черепашки, включая положение, направление и цвет. При этом действия черепашки выполняются с регулируемой скорость, достаточно низкой, чтобы можно было наблюдать за последовательным появлением всех промежуточных результатов. С помощью библиотеки можно выполнять простые геометрические построения. Однако она естественным образом интегрируется с прочими возможностями Python, что позволяет строить сложные объекты, включая фракталы. А поскольку библиотека turtle в языке python предоставляет

расширенные по сравнению с базовой реализацией возможности, такие как возможность создавать множество исполнителей-черепашек и возможность обрабатывать события от клавиатуры и мыши, то помимо статических изображений можно создавать и анимированные интерактивные сцены.

Есть у такого к обучению программированию подхода и недостатки. В связи, в первую очередь с тем, что группы разновозрастные, у учащихся в разной мере сформировано абстрактное мышление, из-за чего они плохо воспринимают такие понятия, как переменные, функции, циклы. Ученики зачастую могут объяснить значение таких конструкций в конкретном, ранее разобранном месте кода. Но перенести это понимание в другой контекст, или придумать, как использовать эти абстракции в новых задачах им не по силам. Одним из вариантов решения такой проблемы является групповая работа над заданием, где учащиеся, совместно выполняющие задание, прислушиваясь к рекомендациям педагога, распределяют задачи по уровням абстракции в соответствии с текущим уровнем развития у них умения мыслить абстрактно.

Следующей проблемой является слабое знание английского языка, чья лексика лежит в основе Python и его библиотек. Поэтому даже заучив перевод основных операторов, коих не так много, ученики делают множество опечаток в иностранных словах, и так как они плохо помнят написание слов-операторов, на поиск опечаток учащиеся затрачивают несоразмерно много времени, сличая свой код не с запечатленным в памяти визуальным образом слов, но с последовательностью букв, записанных на доске или в тетради. Можно смягчить эту проблему, выбирая название переменных и имена функций в виде транслителированных русских слов, или даже в виде слов, написанных на русском языке кириллицей, что Python, в силу использования для представления текста программ стандарта Unicode позволяет делать. Можно даже пойти ещё дальше, заменяя стандартные имена библиотечных функций и классов алиасами, представляющими собой их перевод на русский язык. Но такие крайности представляются тупиковыми вследствие отсутствия адаптации к, в целом, англоязычной среде области разработки программ. Поэтому более подходящим является изучение английского языка как технического, позволяющего понимать основные термины и читать документацию. Тем более, что в Новокузнецке имеется соответствующая программа дополнительного образования, по которой дети могут обучаться бесплатно.

Ещё одной проблемой являются ограниченные познания в области геометрии, что определяется школьной программой. В рамках обучения программированию в системе дополнительного образования трудно обучить ребёнка геометрии как целостной индуктивной системе, но можно снять остроту проблемы, рассказывая об отдельных понятиях, свойствах, соотношениях, не приводя формальных доказательств, а также давая готовые решение геометрических задач возникающих в ходе написания программ.

Ещё одна проблема в организации изучения программирования через изучение черепашьей графики связана с тем, что в черепашьей графике нет простых задач, требующих для своего решения ветвления. Можно вводить несколько искусственные условия, объединяя черепашью графику и получение ввода с консоли. Но такого рода задачи плохо воспринимаются детьми, так как требуют мультимодального (визуальная и визуально-дигитальная сигнальные системы) взаимодействия с программой. Поэтому, как правило, ветвление осваивается значительно позже, чем при решении обычных задач, когда учащиеся начинают строить параметризируемые объекты, обрабатывать состояние черепашки и создавать программы, управляемые событиями.

Среди трёх рассмотренных подходов – программирования в визуальной среде Scratch, разработки на базе игровых движков и работы с черепашьей графикой – именно последний вариант демонстрирует наибольший потенциал в рамках вводного курса программирования для учащихся среднего звена школ в системе дополнительного образования.

Список литературы

1. Scratch : [сайт] / MIT Media Lab. – URL : https://scratch.mit.edu/ (дата обращения: 20.02.2025). – Текст : электронный.
2. Brennan, K., Resnick, M. New Frameworks for Studying and Assessing the Development of Computational Thinking // Proceedings of the 2012 Annual Meeting of the American Educational Research Association. – 2012. – P. 1–25. – URL: https://scratched.gse.harvard.edu/ct/files/AERA2012.pdf (дата обращения: 22.02.2025). – Текст : электронный.
3. Autodesk Tinkercad / Autodesk, Inc, – 2025. – URL: https://www.tinkercad.com/ (дата обращения: 15.01.2025). – Текст : электронный.
4. Mit App Inventor / разработчик Massachusetts Institute of Technology. – 2025. – URL: https://appinventor.mit.edu/ (дата обращения: 20.01.2025). – Текст : электронный.

5. Blockly. Google for Developers : [сайт]. – URL : https://developers.google.com/blockly?hl=ru (дата обращения: 20.01.2025). – Текст : электронный.
6. Roblox. Roblox Corporation : [сайт]. – URL: https://create.roblox.com/ (дата обращения: 20.02.2025). – Текст : электронный.
7. Code.org : [сайт]. – URL: https://code.org/ (дата обращения: 18.02.2025). – Текст : электронный.
8. Godot Engine / J. Linietsky, A. Manzur and contr. // Godot : [сайт]. – 2007–2025. – URL: https://godotengine.org/ (дата обращения: 18.02.2025). – Текст : электронный.
9. Pygame Zero / D. Pope // Godot : [сайт]. – 2007-2025. – URL: https://godotengine.org/ (дата обращения: 14.02.2025). – Текст : электронный.
10. Microsoft MakeCode : [сайт]. – 2025. – URL: https://www.microsoft.com/en-us/makecode (дата обращения: 16.02.2025). – Текст : электронный.
11. Wing 101 Reference Manual / Wingware : [сайт]. – 1999–2024 – URL : https://wingware.com/pub/wing-101/10.0.8.0/doc/wing-101-manual-en-a4.pdf (дата обращения: 19.02.2025). – Текст : электронный.
12. PyCharm Professional vs. Community Edition : [сайт]. / JetBrains s.r.o. : [сайт]. – 2000-2025. – URL : https://www.jetbrains.com/pycharm/editions/ (дата обращения: 19.02.2025). – Текст : электронный.