МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
федеральное государственное бюджетное образовательное учреждение высшего образования
КРАСНОЯРСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им.В.П.АСТАФЬЕВА
(КГПУ им.В.П.Астафьева)

Институт/факультет Институт математики, физики и информатики
(полное наименование института/факультета/филиала)

Выпускающая кафедра Базовая кафедра информатики и информационных технологий в образовании
(полное наименование кафедры)

Станковский Данила Вадимович

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Тема **Мобильное приложение как средство обучения основам алгоритмизации детей младшего школьного возраста**

Направление подготовки 44.03.05 Педагогическое образование
(код и наименование направления)

Профиль физика и информатика
(наименование профиля для бакалавриата)

ДОПУСКАЮ К ЗАЩИТЕ
Заведующий кафедрой
д.п.н., профессор Пак Н.И.
(ученая степень, ученое звание, фамилия, инициалы)

(дата, подпись)

Руководитель
к.п.н., Сокольская М.А.
(ученая степень, ученое звание, фамилия, инициалы)

Дата защиты 28. 06. 2018

Обучающийся Станковский Д.В.
(фамилия, инициалы)

(дата, подпись)

Оценка
(прописью)

Красноярск 2018

Оглавление

Введение

Актуальность. В настоящее время почти все сферы жизнедеятельности зависят от информационных технологий. Для обеспечения бесперебойной работы технических систем требуется программное обеспечение, следовательно, одной из самых востребованных профессий является профессия программиста.

В 2014 году по всему миру насчитывалось около 18,5 миллионов программистов. Тем не менее, в мире наблюдается их острая нехватка. Например, в России количество рабочих мест для программистов примерно в 4 раза больше, чем таких кадров [23]. Поэтому государство через ФГОС предъявляет одно из требований к выпускникам основной школы: знать хотя бы один язык программирования и общие принципы построения программ.

Начинать изучать программирование нужно уже в начальной школе, это не только не навредит ребёнку, но ещё и будет полезно для развития его памяти, внимания и самоконтроля [3].

В сложившейся практике, впервые дети знакомятся с программированием в основном на занятиях по робототехнике. Однако учителям не хватает качественного методического обеспечения для пропедевтики программирования и алгоритмизации. Одним из средств упрощающих задачу раннего обучения детей основам алгоритмизации и программирования является мобильное приложение. Данная работа предлагает мобильное приложение, представляющее собой игру, которая:

- имеет сюжет и предполагает построение игроком алгоритма действий для игрового персонажа;
- соответствует возрастным особенностям обучающихся;
- включает в себя методические рекомендации для учителей информатики и преподавателей робототехники по использованию данной игры на уроках.

Объект исследования: процесс обучения основам алгоритмизации детей младшего школьного возраста.

Предмет исследования: мобильное приложение как средство обучения детей младшего школьного возраста основам алгоритмизации на занятиях по робототехнике.

Цель исследования: разработать мобильное приложение, которое позволит обучить детей младшего школьного возраста основам алгоритмизации.

Ведущие задачи исследования:

1. Изучить и проанализировать научно-педагогическую и методическую литературу по данной проблеме с целью выделения целей, особенностей и условий обучения младших школьников алгоритмизации и программированию в рамках занятий по робототехнике;
2. Проанализировать существующие программные продукты с целью выявления положительных и отрицательных сторон обучающих приложений, предназначенных для младших школьников и определить требования к содержательной и графической составляющей приложения с учетом возрастной группы;
3. Изучить средства, инструменты и технологии создания мобильных приложений;
4. Разработать игру, в которой игровой персонаж будет выполнять алгоритм действий, построенный игроком, будет сюжет и несколько игровых миссий для поддержания интереса детей к игре, все базовые алгоритмические конструкции, а также максимально простая и наглядная система команд исполнителя;
5. Разработать методические рекомендации для учителей информатики и преподавателей робототехники по использованию данной игры при обучении детей основам алгоритмизации.

Теоретико-методологические основания исследования:

Положения о психолого-физиологических особенностях младших школьников: Волков Б.С.; Гамезо М.В.; Давыдов В.В.; Зимняя И.А.; Кулагина И.Ю.; Цукерман Г.А.; положения о раннем обучении программированию: Козлова, Л.Л.; Козлова С.А.; Крылова О.Н.; Первин Ю.А.; Сорокина Т.Е; Рыжиков С.; Давайте учить детей программированию; методики обучения робототехнике младших школьников: Филиппов С.А.; Шимов И.В.; Berns K.; средства и инструменты создания мобильных приложений: Стивен Хубер; руководство по Unity, руководство по программированию на C#, центр правил для разработчиков Google Play; как выполнять сохранение и загрузку игры в Unity; карты тайлов; основы создания 2D персонажа в Unity; пример создания простой 2D игры для Android; работа с корутинами в Unity; дизайн мобильного приложения.

Для решения поставленных задач использовались следующие ***методы исследования:*** теоретические (изучение и анализ педагогической, психологической, методической и предметной литературы по теме исследования, анализ теоретических и эмпирических данных, изучение и обобщение педагогического опыта, сравнительный анализ, классификация); эмпирические (наблюдение, анкетирование, беседа, тестирование).

Апробация и внедрение результатов. Материалы данного исследования были представлены на XVIII международном научно-практическом форуме студентов, аспирантов и молодых учёных «Молодёжь и наука XXI века» в рамках работы секции «Актуальные проблемы информатики и компьютерных наук» (приложение В).

Работа (61 стр.) состоит из введения, двух глав, заключения, библиографического списка (32 источников) и приложений. Работа содержит 19 рисунков и 2 таблицы.

Глава 1. Обучение основам алгоритмизации детей младшего школьного возраста в рамках занятий по робототехнике

1.1. Психолого-педагогические особенности обучения детей 6-9 лет основам построения алгоритмов

«Педагогические эксперименты по обучению программированию и информатике детей младшего школьного и даже дошкольного, возраста за рубежом, и в нашей стране показали, что дети младшего школьного возраста не только могут усвоить приемы программирования, но быстрее, прочнее и естественнее осваивают фундаментальные понятия информатики, которые способствуют формированию мировоззрения ребенка» [16]. Но прежде чем говорить о том, как можно реализовывать обучение программированию младших школьников, стоит проанализировать психолого-педагогические особенности детей этого возраста.

Младший школьный возраст называют вершиной детства. Ребенок сохраняет много детских качеств — легкомыслие, наивность, взгляд на взрослого снизу вверх. Но он уже начинает утрачивать детскую непосредственность в поведении, у него появляется другая логика мышления. Учение для него — значимая деятельность. В школе он приобретает не только новые знания в умения, но и определенный социальный статус. Меняются интересы, ценности ребенка, весь уклад его жизни [14].

Память

Дети младшего школьного возраста обладают очень развитой механической памятью. Одна из задач учителя в начальных классах — научить детей использовать определенные мнемонические приемы. Это, прежде всего, деление текста на смысловые части, прослеживание основных смысловых линий, выделение смысловых опорных пунктов или слов, возвращение к уже прочитанным частям текста для уточнения их

содержания, мысленное припоминание прочитанной части и воспроизведение вслух и про себя всего материала, а также рациональные приемы заучивания наизусть. В результате, учебный материал понимается, связывается со старым и включается в общую систему знаний, имеющуюся у ребенка. Такой осмысленный материал легко «извлекается» из системы связей и значений и воспроизводится [14].

Одним из мнемонических приёмов для развития памяти ребёнка является составление плана чего-либо. Составление алгоритма – это и есть составление плана действий. Поэтому, можно сделать вывод, что обучение детей программированию и основам алгоритмизации является хорошим средством для развития памяти ученика.

Младшие школьники склонны дословно воспроизводить то, что запомнили. Они непроизвольно запоминают учебный материал, вызывающий у них интерес, преподнесенный в игровой форме, связанный с яркими наглядными пособиями или образами-воспоминаниями. Следовательно, среда, в которой ребёнок будет составлять алгоритм, должна быть яркой и привлекательной. От ручки и бумаги при современном уровне развития технологий можно отказаться вовсе.

Внимание

В младшем школьном возрасте развивается внимание. Без достаточной сформированной этой психической функции процесс обучения невозможен. По сравнению с дошкольниками младшие школьники гораздо более внимательны. Они уже способны концентрировать внимание на неинтересных действиях, но у них все еще преобладает *непроизвольное* внимание. Для них внешние впечатления — сильный отвлекающий фактор, им трудно сосредоточиться на непонятном сложном материале. Затруднены распределение внимания и его переключение с одного учебного задания на другое. Поэтому, на занятии по робототехнике, учитель должен привлекать внимание учеников к учебному материалу, стараться удерживать его длительное время, переключать с одного вида работы на другой.

В учебной деятельности развивается произвольное внимание ребенка. Первоначально следуя указаниям учителя, работая под его постоянным контролем, он постепенно приобретает умение выполнять задания самостоятельно — сам ставит цель и контролирует свои действия. Контроль над процессом своей деятельности и есть, собственно, произвольное внимание ученика [14].

Развитие мышления и интереса у младшего школьника приводит к развитию его внимания, формируется внимательность как свойство личности. Произвольное внимание развивается вместе с развитием его свойств. Сосредоточенность и устойчивость внимания младшего школьника развиваются в работе, требующей большой умственной активности. Это происходит тогда, когда [2]:

1. Учащийся занят умственной деятельностью (реализация анализа, сравнение, выделение существенного, классификация предметов и других видов мыслительных операций);
2. Изучаемый материал доступен пониманию учащихся;
3. Изучаемый материал вызывает сильные переживания;
4. Изучаемый материал интересен учащимся и соответствует их потребностям.

Из этого критерия следует, что задания на составление алгоритм должны быть доступными и интересными.

Мышление

Доминирующей функцией в младшем школьном возрасте становится мышление**.** Благодаря этому интенсивно развиваются, перестраиваются сами мыслительные процессы и, с другой стороны, от интеллекта зависит развитие остальных психических функций.

При организации учебной деятельности детей младшего школьного возраста очень важно учитывать то, что у них преобладающим является

конкретно-образное мышление. Это самый подходящий возраст для развития аналитических способностей и логического мышления. На это и должна быть направлена работа учителя, по мнению Гамезо М. В. [3].

Успешное развитие абстрактного мышления обеспечивается на основе перевода внешних действий в план внутренний. При этом важно обучить планированию своей работы и нахождению путей и средств реализации поставленной цели [2]. Обучение планированию, составлению чертежей, пооперационным действиям развивает у младших школьников произвольность психических процессов, способность действовать последовательно, целеустремленно [6]. Все эти задачи решаются при обучении детей основам построения алгоритмов.

Очень важна форма, в которой осуществляется учебная деятельность младших школьников. Наиболее эффективной является кооперация детей, вместе решающих одну учебную задачу. Учитель, организуя совместную работу в группах учеников, организует тем самым их деловое общение друг с другом. При групповой работе повышается интеллектуальная активность детей, лучше усваивается учебный материал. Развивается саморегуляция, поскольку дети, контролируя ход совместной работы, начинают лучше оценивать свои возможности и уровень знаний. Что касается собственно развития мышления, то кооперация учеников невозможна без координации их точек зрения, распределения функций и действий внутри группы, благодаря чему у детей формируются соответствующие интеллектуальные структуры [6]. То есть, при решении задач на построение алгоритма, можно и нужно объединять детей в группы.

Несмотря на то, что в этот период большое значение имеет наглядно-образное мышление, непосредственно воспринимаемое ребенком уже не мешает ему рассуждать и делать правильные выводы. Поэтому после решения алгоритмических задач, необходимо проводить с детьми рефлексию, для осознания ребёнком оснований собственных действий.

Игровая деятельность

У ребёнка младшего школьного возраста ещё есть потребность в игре. Поэтому в первое время пребывания в школе существенным фактором для пробуждения интереса к обучению, для облегчения сложной учебной деятельности является введение игровой ситуации. То есть, игра стимулирует лучшее запоминание и понимание изучаемого материала, а также игра способствует повышению мотивации и позволяет обучаемому комплексно использовать органы чувств, при восприятии информации, а также самостоятельно и неоднократно воспроизводить ее в новых ситуациях [3].

Дети в 5-6 лет начинают получать удовольствие не только от процесса игры, но и от результата, т.е. выигрыша. В играх по правилам, характерных для старшего дошкольного и младшего школьного возрастов, выигрывает тот, кто лучше освоил игру. Ребенок стремится отработать учебные действия, учится успешно выполнять отдельные, может быть, не слишком интересные сами по себе действия. В игровой мотивации смещается акцент с процесса на результат; кроме того, развивается мотивация достижения. Сам ход развития детской игры приводит к тому, что игровая мотивация постепенно уступает место учебной, при которой действия выполняются ради конкретных знаний и умений, что, в свою очередь, дает возможность получить одобрение, признание взрослых и сверстников, особый статус [14].

Мотивационная сфера

В начале своей школьной жизни, имея внутреннюю позицию школьника, он хочет учиться. Причем учиться хорошо, отлично. Среди разнообразных *социальных мотивов учения,* пожалуй, главное место занимает мотив получения высоких отметок. Высокие отметки для маленького ученика — источник других поощрений, залог его эмоционального благополучия, предмет гордости. Когда ребенок успешно учится, его хвалят и учитель, и родители, его ставят в пример другим детям, его особенно значительные успехи отмечают сладким пирогом или подарком, в зависимости от семейных традиций. Более того, в классе, где мнение

учителя — не просто решающее, но единственное авторитетное мнение, с которым все считаются, высокие отметки и прочие оценки обеспечивают соответствующий статус.

Отметка — реально действующий мотив; чтобы получить высокую отметку или похвалу, ребенок готов немедленно сесть заниматься и старательно выполнить все задание. Абстрактное для него понятие долга или далекая перспектива получить хорошее образование непосредственно побуждать его к учебной работе не могут. Поэтому очень важно и при изучении ребёнком основ алгоритмизации, хвалить его, ставить отметки, подчеркивать его достижения.

Воля

В процессе обучения ребенку необходимо постоянно действовать в соответствии с требованиями учителя или по образцу. Часто учебные действия, совершаемые им, непосредственно его потребностям не удовлетворяют и требуют применения волевых усилий. Так, ученик, желая заслужить похвалу учителя, получить хорошую оценку, выполняет упражнение, направленное на научение правильному написанию букв, или учит таблицу умножения, которая сама по себе его не привлекает [3].

Очевидно, что обучать учеников начальной школы программированию можно и нужно, это обучение позволяет решить множество развивающих задач. Но преподаватели робототехники отмечают, что программирование роботов вызывает у детей большие затруднения. Что не удивительно! Возрастные особенности детей младших классов не дадут им легко понять программирование. Начинать изучение необходимо с чего-то более легкого, то есть, необходима пропедевтика и изучение основных алгоритмических конструкций на простых примерах. Однако, для реализации этого, учителям не хватает качественного методического обеспечения.

Сорокина Т. Е. в своей статье, посвященной пропедевтике программирования, говорит о необходимости инструмента (программной среды, интуитивно понятной и визуально привлекательной), с помощью

которого можно сделать программирование общедоступным и обеспечить раннее знакомство детей с ним и предлагает обучать программированию с помощью «Scratch», начиная с пятого класса [25].

В связи с выше сказанным, наиболее подходящим инструментом пропедевтики программирования представляется обучающая мобильная игра. Использование такой игры может помочь решить задачи привлечения школьников к занятию программированием, а также мотивировать их к выбору профессии в отрасли информационных технологии. В данной работе в качестве такого инструмента рассматриваются игры обучающие основам алгоритмизации и программированию, которые можно было бы использовать в рамках занятий по робототехнике.

1.2. Цели и условия обучения основам алгоритмизации младших школьников на занятиях по робототехнике

В ФГОС начального общего образования описаны требования к результатам изучения предметной области «Математика и информатика». Требования, которые можно реализовать на занятиях по робототехнике – это «овладение основами логического и алгоритмического мышления» и «умение действовать в соответствии с алгоритмом и строить простейшие алгоритмы» [27]. Эти умения также включены в перечень необходимых умений при формировании ИКТ-компетенции ученика, описанных в этом документе. Так как любая робототехническая конструкция выступает в роли исполнителя программы (алгоритма), то робототехника выступает отличным средством формирования у детей этих умений.

Образовательная робототехника также является хорошим средством вовлечения в процесс инженерного творчества детей, начиная с младшего школьного возраста. На занятиях дети не только собирают роботов, развивая навык моделирования (который включен в перечень планируемых результатов примерной образовательной программы), но и программируют

их, развивая, как раз, алгоритмическое мышление и умение строить простейшие алгоритмы.

Алгоритмический стиль мышления необходим не только будущим программистам, но и каждому человеку, живущему в современном информационном обществе. Этот стиль мышления подробно описан в статье Первина Ю.А. [16]. Вот некоторые его черты:

1. Умение планировать структуру действий, необходимых для достижения цели при помощи фиксированного набора средств;
2. Умение строить информационные модели для описания объектов и систем;
3. Умение организовать поиск информации, нужной для решения поставленной задачи;
4. Умения четко излагать свои мысли, чтобы они были понятны любому собеседнику, в том числе и компьютеру;
5. Навык своевременного обращения к компьютеру при решении задач из разных предметных областей (если его не привить, то человек может и не додуматься использовать компьютер);
6. Умение использовать информационные ресурсы и средства Интернет-коммуникаций;
7. Технические навыки взаимодействия с компьютером, в частности, умение работать клавиатурой, мышью, организовывать обмен информацией между компьютером и современной цифровой мультимедийной периферийной аппаратурой.

Формирование этого стиля мышления теперь возложено на массовую общеобразовательную школу, и эта задача может быть решена в рамках занятий по робототехнике.

Содержание обучения в процессе изучения программирования на занятиях по робототехнике включает в себя довольно сложные темы. Вот несколько из них:

1. Основные команды языка;
2. Структура и технология построения программы;
3. Команды действия;
4. Команды ожидания времени;
5. Движение робота на заданное время;
6. Программная реализация реакции на датчик;
7. Базовые конструкции языка при решении задач;
8. Использование циклической конструкции;
9. Конструкция ветвления в среде программирования;
10. Алгоритм движения по черной линии с одним/двумя датчиками света;
11. Пропорциональный регулятор: подбор коэффициента;
12. Разработка алгоритма управления с использованием подпрограмм.

Темы определяют одно из важнейших условий обучения программированию в курсе робототехники: подготовленность педагога. Педагог должен:

1. Знать основы конструирования и механики (устойчивость и прочность конструкции, зубчатые и ременные передачи, рычаги, колеса, оси, блоки, преобразование движения);
2. Уметь программировать любое робототехническое устройство, с помощью заранее выбранной среды программирования;

3. Уметь грамотно подбирать формы и методы обучения (мониторинг существующего мирового и российского опыта использования образовательных конструкторов в учебном процессе).

На выборе метода обучения остановимся подробнее. В примерной основной образовательной программе начального общего образования сказано о необходимости организации интеллектуальных и творческих соревнований, научно-технического творчества и проектно-исследовательской деятельности [19].

Также, практика использования образовательных конструкторов в процессе изучения робототехники показала, что наиболее эффективным и часто используемым в робототехнике методом обучения является метод проектов. Метод проектов имеет ряд преимуществ перед традиционными методами обучения. Он дает возможность организовать учебную деятельность учащегося, соблюдая баланс между теорией и практикой. В качестве тем проектов удачно вписывается подготовка учащихся к различным соревнованиям по робототехнике.

Существует множество различных соревнований, к которым могут быть допущены учащиеся разных уровней подготовки. Такие соревнования являются своеобразным смотром достижений, показателем уровня развития соответствующей учебной деятельности. Подобные соревнования направлены, прежде всего, на повышение уровня мотивации учащихся к занятиям.

К материально-техническим условиям обучения основам алгоритмизации, которые позволят решить поставленные учебные задачи, относятся, выбранные образовательным учреждением, конструктор и среда программирования. Среди имеющихся сегодня на рынке образовательных конструкторов бесспорным лидером, являются конструкторы, разработанные датской фирмой LEGO [31].

1.3. Электронные средства обучения программированию младших школьников на занятиях по робототехнике

Основой управления моделями является программирование блока с микрокомпьютером NXT. В нём имеется возможность использования электрических сервомоторов и лампочек, а также датчиков (звука, касания, расстояния, освещенности). С помощью программного обеспечения школьники могут планировать, тестировать и изменять набор последовательных команд, выполняемых роботом.

Существует несколько сред программирования, используемых для разработки программы управления микрокомпьютером NXT [29]:

1. Lego Mindstorms NXT Software (см. рис. 1):
   - язык программирования – NXT-G;
   - возраст – 8-12 лет.
2. Robolab 2.9.4 (см. рис. 2):
   - язык программирования – Robolab;
   - возраст – 8-16 лет.
3. RobotC for Mindstorms (см. рис. 3):
   - язык программирования – RobotC;
   - возраст 14-99 лет.

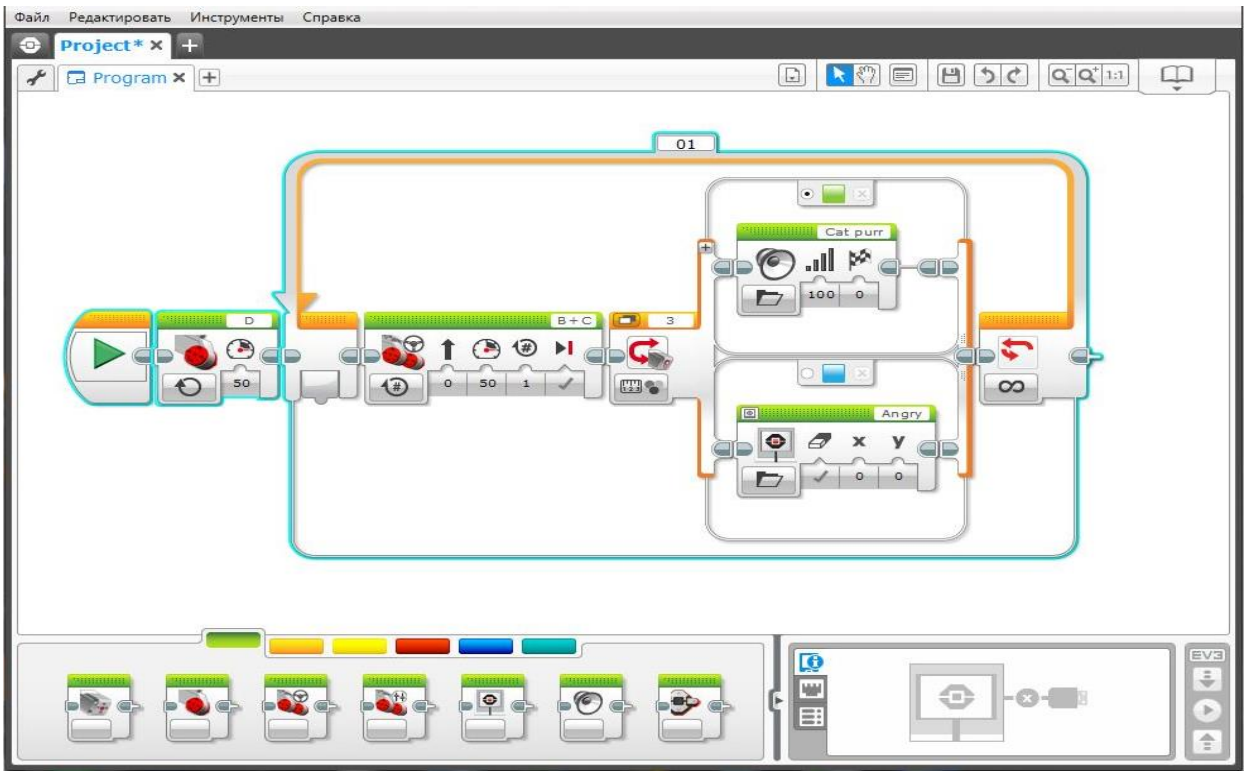


Рисунок 1. Визуальная среда программирования Lego Mindstorms NXT Software

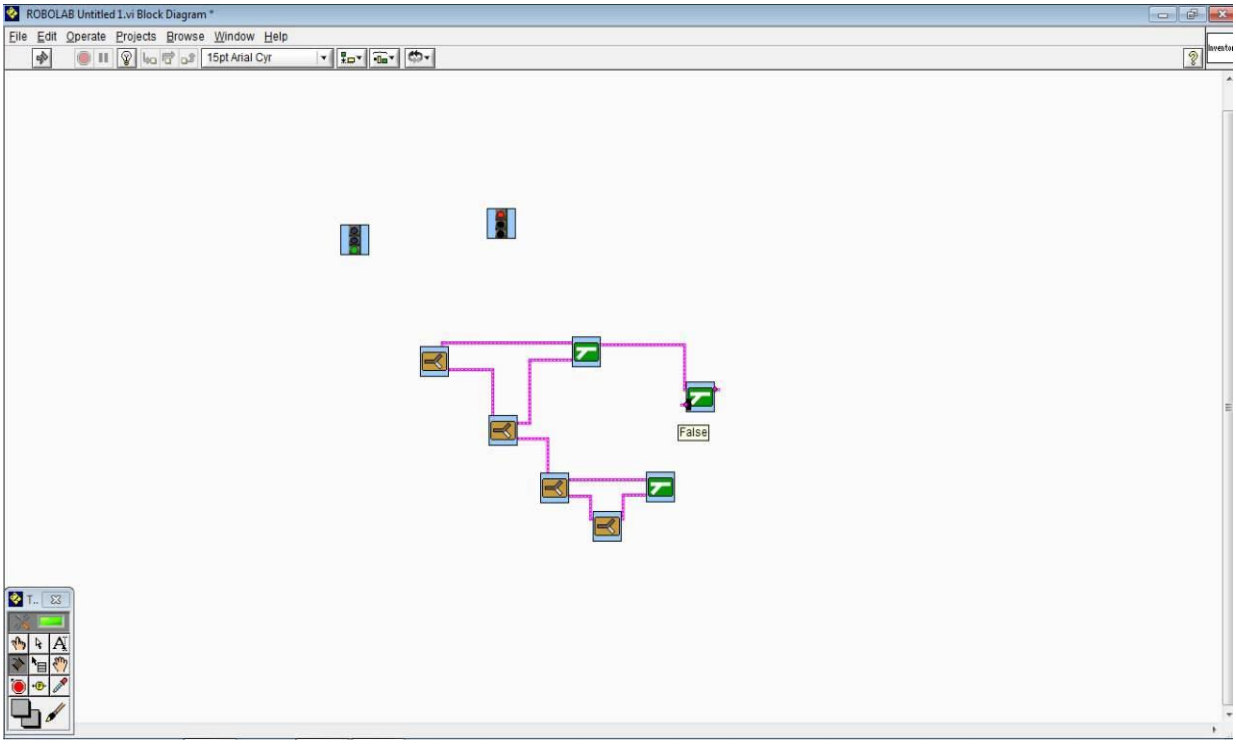


Рисунок 2. Визуальная среда программирования Robolab 2.9.4

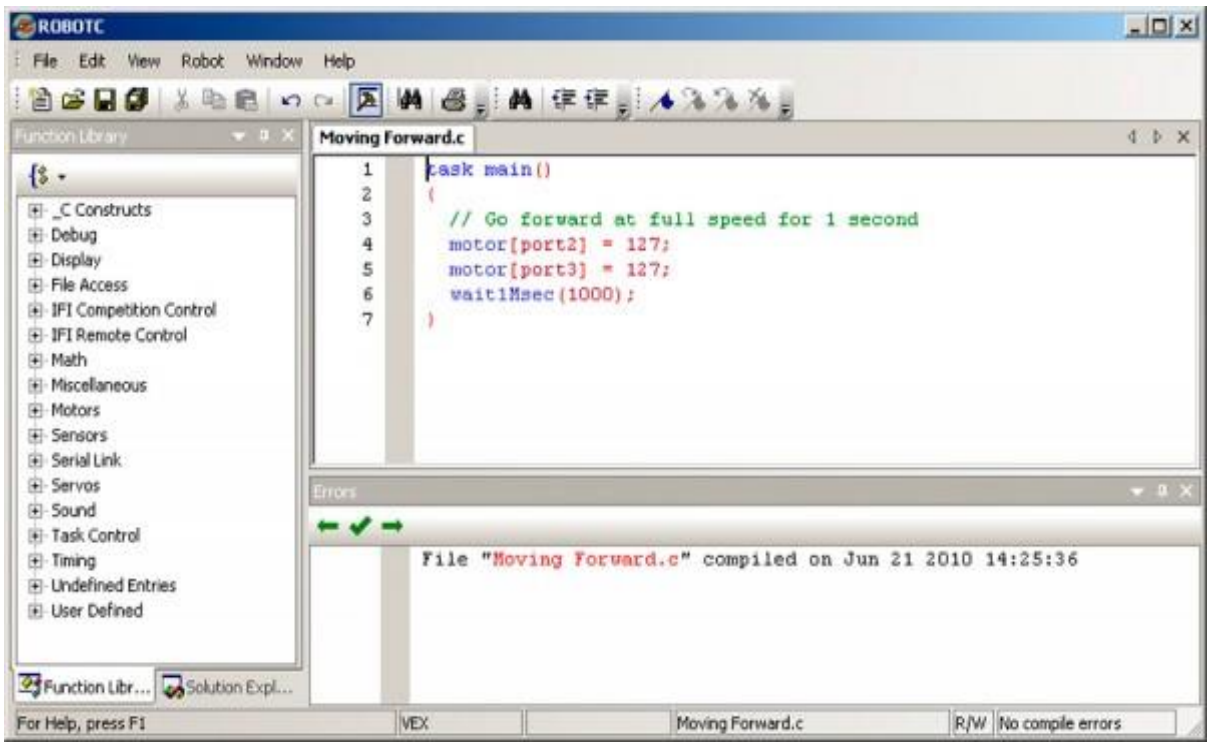


Рисунок 3. Визуальная среда программирования RobotC for Mindstorms

Выбор среды программирования может основываться на следующих факторах [31]:

- высокая скорость работы среды;
- высокая скорость загрузки программы;
- легкость в освоении;
- поддержка всех имеющихся датчиков;
- наличие помощи, форума технической поддержки;
- небольшая стоимость лицензии;
- маленький объем программ.

Ниже представлена сравнительная таблица 1 этих сред по приведённым выше критериям. Из таблицы видно, что наиболее подходящей средой для изучения программирования является Lego Mindstorms NXT Software.

Большинство школ и клубов используют на занятиях именно эту визуальную среду программирования.

Таблица 1. Таблица сравнения визуальных сред программирования робототехнических устройств

Параметр	NXT-G	Robolab	RobotC
Высокая скорость работы среды	+	+	+
Высокая скорость загрузки программы	+	+	+
Легкость в освоении	+	+	−
Поддержка всех датчиков	+	−	+
Наличие технической поддержки	+	−	+
Небольшая стоимость лицензии	+	+	−
Маленький объем программ	+	+	+

Но даже, не смотря на достоинства по сравнению с другими средами, как отмечают преподаватели робототехники, среда программирования NXT-G не может быть легко освоена детьми младшего школьного возраста. Из этого следует вывод, что нужна пропедевтика программирования, и что изучение основ алгоритмизации необходимо начинать с чего-то ещё более доступного. Например, с какой-нибудь обучающей игры.

1.4. Мобильные приложения для младших школьников, обучающие алгоритмизации: требования и критерии

Самая привлекательная обучающая игра для детей – это мобильная игра. Далее в параграфе будет представлен обзор мобильных приложений, которые можно использовать для обучения основам построения алгоритмов. Но прежде чем перейти к нему, необходимо определиться с правилами и

критериями, по которым можно однозначно сказать, подходит данное приложение для этих целей или нет.

При разработке приложения для детей необходимо придерживаться некоторых правил. Следующие правила были составлены на основе рекомендаций для детских приложений с сайта Google Play [17]:

1. При размещении приложения в сети Интернет необходимо указывать, что оно предназначено для детей, чтобы реклама в приложении и на сайте соответствовала возрасту пользователя;
2. В нем нельзя использовать авторизацию через аккаунт Google или другой подобный сервис;
3. Можно указать, что это приложение для всей семьи, чтобы родителям и учителям было проще его найти;
4. Приложение должно содержать контент, который можно показывать детям (например, нельзя показывать сцены насилия и кровопролития,);
5. Приложение не должно нарушать законов о защите личных сведений детей в Интернете (COPPA) и аналогичные законы тех стран, где они распространяются;
6. В приложение нельзя не в образовательных целях упоминать употребление табака и алкоголя;
7. Приложение не должно имитировать азартные игры;
8. Приложения для знакомств, а также содержащие рекомендации брачного или сексуального характера не могут быть предназначены для детей.

Но все эти правила довольно примитивны, и каждое детское приложение выполняет большинство из них. Поэтому на популярность и привлекательность приложения эти правила практически никак не влияют.

Для того чтобы приложение пользовалось спросом среди пользователей смартфонов младшего школьного возраста, при его разработке

можно придерживаться некоторых требований к его оформлению. В результате анализа рекомендаций разработчиков детских приложений были составлены следующие требования:

1. Интерфейс приложения должен быть раскрашен яркими цветами и иметь привлекающие внимание анимации;
2. В приложении должна играть весёлая музыка и, в качестве отклика на действие пользователя, должны воспроизводиться различные забавные звуки;
3. Игровые объекты должны быть оживлёнными, отражать свет и отбрасывать тень, а также иметь реалистичную анимацию движения;
4. Иконка приложения должна быть красочной и с изображением игрового персонажа или игрового объекта, чтобы дети понимали, что это игра, а не программа [7];
5. Не смотря на маленький размер пальцев, дети хуже контролируют точность нажатия, поэтому кнопки в приложении должны быть достаточно крупными, но не больше 22 мм. [26].

Безусловно, на Google Play уже есть приложения, которые можно использовать для пропедевтики программирования. У каждого из них есть свои достоинства и недостатки. Для анализа таких приложений, с учётом психологических особенностей детей целевого возраста и пожеланий преподавателей робототехники, были составлены следующие критерии. Приложение должно:

1. Поддерживать русский язык;
2. Иметь интуитивно понятный интерфейс;
3. Иметь открытые уровни;
4. Иметь доступные для детей уровни;

5. Иметь сюжет;
6. Не иметь ограничений на количество команд для робота;
7. Включать в себя все базовые алгоритмические конструкции;
8. Сопровождаться методическим обеспечением для учителей и преподавателей.

Ниже представлена сравнительная таблица 2 наиболее популярных детских приложений, предполагающих построение алгоритма действий для игрового персонажа.

Таблица 2. Таблица сравнения мобильных приложений, предполагающих построение алгоритма действий для игрового персонажа

Параметр	LightBot	Coddy: World on Algorithm	Robotizen
Поддержка русского языка	+	+	−
Понятность интерфейса	+	+	−
Открытость уровней	−	−	−
Доступность уровней	−	−	+
Наличие сюжета	−	−	+
Отсутствие ограничения на количество команд для робота	−	−	+
Использование всех базовых алгоритмических конструкций	−	−	−
Наличие методического материала	−	−	−

Самым популярным приложением в этой таблице является игра LightBot (см. рис. 4), платная версия которой имеет более 25 тысяч скачиваний, а бесплатная – более 500 тысяч. У этой игры отсутствует сюжет и имеется ограничение на количество действий, что увеличивает сложность и ограничивает свободу мышления ребенка. У Coddy схожие недостатки (см.

рис. 5). Но если в LightBot задания под силу детям от десяти лет, то в Coddy с некоторыми заданиями справится далеко не каждый взрослый.

Далее идёт игра Robotizen (см. рис. 6). Разработчик заявляет, что она рассчитана на возраст младшего школьника. Но отсутствие русского языка и обилие не понятных по назначению кнопок, в которых легко запутаться, делает его неподходящим для пропедевтики программирования.

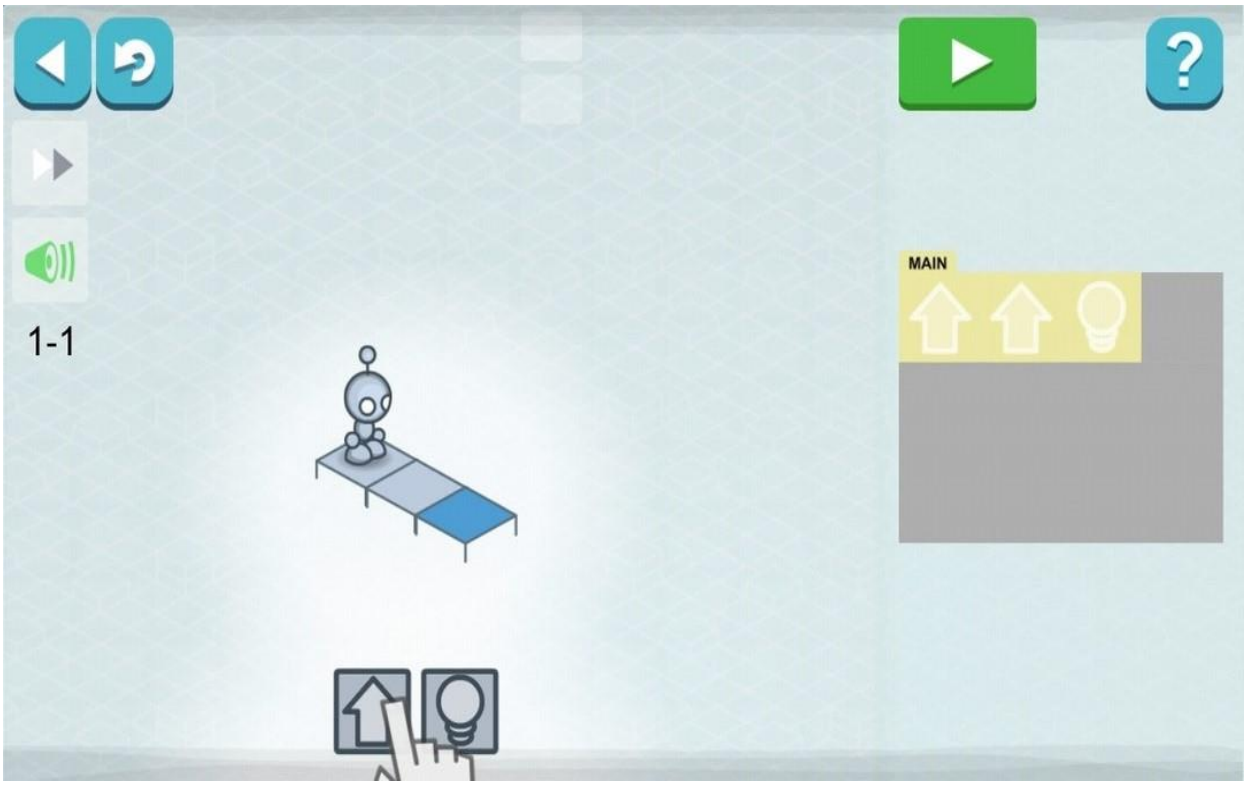


Рисунок 4. Основная сцена игры LightBot

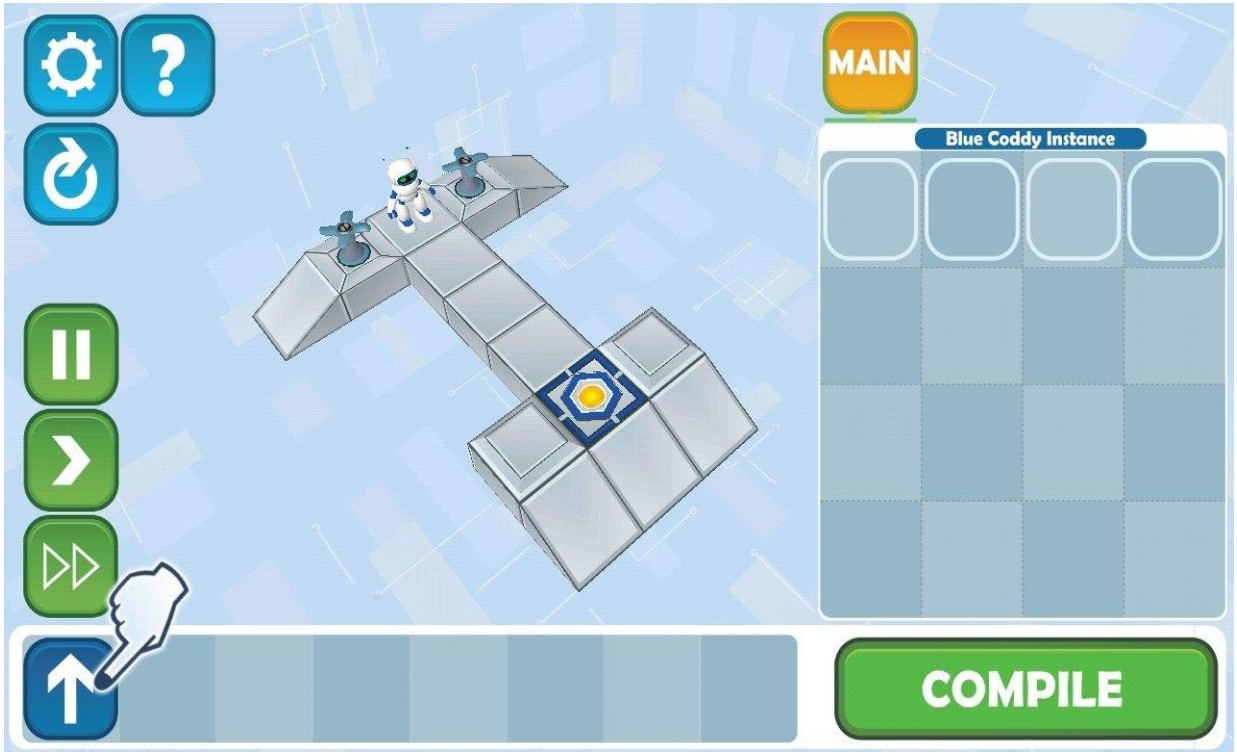


Рисунок 5. Основная сцена игры Coddy

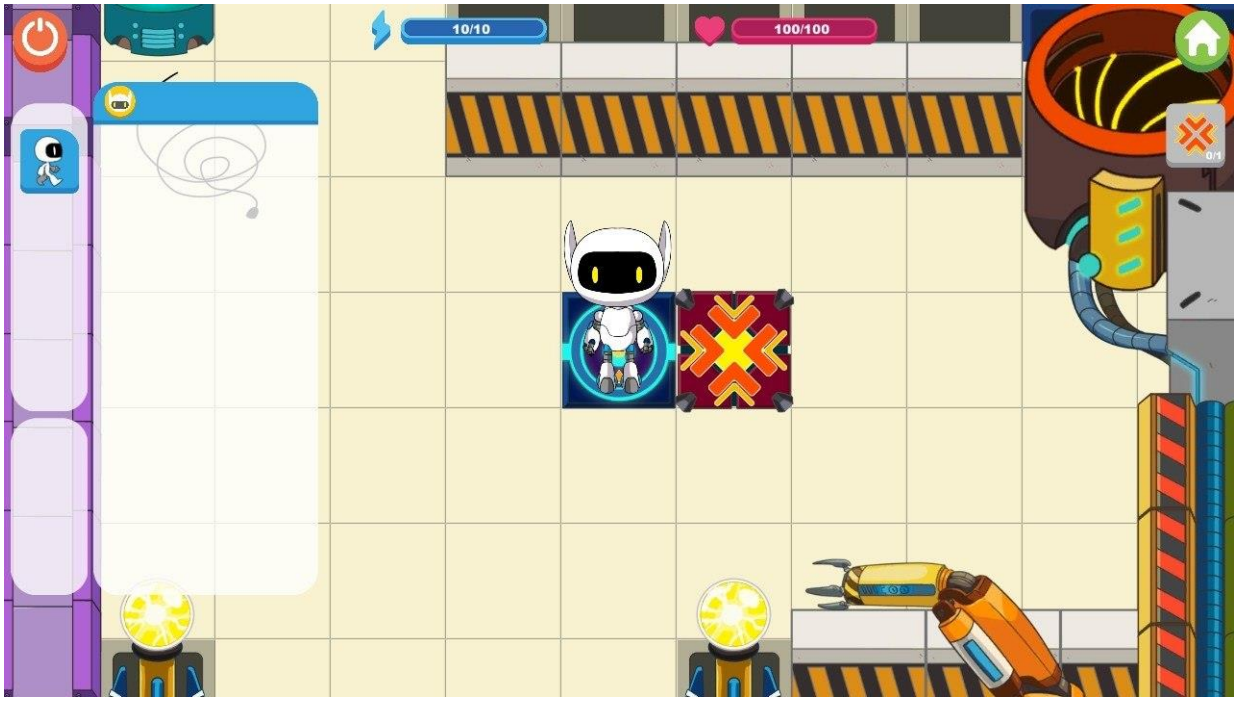


Рисунок 6. Основная сцена игры Robotizen

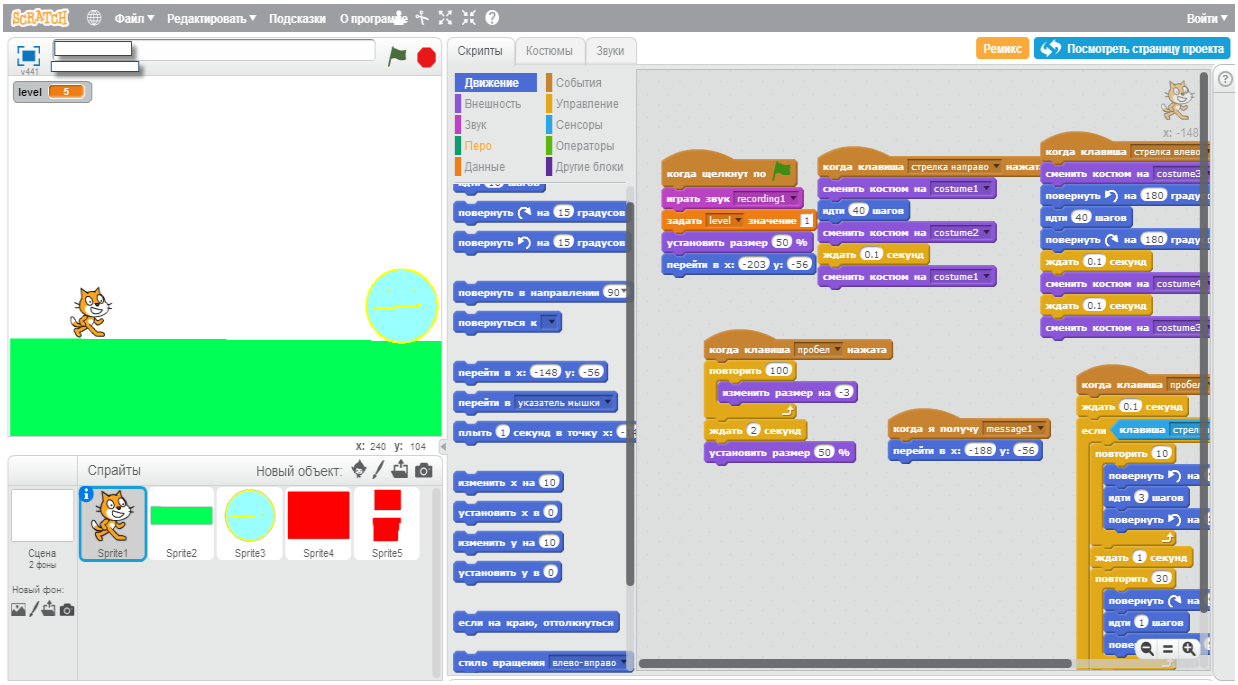


Рисунок 7. Основная сцена игры Scratch

Среди общих недостатков в таблице можно выделить необходимость открывать уровни, что неудобно для использования приложения на уроке. Также во всех этих играх, не реализована одна из основных алгоритмических конструкций – ветвление. Это является большим недостатком, если рассматривать данные приложения, как средство обучения основам алгоритмизации.

Невозможно оставить без внимания, используемую во многих школах для пропедевтики программирования, визуальную среду Scratch (см. рис. 7). Несмотря на все её плюсы, которые подробно описаны в статье [25], она тоже имеет некоторые недостатки. В Scratch отсутствует сюжетная линия и задания, а некоторые команды пишутся словами. Получается, что ребёнку не нужно самостоятельно определять цель написания программы – это придётся делать учителю. А печатание команд словами, может вызвать затруднения при построении алгоритма.

Выводы по главе 1

В результате анализа научно-педагогической и методической литературы, были выявлены психолого-педагогические особенности обучения младших школьников основам алгоритмизации, а также выделены цели и условия обучения детей целевого возраста программированию в рамках занятий по робототехнике.

В процессе изучения условий обучения алгоритмизации, в рамках занятий по робототехнике, был произведен обзор и сравнение электронных средств обучения программированию. В результате, было выяснено, что учителям не хватает качественного методического обеспечения и хороших инструментов для обучения.

Также, были определены требования к содержательной и графической составляющей детского мобильного приложения, на основании которых был произведен анализ существующих обучающих приложений, предназначенных для младших школьников, с целью выявления их положительных и отрицательных сторон. Анализ показал, что ни одно приложение или среда не удовлетворяет предъявленным требованиям. Было принято решение о создании собственного приложения.

Глава 2. Мобильное приложение как средство обучения основам алгоритмизации школьников 1-3 классов

2.1. Цели и задачи создания мобильного приложения "КоробОК"

Анализ сложившейся ситуации, от потребностей до имеющихся средств, приводит к необходимости создания нового приложения, которое может выступать средством начального обучения алгоритмизации.

Цель:

Создать приложение, сочетающее игру, наглядность и, в то же время, предполагающее необходимость построения последовательности действий для игрового персонажа.

Задачи:

1. Написать мини-сюжет для игры;
2. Разработать интерфейс приложения;
3. Смоделировать игровые объекты;
4. Внедрить объекты и детали интерфейса в приложение и запрограммировать их.

Проанализировав особенности разработки мобильных приложений для детей, описанных в параграфе 1.4, на момент начала разработки к данному приложению были составлены конкретные технические требования:

1. Приложение должно быть работоспособным на платформах Android версий 4.2 - 6.0, как наиболее распространенных;
2. Иметь яркий и привлекательный для детей интерфейс с оптимальным размером кнопок, с музыкой и звуками;

3. Содержать сцены: выбора уровня, игровую, с панелью ввода команд (без ограничения количества действий). Написанный алгоритм, а также результаты его выполнения должны сохраняться внутри игры;
4. Иметь тематику игры и мини-сюжет: восстановление экологии планеты после экологической катастрофы, действующий персонаж – робот, с кодовым именем «КоробОК»;
5. Иметь прямоугольное игровое поле размером 6×6 клеток, причем один игровой объект занимает ровно одну клетку;
6. Игровая сцена должна быть выполнена в 2D графике с видом от третьего лица, в изометрической проекции, для наилучшего соотношения реалистичности и производительности игры и простоты разработки;
7. Иметь максимально простую и наглядную систему команд робота (робот может поворачиваться влево и вправо, а также перемещаться вперёд и назад) и содержать все базовые алгоритмические конструкции (линейная, ветвление и циклы);
8. Определять баллы за прохождение уровня, в зависимости от количества команд в программе и использования для прохождения алгоритмической конструкции, если это предполагается в конкретном уровне;
9. Содержать 12 изначально открытых уровней разной сложности, доступных для детей 6-9 лет. По 4 уровня на каждую алгоритмическую структуру (для альфа-версии).

Все эти требования, так или иначе, предполагают, что эта игра будет использована при обучении детей основам алгоритмизации, так как это основная задачи разработки данной игры.

Суть игры заключается в том, что робот должен, по написанному учеником алгоритму движения, доставлять ящики с растениями на плодородную почву. На его пути может оказаться радиация. Для

преодоления этого препятствия необходимо использовать условную алгоритмическую конструкцию. В некоторых уровнях роботу необходимо выполнять повторяющиеся действия, что предполагает использованием циклов, при построении алгоритма.

2.2. Разработка мобильного приложения в межплатформенной среде разработки «Unity» на языке программирования «C#», с использованием программ «Blender» и «Paint.NET»

При создании игры «Коробок» использовались всего три программы, о которых подробнее будет рассказано ниже. Это популярный игровой движок Unity, единственная качественная и одновременно свободная среда для 3D-моделирования Blender и очень удобный растровый графический редактор Paint.NET. Все эти среды можно устанавливать и использовать на платформе Windows, но для более быстрой работы Blender рекомендуется ставить на Linux.

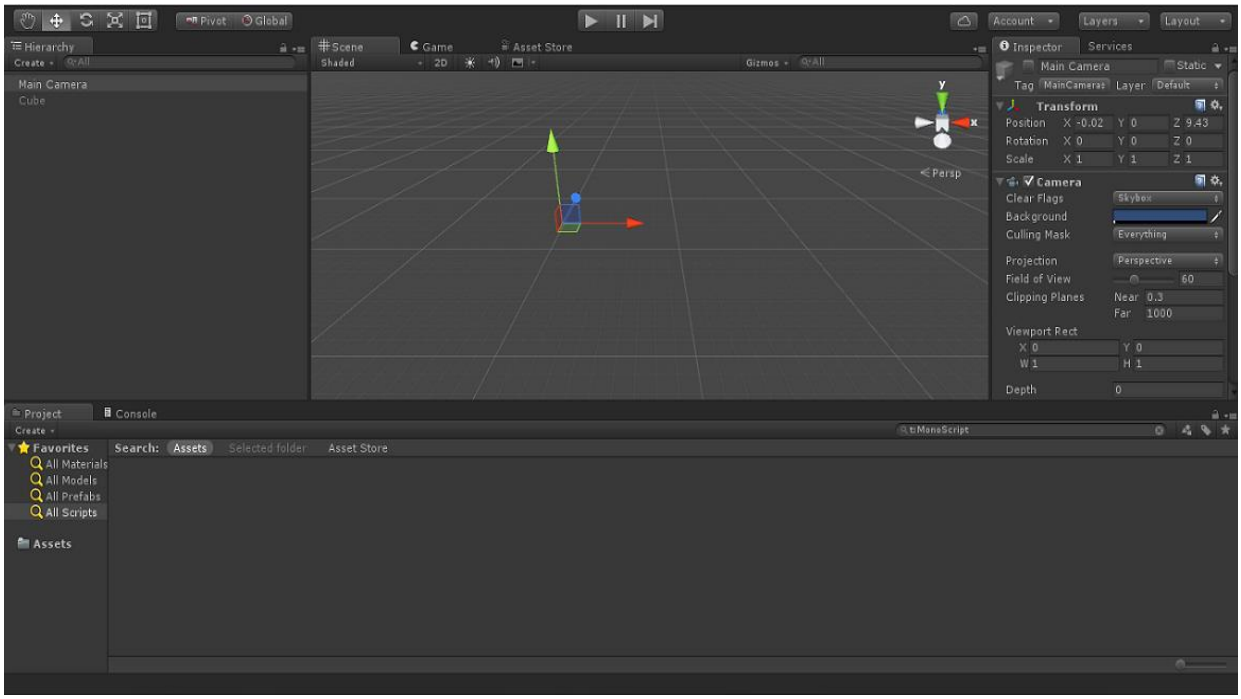


Рисунок 8. Интерфейс Unity

Unity — межплатформенная среда разработки компьютерных игр. Среда поддерживает языки программирования Java и C#. Unity позволяет создавать приложения, работающие под более чем 20 различными операционными системами, включающими персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. Выпуск Unity состоялся в 2005 году и с того времени идёт ее постоянное развитие.

Основными преимуществами Unity являются наличие визуальной среды разработки (см. рис. 8), межплатформенной поддержки и модульной системы компонентов. На Unity написаны тысячи игр, приложений и симуляций, которые охватывают множество платформ и жанров. При этом Unity используется как мелкими разработчиками, так и крупными студиями [1]. Руководство по работе с Unity на русском языке размещено на сайте: https://docs.unity3d.com.

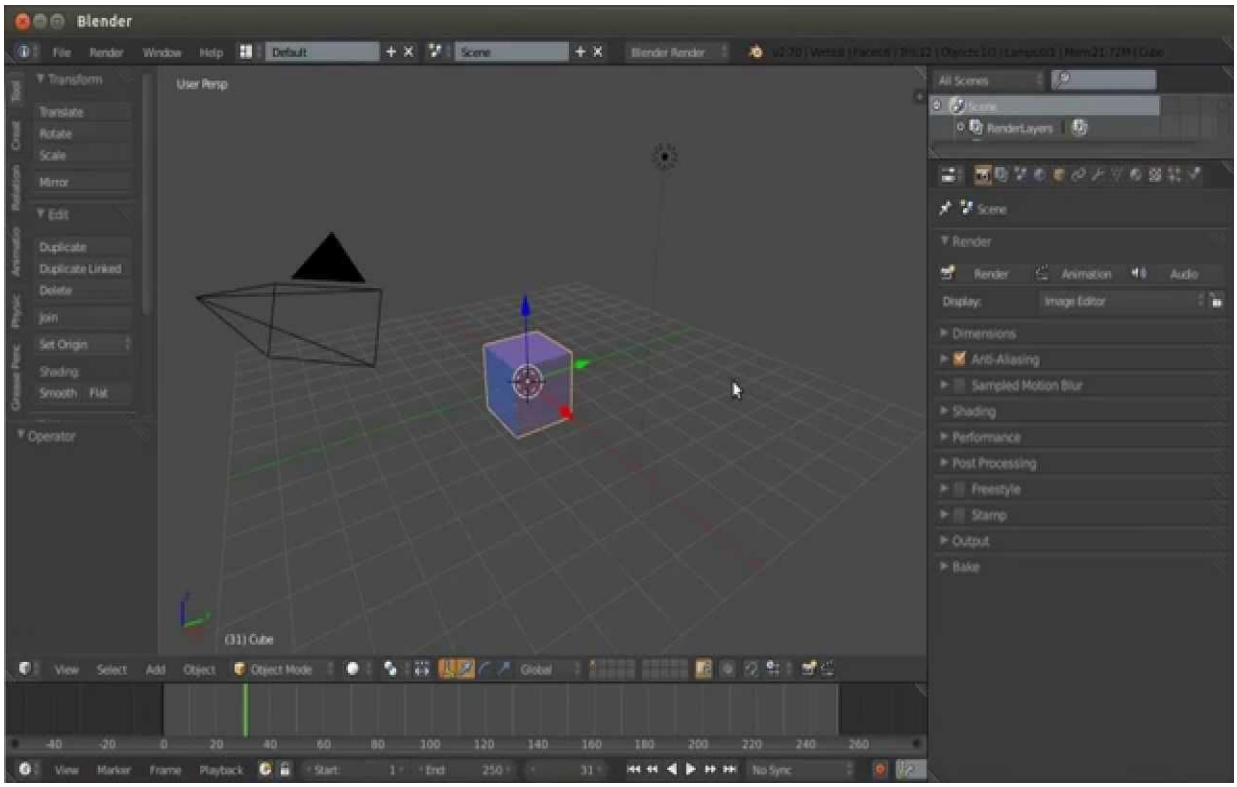


Рисунок 9. Интерфейс Blender

Blender — программное обеспечение для создания трёхмерной компьютерной графики, включающий в себя средства моделирования, анимации, рендеринга, постобработки и монтажа видео со звуком, компоновки с помощью «узлов» (Node Compositing), а также для создания интерактивных игр (см. рис. 9). В настоящее время пользуется наибольшей популярностью среди бесплатных 3D-редакторов в связи с его быстрым и стабильным развитием, которому способствует профессиональная команда разработчиков. Подробную инструкцию по работе с этой замечательной программой можно найти на сайте: https://docs.blender.org/manual/ru/dev/getting_started/help.html.

Paint.NET — это программа для работы с изображениями и фотографиями. Paint.NET поддерживает работу со слоями, неограниченную историю, специальные эффекты и широкое разнообразие полезных и мощных инструментов (см. рис. 10).

Первоначально редактор Paint.NET разрабатывался (под руководством Microsoft) группой студентов Университета штата Вашингтон, некоторые из которых работают над программой и сегодня. Первоначально созданная как бесплатная замена утилиты MS Paint (идущей вместе с системой Windows), программа превратилась в мощный, но все еще простой инструмент для редактирования фотографий и изображений. На этом сайте http://paintnet.ru/help-2/ размещена подробная инструкция по работе с Paint.NET на русском языке.

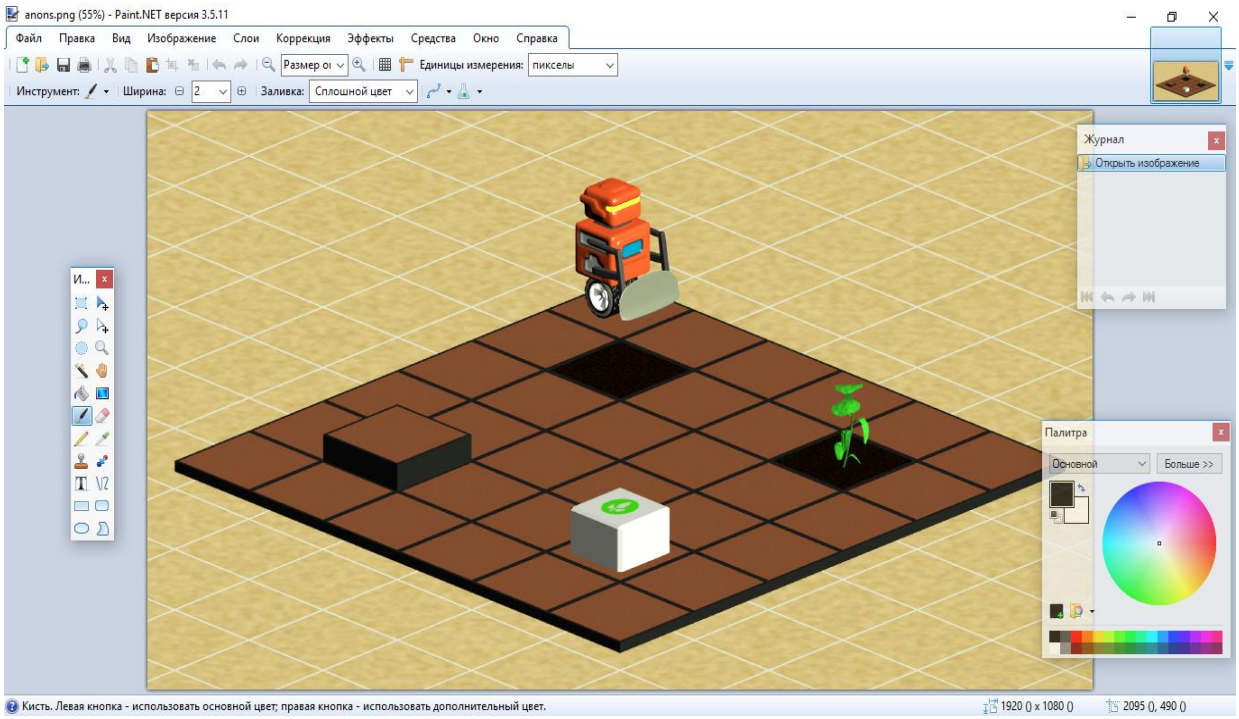


Рисунок 10. Интерфейс Paint.NET

Все эти программы распространяются в свободном доступе и являются «джентельменским набором» для разработки мобильных приложений.

Разработка игры началась с моделирования объектов основной сцены. Для наибольшей производительности и при этом не малой реалистичности – было принято решение сделать сцену в псевдо-3D в изометрической проекции. Большинство объектов этой сцены (кроме кнопок) были смоделированы и отрендерены в трёхмерном редакторе Blender. Все анимации были смоделированы в этой же программе и конвертированы в спрайт-листы для двухмерной анимации.

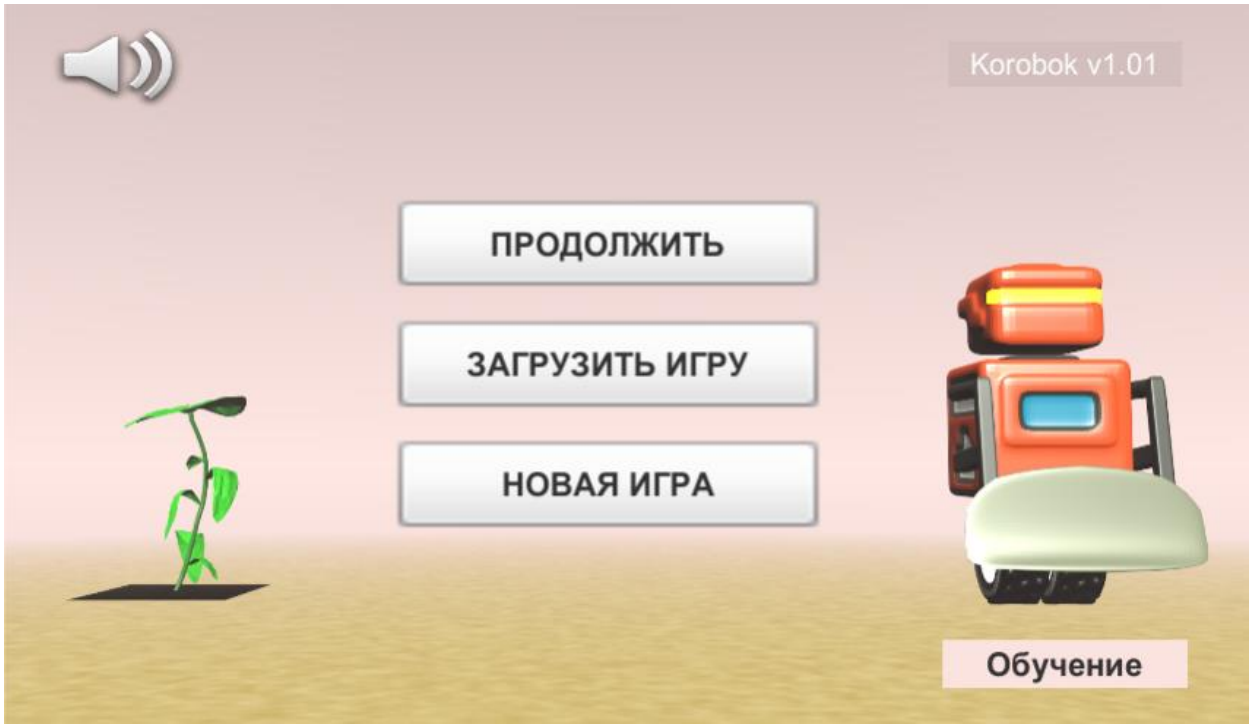


Рисунок 11. Стартовая сцена игры

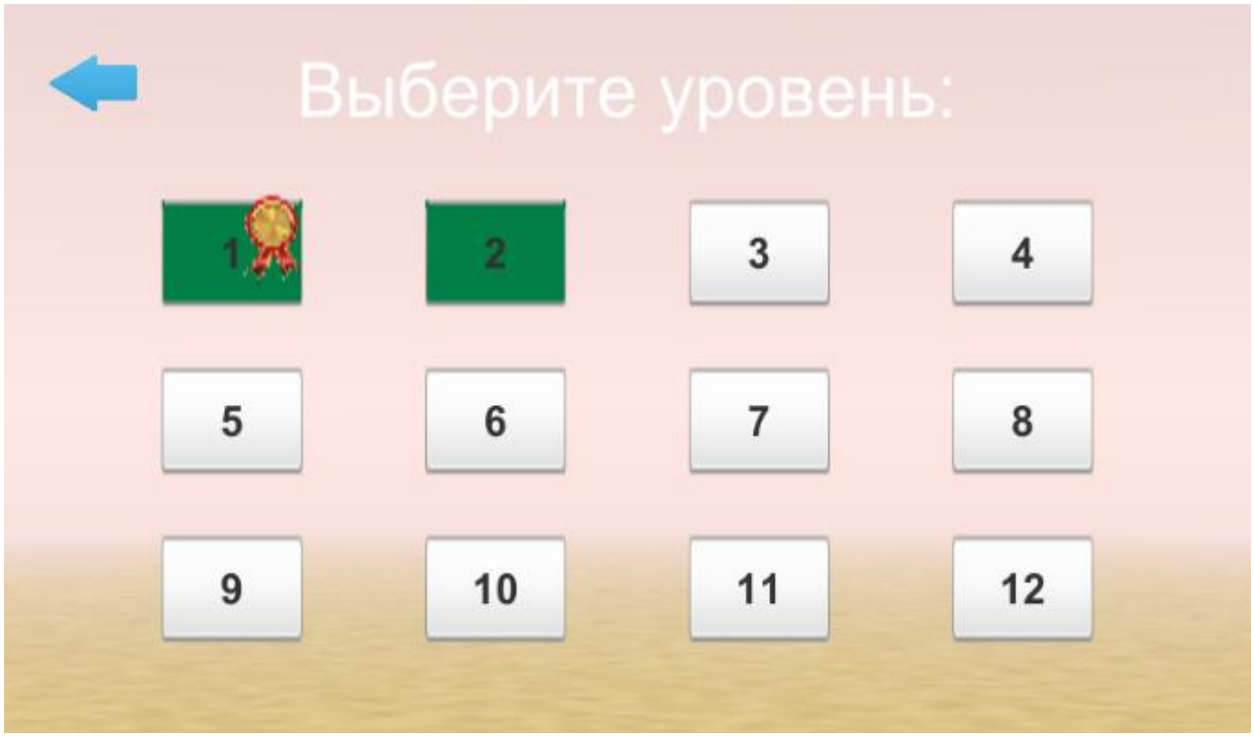


Рисунок 12. Сцена выбора уровня

Следующим шагом был перенос этих анимаций в Unity. Стоит отметить, что можно было моделировать объекты прямо в самом движке. Однако, это плохо бы повлияло на производительность. Тем не менее,

реализация 3D сцен в Unity продумана очень хорошо, поэтому, возможно, программирование поведений и анимаций объектов оказалось бы проще. Чего точно нельзя сказать о программировании 2D объектов.

Рисунок 13. Сцена составления алгоритма для игрового персонажа

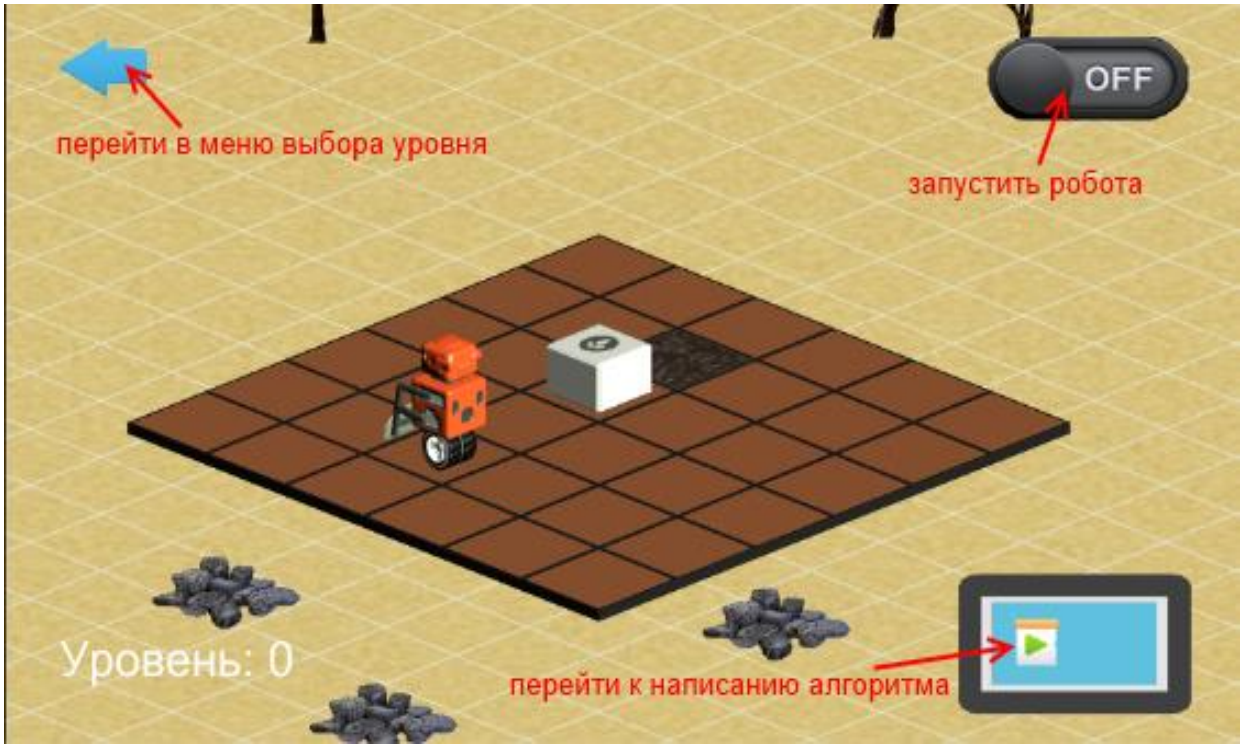


Рисунок 14. Основная сцена игры

Выбор языка программирования зависит только от ваших предпочтений. Руководство по программированию в Unity есть как для Java, так и для C#. Также для обоих языков есть много примеров кода в сети на форумах. Но данное приложение было написано на C#, с использованием сайта с руководством по программированию на C# [22].

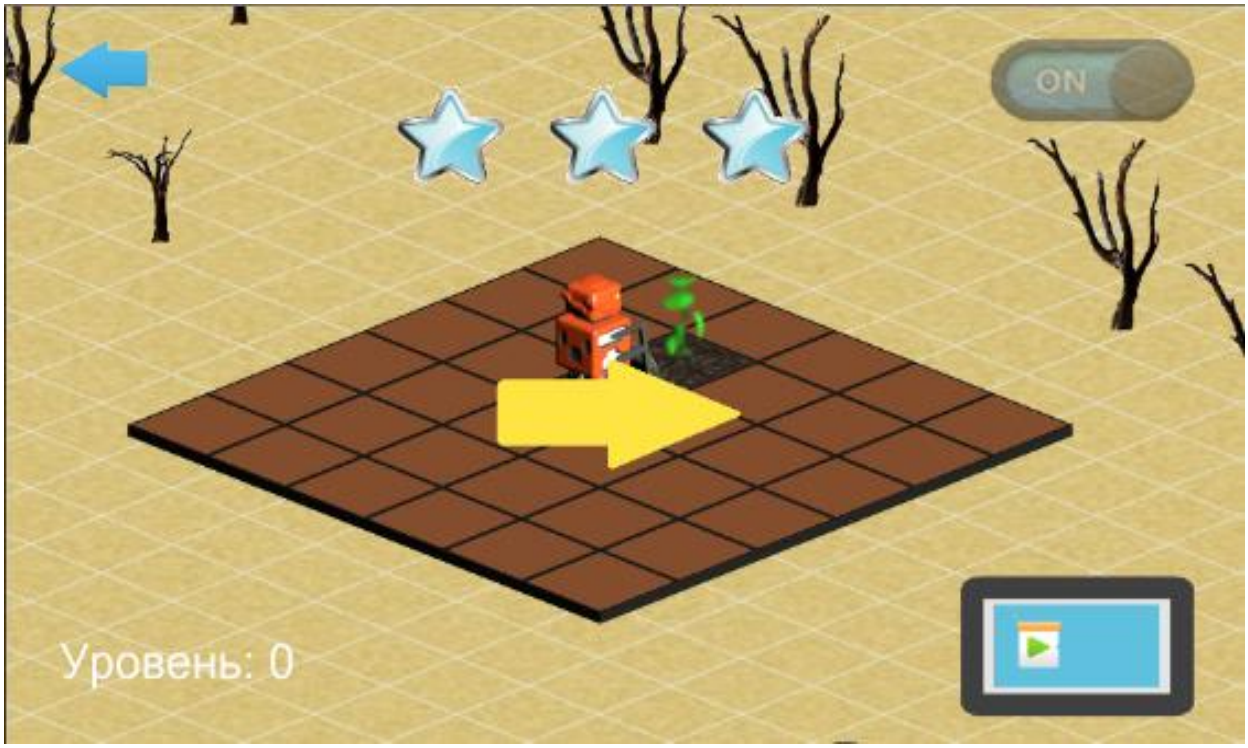


Рисунок 15. Успешное прохождение уровня

В Paint.NET были нарисованы все кнопки и иконки, а также блоки для сцены, где составляется алгоритм для игрового персонажа. Это один из самых удобных растровых редакторов, и он очень прост в использовании.

На данный момент разработана альфа-версия игры, которую можно использовать для проведения различных исследований, в том числе и на предмет понимания детьми основ алгоритмизации. «КоробОК» соответствует всем требованиям, описанным в пункте 1.3. Игра включает в себя:

1. Несколько открытых изначально игровых миссий и мини-сюжет (по сюжету на Земле, из-за загрязнения окружающей среды, вымерли все растения и роботу необходимо доставлять ящики с растениями на плодородную почву);

2. Стартовое меню, где можно загрузить старую или начать новую игру, а также отключить звук или запустить обучение (см. рис 11);
3. Меню выбора уровня, в котором пройденные уровни помечаются зеленым цветом, а уровни, пройденные с эффективно составленным алгоритмом, помечаются медалькой (см. рис 12);
4. Сцену с игровым полем, по которому робот перемещается, выполняя заданные игроком команды (см. рис 13);
5. Простую и наглядную систему команд игрового персонажа и все базовые алгоритмические конструкции (см. рис. 14);
6. Систему оценивания эффективности написанной игроком программы, после прохождения уровня даются звёздочки (см. рис. 15).

Перейдя по этой ссылке, можно посмотреть более подробный обзор этого приложения, записанный в формате видеоролика: https://drive.google.com/file/d/15jXj241JvUD8SWeBQFMJJoisYs2XzY4A/view?usp=sharing.

На пути создания приложения с изометрическим видом можно встретить много трудных задач, решение которых сложно найти в сети. Полезным ресурсом в этой ситуации окажется самый крупный видеохостинг в мире YouTube. Ниже приведена ссылка на пример реализации ромбической изометрии, правда, на не поддерживаемом Unity языке программирования Pascal:
https://www.youtube.com/watch?v=nls0dyTeEns&list=PLHcq_lDrZqm0pcMN36rKfFUnxQvasRGRP.

На языке C# получился вот такой код:

```
using UnityEngine;
// для удобства создаётся структура содержащая две целые координаты
public struct PositionXY
{
    public int X;
    public int Y;
```

```
    public PositionXY(int X, int Y)
    {
        this.X = X;
        this.Y = Y;
    }
}
// и структура содержащая две вещественные координаты
public struct floatXY
{
    public float X;
    public float Y;

    public floatXY(float X, float Y)
    {
        this.X = X;
        this.Y = Y;
    }
}
public class Position {
    private static PositionXY valueInt;
    private static floatXY valueFloat;
        // ...
// смещение реальных координат при перемещении объекта на одну
клетку    public static float dx = 1.132f, dy = 0.42f;
// определение координат на поле через координаты объекта на экране
    public static floatXY XYfloat(float Cx0, float Cy0, int X, int Y)
    {
        valueFloat.X = Cx0 + dx * (X + Y);
        valueFloat.Y = Cy0 + dy * (X - Y);
        return valueFloat;
    }
    // определение координат на экране через координаты объекта на
поле
    public static PositionXY intXY(floatXY Pos0, float fX, float fY){
        int a = 0, b = 0;
        a = Mathf.RoundToInt ((fX - Pos0.X) / dx);
        b = Mathf.RoundToInt ((fY - Pos0.Y) / dy);
        valueInt.X = ((a + b) / 2);
        valueInt.Y = ((a - b) / 2);
        return valueInt;
    }
}
```

Ещё одна проблема, связанная с реализацией изометрии – это порядок отображения объектов на поле. Должно казаться, что объекты, находящиеся на поле ниже, ближе к игроку, то есть отображаться поверх тех объектов, которые на игровом поле стоят выше. Решение довольно простое. Отключить

режим 3D редактирования полностью нельзя. С одной стороны это минус, но он пригодится в данном решении:

```
// создаётся отдельный скрипт, который затем привязывается
// ко всем игровым объектам на поле
using UnityEngine;
public class Zindex : MonoBehaviour
{
    void Update()
    {
// координата z объекта должна зависеть от y, а можно просто их
прировнять
// и Unity автоматически будет рисовать объект, который ближе по z,
поверх
// других объектов
transform.position = new Vector3(transform.position.x, transform.position.y,
transform.position.y);
    }
}
```

Очень частая проблема в Unity: игровые префабы (заготовка для клонов схожих объектов) нельзя подключить в скрипт как статичный объект. В результате этого приходится преодолевать ряд трудностей, чтобы передать какие-либо параметры в другой скрипт. Также приходиться переопределять нестатичные методы класса, при использовании в нём параметров префаба.

Первый способ решения данной проблемы связан с созданием дополнительной публичной и статичной переменной логического типа, и делать ее истинной или ложной в зависимости от текущего состояния конкретного параметра объекта. Так, например, переменная plant принимает параметр true, когда координаты всех ящиков на поле, совпадают с координатами почвы:

```
// инициализация логической переменной в классе скрипта-контроллера
ящика
public class CubeController : MonoBehaviour
{
    public static bool Plant = false;
    // …
    // здесь определяются координаты ящиков и почвы
    IEnumerator OffsetCoroutine()
    {
        // ...
         // если координаты почвы совпадают с координатами ящика
```

```
if (MapScript.SoilArr.Contains(Pos) && !flagSoil)
        {
          flagSoil = true;
          MapScript.counter++;
        }
        // если ящик стоял на почве, но его сдвинули
if (!(MapScript.SoilArr.Contains(Pos)) && flagSoil)
        {
          flagSoil = false;
          MapScript.counter--;
        }
        // ...
    }
}


// класс определяющий начальные координаты объектов на игровом
поле
public class MapScript : MonoBehaviour
{
    // переменные для определения, что все ящики на местах
    public static int counter;
    private int counterCube = 0;
    void Update ()
    {
        // когда количество ящиков, доставленных на почву
        // равно количеству ящиков на поле, переменная принимает
        значение
        if (counter == counterCube && counterCube != 0)
        {
        CubeController.Plant = true;
        }
    }
    // …
}
```

Есть второй способ решения этой проблемы – создать экземпляр класса статичного объекта внутри скрипта. Так, например:

```
// в скрипте блочного программирования, при удалении блока
// необходимо смещать камеру влево
public class MovePanel : MonoBehaviour
{
    //...
    void Destroyed(int px, int py)
    {
    // ...
    delete:
        ControlCamera.Back(coordinate.X);
```

```
    }
    // ...
}

// для этого в скрипте, управляющем камерой
public class ControlCamera : MonoBehaviour
{
    public static GameObject empt;
    public static void Next(float x)
    {
        empt = GameObject.Find("Main Camera");
        // вызываем не статичный метод из статичного
        ControlCamera nonstat = empt.AddComponent<ControlCamera>();
        if ((x - nonstat.transform.position.x) > 3f)
        {
            while (!((x - nonstat.transform.position.x) <= 3))
            {
nonstat.transform.position = new Vector3(nonstat.transform.position.x + 1f,
nonstat.transform.position.y, nonstat.transform.position.z);
            }
        }
    }
    // ...
}
```

Следующий фрагмент кода может пригодиться многим разработчикам игр. Поскольку в Unity при переходе с одной игровой сцены на другую, уничтожаются все объекты данной сцены, то без этого кода невозможно сделать непрерывное воспроизведение музыки в игре:

```
// создаётся скрипт, который потом привязывается
// к объекту, проигрывающему музыку
using System.Collections;
using UnityEngine;
public class DontDestroy : MonoBehaviour
{
    private AudioSource audioManager;
    public static bool flagPause = false;
    // ...
    public void Awake()
    {
        audioManager = GetComponent < AudioSource > ();
        // обозначаем в коде, что этот объект удалять нельзя
        DontDestroyOnLoad(this);
        // но при переходе на новую сцену иногда наблюдается
дублирование
        // аудио-объекта, поэтому уничтожаем лишний
        if (FindObjectsOfType(GetType()).Length > 1)
```

```
        {
            Destroy(gameObject);
        }
// уничтожается последний созданный, поэтому музыка не начнется заново
    }
}
```

В результате, была создана альфа-версия мобильного приложения, названного «КоробОК».

2.3. Методические рекомендации по использованию игры «КоробОК» при обучении основам алгоритмизации для учителей информатики и преподавателей робототехники

Данную игру рекомендуется использовать перед началом изучения программирования на занятиях по робототехнике и на уроках по информатике. Также игру «Коробок» можно использовать детям для развлечения и родителям для организации полезного досуга детей. Игра предназначена для детей младшего школьного возраста, однако, с не меньшим интересом и пользой в нее играют дети и более старшего возраста.

Приложение включает в себя 12 уровней на три алгоритмических конструкции. Первые 4 уровня – это первый этап, где для успешного прохождения используется только линейная конструкция. Уровни с 5 по 8 включительно – это второй этап, где необходимо использовать ветвление. Последний третий этап (уровни с 9 по 12) предполагает обязательное использование циклов. Для удобства использования игры на занятиях – все уровни изначально открыты. Уровни, предусматривающие использование ветвления и циклы можно пройти и без использования этих конструкций. Но для получения максимального количества баллов – их использование обязательно.

Во избежание путаницы в сцене построения алгоритма запрещено использовать ветвление внутри ветвления и цикл в цикле. Но можно использовать цикл в ветвлении и наоборот. Чтобы не ограничивать свободу

мышления ребёнка, изначально открыты все команды робота и все конструкции, ведь каждый уровень можно пройти несколькими разными алгоритмами.

Уровни второго и третьего этапа можно сначала пройти ученику без использования ветвлений и циклов, а затем показать целесообразное их использование. Ведь, если проходить уровни первого этапа без ветвления, то алгоритм не всегда будет рабочим, так как радиация появляется в случайной клетке. А если проходить уровни второго этапа без циклов, то придётся писать большое количество повторяющихся команд.

Если ребёнок уже знаком с основными алгоритмическими конструкциями, то можно эту игру дать ему на самостоятельное прохождение. Возможна организация прохождения этапов в группах по 2-3 человека.

Рекомендуется разделить прохождение игры по алгоритмическим конструкциям на три занятия. Порядок прохождения уровней внутри каждого этапа не имеет значения, однако, порядок прохождения этапов менять не следует.

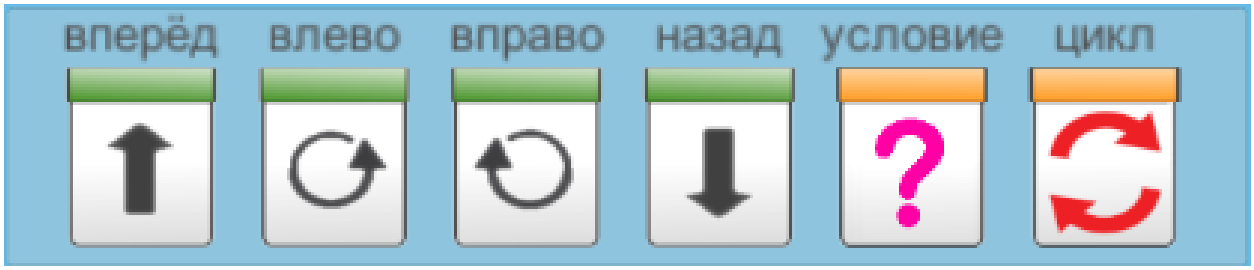


Рисунок 16. Панель команд Коробка

Первое занятие: Линейные алгоритмы

Преобладающим методом обучения на занятии будет являться игра, поскольку использование игрового приложения предполагает использование данного метода. Но в начале урока потребуется использовать объяснение. При объяснении – учитель излагает строго и последовательно основные и необходимые для данного этапа правила игры, а также знакомит ученика с её функционал и интерфейсом.

В конце занятия, если осталось время, в качестве контроля полученных знаний, можно предложить детям решение задач из теста (см. приложение А). Тест составлен с использованием заданий на понимание алгоритмов из рабочей тетради Крыловой О. Н. [13].

На первом занятии рекомендуется просмотр обучающего ролика, который предлагается в начале игры. Также проводиться ознакомление ученика с основными командами игрового персонажа.

Правила, которые должен знать ученик перед началом игры

Игровой персонаж – робот с кодовым именем «КоробОК» должен доставлять ящики с растениями на плодородную почву. Он может перемещаться по игровому полю, которое имеет размер 6×6 клеток, вперёд и назад, а также поворачиваться вправо и влево. На его пути может встретиться препятствия в виде каменных завалов. По таким клеткам робот двигаться не может. Ящик с растением он может толкать только ковшом, то есть только когда движется вперёд. Он не может толкать два ящика сразу или толкать его при движении назад.

Для того чтобы робот выполнил свою миссию, необходимо в отдельной игровой сцене составить для него линейный алгоритм. Алгоритмом в данном случае будет набор действий робота, выбранных ребёнком. Линейный алгоритм – это когда каждое действие выполняется по очереди друг за другом. Каждой команде робота соответствует кнопка на панели команд, которую необходимо нажимать, чтобы записать команду в алгоритм (см. рис. 16). Команды можно удалять отдельно или удалить все сразу с помощью кнопок под мини-картой.

Краткое содержание этапа

Для удобства в этой сцене отображается мини-карта. На рисунке 17 представлены мини-карты для первых четырёх игровых уровней. Ученик составляет алгоритм и пробует запустить его.

Рисунок 17. Мини-карты уровней первого этапа

После каждой попытки прохождения уровня рекомендуется проводить рефлексию и анализ алгоритма, почему, например, данный алгоритм оказался неудачным, или почему он не был оценен на три звезды. Если у ребёнка совсем не получается пройти ни один из уровней, то можно попробовать показать ему правильный алгоритм прохождения одного из уровней этого этапа. Либо демонстрировать выполнение персонажем различных алгоритмов, не обязательно выполняющих основную задачу (доставить ящик на плодородную почву).

Примерные вопросы для учеников после прохождения этапа

1. Что такое алгоритм?
2. Какой алгоритм называется линейным?
3. Решение задач 1, 2, 3 и 6 из теста в приложении А.

Второе занятие: Ветвление

На втором занятие будут использоваться те же методы обучения, что и на первом: объяснение, игра, решение задач.

Правила, которые должен знать ученик перед началом игры

Во-первых, необходимо повторить правила из первого занятия. Во-вторых, необходимо объяснить правила связанные с неизвестными клетками поля и радиацией. Ровно половина клеток, отмеченных вопросительным

знаком, заражена радиацией. Робот сканирует клетку, которая находится впереди него. Какая из клеток заражена не известно заранее (так как это определяется случайным образом в момент запуска алгоритма). Через радиацию робот не может ехать и толкать ящики, потому что она губительна и для него и для растения. Во избежание уничтожения растения, роботу запрещено толкать ящик через неизвестные клетки, сначала эти клетки необходимо просканировать.

Чтобы добавить в алгоритм ветвление (можно называть его просто «ветка»), нужно нажать на кнопку с вопросительным знаком на панели команд персонажа (см. рис. 16). Вставлять «ветку» нужно в том месте алгоритма, где робот будет стоять перед клеткой с вопросительным знаком. Теперь алгоритм разветвился. Внизу пишутся команды для случая, если робот не обнаружил на впереди идущей клетке радиации. Сверху добавляются действия, которые робот должен выполнить, если радиация все-таки есть.

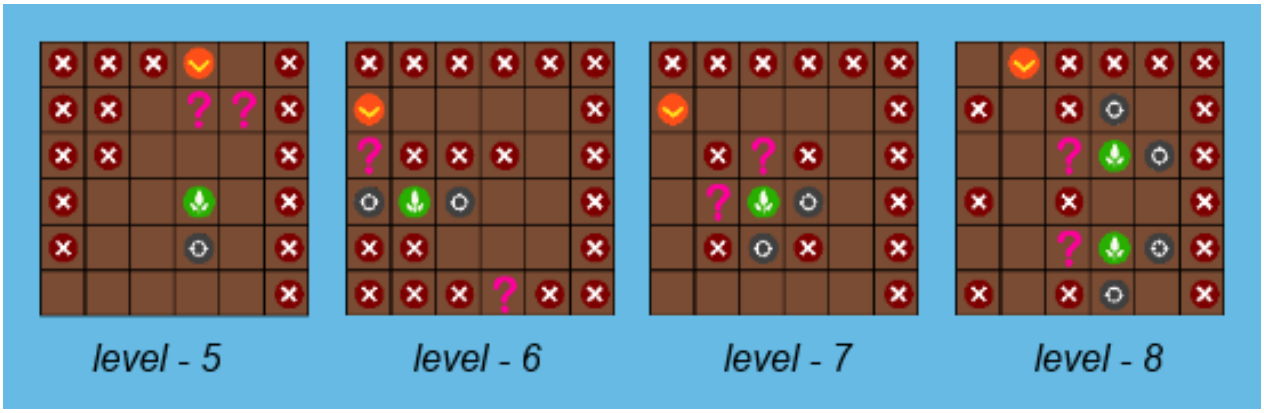


Рисунок 18. Мини-карты уровней второго этапа

Краткое содержание этапа

Задания для робота на втором этапе изображены на рисунке 18. После повторения правил, рекомендуется объяснить ребёнку, что использование Ветвления сделает возможным прохождение уровня с одним и тем же алгоритмом, вне зависимости от того, в каком месте окажется радиация.

Именно для этого используется ветвление. Для лучшего понимания, можно многократно запустить, написанный учителем алгоритм.

Примерные вопросы для учеников после прохождения этапа

1. Для чего нужно ветвление?
2. Можно ли обойтись без него?
3. Решение задач 4 и 5 из теста (см. приложение А).

Третье занятие: Циклы

На третьем занятии необходимо сформировать представление о циклической алгоритмической конструкции. Методы обучения: объяснение, игра, решение задач.

Правила, которые должен знать ученик перед началом игры

Циклы нужны затем, чтобы не записывать повторяющиеся действия в алгоритм, а просто указать сколько раз нужно их повторить. Например, на рисунке 21 среди заданий для третьего этапа в 11 уровне робот, для успешного прохождения должен повернуться влево, проехать вперед, повернуться вправо, проехать вперёд и сделать это он должен ровно 4 раза. То есть алгоритм станет в 4 раза меньше по объему, благодаря циклу. Можно даже наглядно продемонстрировать оба алгоритма и, что они оба рабочие. Однако алгоритм без цикла не получит 3 звезды.

Чтобы добавить в алгоритм цикл, нужно нажать на кнопку с двумя круговыми стрелочками на панели команд персонажа (см. рис. 16). Необходимо сначала вставить цикл, а уже внутри него вставлять повторяющиеся команды. После составления набора команд, необходимо указать сколько раз их нужно повторить. Для этого нужно нажать на левый блок цикла с цифрой (она и означает, сколько раз будут повторяться команды) и выбрать другую цифру от 1 до 8. Теперь алгоритм зациклился.

Краткое содержание этапа

На рисунке 19 изображены задания для третьего этапа. После прохождения учеником этапа, обязательно обсудить, можно ли использовать эту алгоритмическую структуру в предыдущих уровнях. Если ребёнок справился с заданиями быстро, то можно попробовать пройти с ним какой-нибудь из предыдущих уровней с наименьшим количеством команд.

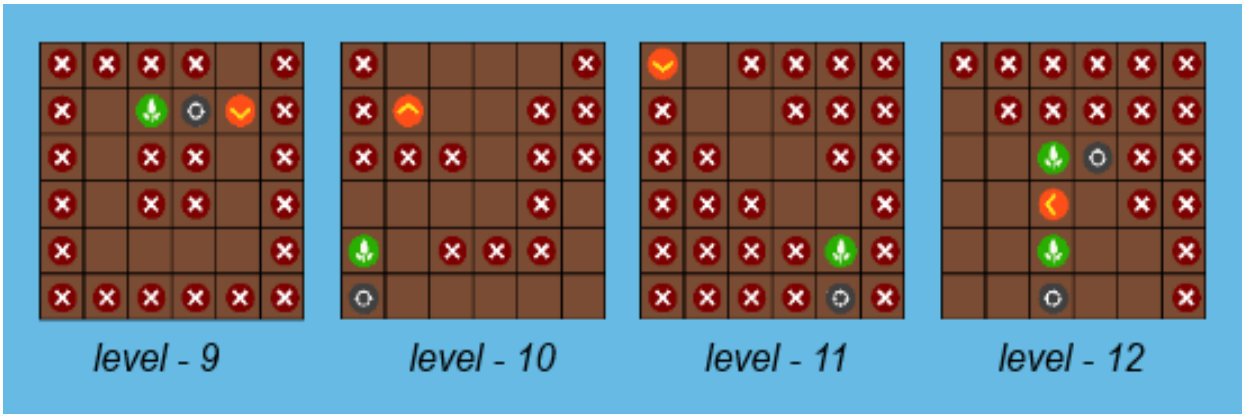


Рисунок 19. Мини-карты уровней третьего этапа

Примерные вопросы для учеников после прохождения этапа

1. Зачем нужны циклы?
2. Можно ли обойтись без них?
3. Решение задач 7 и 8 из теста (см. приложение А).

Хорошим критерием понимания алгоритмических конструкций является прохождение всех уровней на три балла. Успешное построение учеником алгоритмов для реальных моделей в будущем также является показателем его понимания основ алгоритмизации. Помимо этого, после прохождения игры, можно тестировать детей на понимание алгоритмов.

Тест, представленный в приложении 1, можно использовать не в конце каждого занятия, а после прохождения всех этапов, для получения более конкретных результатов обучения. В тесте 8 вопросов на построение или восполнение алгоритмов. 1, 2, 3 и 6 задание на понимание линейной алгоритмической структуры. 4 и 5 задание на понимание условной

структуры. 7 и 8 на понимание циклов. Тест не предполагает выбора ответа. Ученики должны полностью записать ответ сами. Если результаты теста плохие, стоит ещё раз проработать с учеником конструкцию, которая вызвала у него затруднения.

В клубе робототехники «SmartLab» была проведена первичная апробация альфа-версии приложения. Преподаватели этого клуба составили на него рецензию (см. приложение Б). В ней описаны основные недостатки и достоинства приложения, советы по доработке, а также сказано о том, что приложение вполне можно использовать для обучения детей младшего школьного возраста основам алгоритмизации.

Выводы по главе 2

В процессе анализа инструментов для создания мобильных игр, был выделен минимальный набор программ для разработки, которые распространяются в свободном доступе – Unity, Blender и Paint.NET. Помимо этого был выполнен их краткий обзор, выделены основные функции и приведены примеры, для чего они могут использоваться.

В ходе создания обучающего мобильного приложения были выполнены все технические требования к нему, описанные в параграфе 2.1, а также были разработаны методические рекомендации для учителей информатики и преподавателей робототехники по использованию этого приложения на занятиях. Методические рекомендации включают в себя тест для проверки понимания детьми алгоритмизации.

В результате, была разработана игра «КоробОК» и проведена её апробация в клубе робототехники SmartLab. Преподаватели этого клуба оставили официальный отзыв о данном приложении и о возможности его использования при обучении детей основам алгоритмизации.

Заключение

В рамках данной дипломной работы были рассмотрены цели и условия обучения программированию на занятиях по робототехнике. Описаны психолого-педагогические особенности обучения детей младшего школьного возраста программированию и алгоритмизации. Были составлены некоторые критерии для инструмента, который может обеспечить пропедевтику программирования. Также был проведен анализ уже существующих приложений и программ, которые могут выступать в качестве такого инструмента. Затем были рассмотрены и рекомендованы к использованию при разработке мобильных приложений под Android некоторые программы, распространяемые свободно. Подробно описан процесс создания игры «КоробОК» и добавлены некоторые рекомендации для разработчиков.

Цель работы была выполнена – было разработано мобильное приложение, которое можно использовать для обучения детей младшего школьного возраста основам построения алгоритмов. Помимо этого были составлены методические рекомендации для учителей и преподавателей робототехники. А также была проведена апробация приложения, которая показала, что использование данного приложения для обучения детей основам алгоритмизации вполне возможно и даже даёт положительные результаты.

Все поставленные задачи были полностью реализованы, но апробация приложения позволила выявить ряд недостатков. В результате был составлен план по дальнейшему развитию игры:

1. *Усовершенствование системы обучения*. Сейчас обучение в игре представляется в виде длинного видеоролика. Планируется создание трёх обучающих уровней и автоматический запуск их перед началом каждого этапа;
2. *Отказ от текста*. Игровой сюжет сейчас описан словами. Планируется их замена на анимацию и картинки. В результате этого, детям младшего

школьного возраста будет проще воспринимать игру. И, возможно, это позволить использовать игру даже на занятиях с дошкольниками;

3. *Усовершенствование сцены программирования*. Во время апробации приложения были выявлены многие недостатки этой сцены. Это: обход правил программы; слабая интерактивность сцены; отсутствие цельности блоков ветвления и циклов. Планируется доработка этих недостатков;
4. *Усовершенствование интерфейса*. Некоторые элементы интерфейса были выполнены «на скорую руку». Планируется их доработка, а также добавление в игру большего количества анимаций. Это позволит улучшить визуальное восприятие программы детьми.

После доработки приложения до бета-версии предполагается его размещение в самом популярном магазине приложений Google Play.

В целом, можно сказать, что детей необходимо обучать программированию уже в начальной школе. Это будет способствовать развитию памяти, мышления и самоконтроля ученика, и формированию стиля мышления соответствующего требованиям современного общества. А предлагаемое приложение может оказаться хорошим инструментом подготовки к изучению языков программирования и освоения основ построения алгоритмов.

Список использованных источников

1. Анимация [Электронный ресурс]: Unity - Руководство: Руководство Unity, 2015. – Режим доступа: https://docs.unity3d.com, свободный, дата обращения: 05.04.2018;
2. Волков Б.С. Психология младшего школьника: Учебное пособие. - Москва: Педагогическое общество России, 2002;
3. Гамезо М.В. Возрастная и педагогическая психология: Учеб. пособие для студентов всех специальностей педагогических вузов / М.В. Гамезо, Е.А. Петрова, Л.М. Орлова. - М.: Педагогическое общество России, 2003. — 512 с.;
4. Государственная программа Российской Федерации «Развитие образования» на 2013-2020 годы;
5. Давайте учить детей программированию [Электронный ресурс]: Сайт «Дев Бай». 2008–2018 ООО «Дев Бай Медиа». – Режим доступа: https://dev.by/lenta/main/davayte-uchit-detey-programmirovaniyu, свободный, дата обращения: 17.04.2018;
6. Давыдов В. В., Слободчиков В. И., Цукерман Г. А. Младший школьник как субъект учебной деятельности // Вопросы психологии. 1992. № 3-4. С.14-19;
7. Дизайн мобильного приложения. Как добиться оптимального результата? [Электронный ресурс]: ресурс для IT-специалистов. – Электрон. журн. – «ТМ», 2006 - 2018. – Режим доступа: https://habr.com/company/littlebeetle/blog/171533/, свободный, дата обращения: 19.06.2018;
8. Зимняя И.А. Педагогическая психология. – Ростов-на-Дону: Феникс,1997.-480 с.;

9. Как выполнять сохранение и загрузку игры в Unity [Электронный ресурс]: ресурс для IT-специалистов. – Электрон. журн., сайт «TProger», 2015. – Режим доступа: https://tproger.ru, свободный, дата обращения: 05.04.2018;
10. Карты тайлов [Электронный ресурс]: Электронная документация, 2013 - 2018. – Режим доступа: http://www.libgdx.ru, свободный, дата обращения: 05.04.2018;
11. Козлова С.А. Развитие логического и алгоритмического мышления у дошкольников и младших школьников / С. А. Козлова // Начальная школа плюс До и После. 2006. №9. с. 23-28;
12. Козлова, Л. Л. Урок информатики в начальной школе по теме: «Алгоритм» [Электронный ресурс]: Фестиваль педагогических идей «Открытый урок». – Электрон. журн. – ИД «Первое сентября», 2003 - 2018. – Режим доступа: http://открытыйурок.рф, свободный, дата обращения: 05.04.2018;
13. Крылова О.Н. Тесты по информатике: 2 класс к учебнику А.В. Горячева, К.И. Гориной, Т.О. Волковой «Информатика в играх и задачах. 2 класс: учебник в 2-ч частях» / О.Н. Крылова. – М.: Издательство «Экзамен», 2011. – 95 с.;
14. Кулагина И.Ю. Возрастная психология. Развитие ребенка от рождения до 17 лет: Учебное пособие. 2-е изд.-М.: Изд-во УРАО, 1997. – 176 с.;
15. Основы создания 2D персонажа в Unity 3D 4.3. Часть 2: бегущий персонаж [Электронный ресурс]: ресурс для IT-специалистов. – Электрон. журн. – «ТМ», 2006 - 2018. – Режим доступа: https://habrahabr.ru, свободный, дата обращения: 05.04.2018;
16. Первин Ю. А. Проблемы раннего обучения информатике в российской школе. // Вопросы образования. – 2005. – №3;
17. Приложения для всей семьи и COPPA [Электронный ресурс]: Центр правил для разработчиков Google Play. – Режим доступа:

https://play.google.com/intl/ru_ALL/about/families/designed-for-families/authentication/, свободный, дата обращения: 19.06.2018;

18. Пример создания простой 2D игры для Android с использованием игрового движка Unity [Электронный ресурс]: ресурс для IT-специалистов. – Электрон. журн. – «TM», 2006 - 2018. – Режим доступа: https://habrahabr.ru, свободный, дата обращения: 05.04.2018;

19. Примерная основная образовательная программа начального общего образования. Одобрена решением федерального учебно методического объединения по общему образованию (протокол от 8 апреля 2015 г. № 1/15);

20. Работа с Корутинами в Unity [Электронный ресурс]: ресурс для IT-специалистов. – Электрон. журн. – «TM», 2006 - 2018. – Режим доступа: https://habrahabr.ru, свободный, дата обращения: 05.04.2018;

21. Распоряжение Правительства РФ от 1 ноября 2013 г. № 2026-р «Стратегия развития отрасли информационных технологий в Российской Федерации на 2014-2020 годы и на перспективу до 2025 года»;

22. Руководство по программированию на C# [Электроныый ресурс]: Электронная документация. 2018 Microsoft. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/67ef8sbd(v=vs.120).aspx, свободный, дата обращения: 17.04.2018;

23. Рыжиков С. Больше кода: что государство может сделать с четырехкратной нехваткой программистов в России? // Интернет-журнал «Forbes». – 2017. – Режим доступа: http://www.forbes.ru/tehnologii/341439-bolshe-koda-chto-gosudarstvo-mozhet-sdelat-s-chetyrehkratnoy-nehvatkoy, свободный, дата обращения: 14.04.2018;

24. Скриптинг [Электронный ресурс]: Unity - Руководство: Руководство Unity, 2015. – Режим доступа: https://docs.unity3d.com, свободный, дата обращения: 05.04.2018;

25. Сорокина Т. Е. Методика раннего общедоступного программирования в основной образовательной программе. // Современные информационные технологии и ИТ-образование. – 2017. [Электронный ресурс]. – Режим доступа: http://sitito.cs.msu.ru/index.php/SITITO/article/view/109, свободный, дата обращения: 27.03.2018;

26. Стивен Хубер Распространённые заблуждения о дизайне для сенсорных экранов. [Электронный ресурс]: CMS Magazine аналитический портал рынка разработок. – Режим доступа: http://www.cmsmagazine.ru/library/items/moblile/common-misconceptions-about-touch/, свободный, дата обращения: 19.06.2018;

27. Федеральный Государственный образовательный стандарт начального общего образования. Утвержден Приказом Министерства образования и науки Российской Федерации от 6 октября 2009 г. № 373;

28. Федеральный закон от 29 декабря 2012 г. N 273-ФЗ «Об образовании в Российской Федерации». Принят Государственной Думой 21 декабря 2012 года. Одобрен Советом Федерации 26 декабря 2012 года;

29. Филиппов С. А. Робототехника для детей и родителей. СПб.: Наука, 2011;

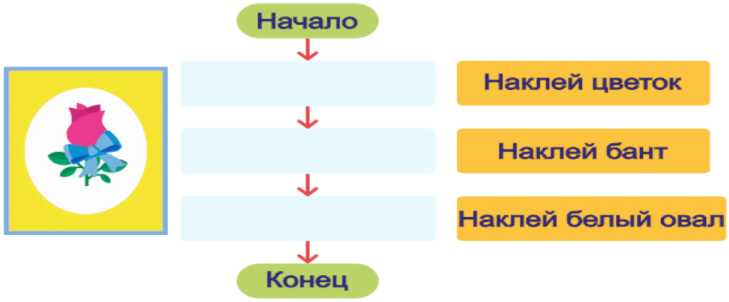
30. Цукерман Г.А. Что развивает и чего не развивает учебная деятельность младших школьников //Вопросы психологии. - 1998. - № 5. - С. 68-81;

31. Шимов И.В. Применение робототехнических устройств в обучении программированию школьников / И.В. Шимов // Педагогическое образование в России, 2013. – № 1. – С. 185–188;

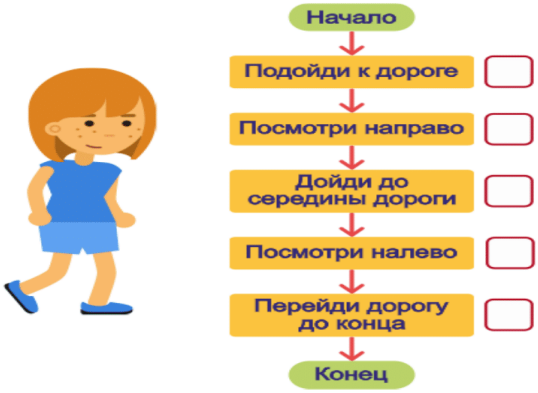
32. Berns K., Programmierung mit LEGO MINDSTORMSNXT / K. Berns, D. Schmidt. – eXamen.press, The LEGO Group 2010. – 246 s.

Приложение А. Тест для учеников начальной школы на предмет понимания основ алгоритмизации

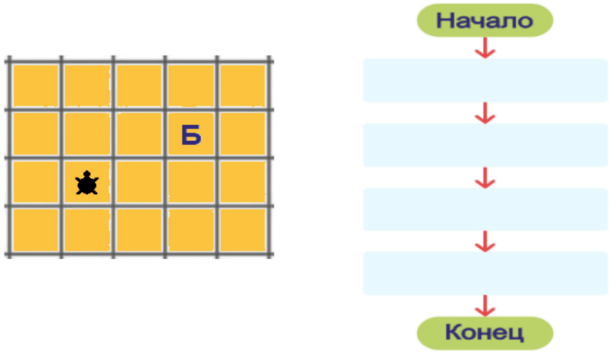
1. Напиши, в каком порядке нужно наклеивать детали картинки?



2. Отметь цифрами правильный порядок действий.



3. Составь план действий для черепашки, которая хочет переместиться в клетку «Б». Черепашка может перемещаться на одну клетку вперед и поворачиваться вправо и влево.



4. Сколько фруктов нужно зачеркнуть по данному алгоритму?

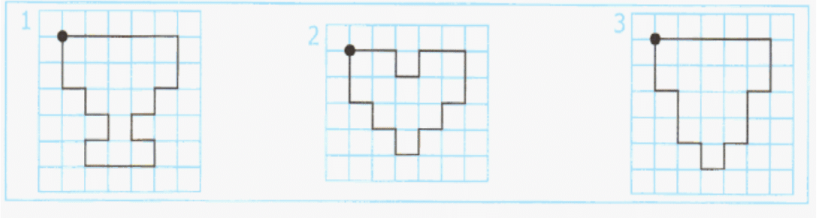
Ответ: _____

5. Какой вопрос нужно вставить, чтобы алгоритм «Иди в школу» был верным?

Ответ: _____

6. Какой из рисунков правильно изображен по алгоритму:

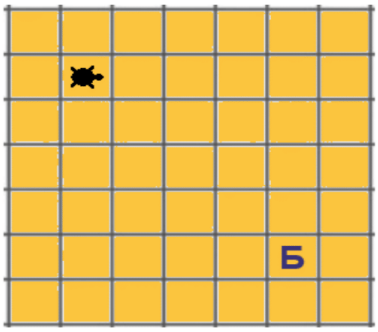
5→, 2↓, 1←, 1↓, 1←, 1↓, 1→, 1↓, 3←, 1↑,
1→, 1↑, 1←, 1↑, 1←, 2↑



Ответ: _____

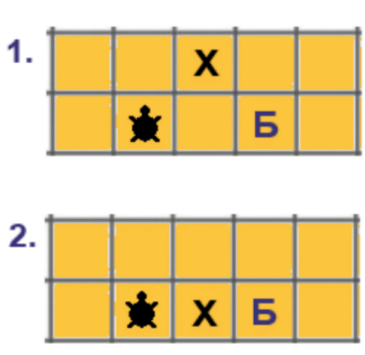
7. Сколько раз должна повторить эти действия черепашка, чтобы переместиться в клетку «Б»?

- Вперёд
- Вправо
- Вперёд
- Влево



Ответ: _____

8. Допиши алгоритм так, чтобы черепашка переместилась в точку «Б» как на первой, так и на второй картинке. Черепашка не может идти по клетке отмеченной крестиком!



Приложение Б. Рецензия на приложение «КоробОК» от преподавателей клуба робототехники SmartLab

РЕЦЕНЗИЯ

на программу «Korobok»

Программа состоит из шести уровней, в каждом из которых нужно составить алгоритм, при котором робот сможет передвинуть груз на нужное место. Апробация программы проходила в ООО "Смартлаб", на детях в возрасте от 6 до 12 лет. В ходе апробации, учитывали отзывы наших учеников и преподавателей, мы выделили преимущества и то, что, на наш взгляд, стоит доработать.

К **преимуществам** программы мы отнесли:

- двенадцать уровней, которые отличаются по сложности;
- случайная генерация уровней (точки положения радиации задается случайным образом, тем самым каждое прохождение уровня является уникальным);
- своя программная среда, с 6 блоками программирования;
- анимированная графика;
- подробная и анимированная инструкция к программе;
- сохранение прогресса (всегда можно вернуться к прохождению программы через некоторое время);

Стоит выделить, что основным преимуществом данной программы является то, что она в игровой форме, с использованием планшета (как показывает практика, детям планшет использовать интереснее, чем бумагу), дает возможность изучить основы алгоритмизации и программирования.

Прежде чем писать о том, что на наш взгляд, нужно доработать или исправить, стоит отметить, что данная программа является альфа-версией. Но мы надеемся, что работа над программой продолжится в дальнейшем.

На наш взгляд, следует доработать или исправить:

- Добавить обучающий уровень, обязательный к прохождению, вводящий в

суть игры и системы команд исполнителя с всплывающими инструкциями/подсказками.

- Систему обучения. Программа подробно объясняет, как взаимодействовать с предметами, но делается это только лишь в начале и в большом объеме, что является сложным для запоминания. Один из вариантов решения, на наш взгляд, делать обучение коротким, в 2-3 этапа, после каждого пройденного уровня. Так же по возможности отказаться от текста, что позволило бы использовать программу для детей младшего школьного и дошкольного возраста.
- Программная среда. Во время составления программы легко запутаться в интерфейсе. Тут две основных проблемы. Первое — это способ перетаскивания и удаления блоков программы. Детям гораздо привычнее переносить пальцем блоки на рабочую область и удалять с помощью переноса блока за пределы рабочей области. И второе — реализация циклов и ветвления. Блоки ветвления и цикла не выглядят едиными, цельными. При общем взгляде на программу плохо прослеживается структура программы.
- Обход правил программы. Некоторые уровни ученики проходили в обход задуманным правилам. Один из вариантов решения, ограничить количество блоков для этапов с использованием циклов.
- Анимирование и графика. Визуальное восприятие программы одно из важнейших для детей, особенного младшего возраста. Стоит изменить некоторые текстуры на их аналоги более высокого качества. А так же элементы навигации на более интуитивные.

В целом программа может быть полезна при обучении детей младшего школьного возраста и, несмотря на рекомендации по доработке, используется в образовательном процессе ООО «Смартлаб» в настоящее время.

Генеральный директор ООО
«Смартлаб» Д.С. Бартош

Приложение В. Сертификат об участии в форуме «Молодежь и наука»

Федеральное государственное
бюджетное образовательное учреждение высшего образования
«Красноярский государственный педагогический университет
им. В.П. Астафьева»

СЕРТИФИКАТ

Настоящий сертификат свидетельствует о том, что

Станковский Данила Вадимович

принял(а) участие в работе XIX Международного научно-практического форума студентов, аспирантов и молодых ученых «Молодежь и наука XXI века»

Председатель мероприятия — канд. пед. наук, доцент кафедры ИИТО Ломаско П.С.

Красноярск, 2018