Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ЛЕСОСИБИРСКИЙ ПЕДАГОГИЧЕСКИЙ ИНСТИТУТ –
филиал Сибирского федерального университета

Физико – математический
факультет

Высшей математики, информатики и естествознания
Кафедра

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

44.03.05 Педагогическое образование
44.03.05.08 Информатика и физика
код и наименование, направления, подготовки, специальности

ОБУЧЕНИЕ ВИЗУАЛЬНОМУ ПРОГРАММИРОВАНИЮ ОБУЧАЮЩИХСЯ В 10-11 КЛАССАХ НА ОСНОВЕ СОЗДАНИЯ ДИНАМИЧЕСКИХ ИГР
тема

Руководитель _____ Е. В. Киргизова
подпись инициалы, фамилии

Выпускник _____ Д.Н. Пестов
подпись инициалы, фамилии

Лесосибирск 2017

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ЛЕСОСИБИРСКИЙ ПЕДАГОГИЧЕСКИЙ ИНСТИТУТ –
филиал Сибирского федерального университета

Физико – математический
факультет
Высшей математики и информатики, естествознания
кафедра

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

44.03.05 Педагогическое образование
44.03.05.08 Информатика и физика
код и наименование, направления, подготовки, специальности

ОБУЧЕНИЕ ВИЗУАЛЬНОМУ ПРОГРАММИРОВАНИЮ ОБУЧАЮЩИХСЯ В 10-11 КЛАССАХ НА ОСНОВЕ СОЗДАНИЯ ДИНАМИЧЕСКИХ ИГР
тема

Работа защищена «21» июня 2017 г. с оценкой «отлично»

Председатель ГЭК _____ (подпись) С.С. Аплеснин (инициалы, фамилия)

Члены ГЭК
_____ (подпись) Е.В. Киргизова (инициалы, фамилия)
_____ (подпись) Н.Ф. Романцова (инициалы, фамилия)
_____ (подпись) А.М. Гилязутдинова (инициалы, фамилия)
_____ (подпись) А.М. Иванова (инициалы, фамилия)

Руководитель _____ (подпись) Е.В. Киргизова (инициалы, фамилия)

Выпускник _____ (подпись) Д.Н. Пестов (инициалы, фамилия)

Лесосибирск 2017

РЕФЕРАТ

Выпускная квалификационная работа по теме «ОБУЧЕНИЕ ВИЗУАЛЬНОМУ ПРОГРАММИРОВАНИЮ ОБУЧАЮЩИХСЯ В 10-11 КЛАССАХ НА ОСНОВЕ СОЗДАНИЯ ДИНАМИЧЕСКИХ ИГР» содержит 55 страниц текстового документа, 3 таблицы, 32 рисунка и 30 использованных источников.

ВИЗУАЛЬНОЕ ПРОГРАММИРОВАНИЕ, ПРОГРАММИРОВАНИЕ В 10-11 КЛАССАХ, СОЗДАНИЕ ДИНАМИЧЕСКИХ КОМПЬЮТЕРНЫХ ИГР.

Целью данной работы состоит в разработке методики обучения визуальному программированию учащихся на основе создания компьютерной игры.

Объект исследования: процесс обучения информатике.

Предмет исследования: особенности обучения визуальному программированию на основе создания учащимися динамических компьютерных игр.

Для достижения поставленной цели предполагается решение следующих задач:

1. Изучить и проанализировать теоретические и методические основы обучения школьников программированию.
2. Рассмотреть программные средства для обучения визуальному программированию
3. Систематизировать основные типы компьютерных игр и сред их разработки для обучения программированию;
4. Проанализировать среду для разработки технологии создания игр для обучения учащихся;
5. Разработать методику обучения программированию на основе создания динамической компьютерной игры.

Разработано содержание обучения, учащихся на основе создания компьютерной игры с использованием визуального программирования.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность исследования. Стремительное развитие научно-технического прогресса оказывает существенное влияние на различные сферы человеческой деятельности, в связи с интенсивным внедрением новых, постоянно меняющихся технологий, что, безусловно, предъявляет ряд требований к человеку, живущему в XXI веке. Получить специальные знания в соответствующих областях техники и технологии, сформировать определенную культуру научного мышления можно благодаря естественнонаучному образованию.

Информатика, сравнительно молодая отрасль науки. Пристальное внимание к информатике связано с бурным ростом объема человеческих знаний, который часто называют «информационным взрывом».

Информатика участвует практически во всех науках, она помогает визуализировать некоторые процессы недоступные человеческому глазу, моделировать опасные ситуации (опасные для жизни) или просто создавать модели для реализации их в жизнь, автоматизировать работу машин. Поэтому информатика чрезвычайна полезна в современно мире и занимает одну из лидирующих позиций в учебных заведениях.

На современном этапе образования цели направлены на формирование и всестороннее развитие творческой, активной личности, умений самостоятельно приобретать и применять знания. В школах, не специализирующихся на программировании, учащимся сложно изучать языки программирования, и они не обращают внимания на профессии связанные с информатикой ИКТ.

Современная молодежь очень интересуется игровой индустрией, дизайном, моделированием, анимацией. И с помощью данных «рычагов» можно подтолкнуть их на изучение информатики и в дальнейшем направить их на профессии, связанные с информационно-коммуникационными технологиями и программированием.

Практика показывает, что существуют динамические компьютерные игры (то есть игры, насыщенные движением, действием, изменением объектов и их

свойств с течением времени), которые школьники могут разрабатывать в процессе обучения. Разработка динамических игр, несложная с точки зрения программирования, может внести серьезный вклад в повышение мотивации к учению, преодоление когнитивных затруднений, интеллектуальное развитие школьников.

Объект исследования: процесс обучения информатике.

Предмет исследования: особенности обучения визуальному программированию на основе создания учащимися динамических компьютерных игр.

Цель исследования состоит в разработке методики обучения визуальному программированию учащихся на основе создания компьютерной игры.

Для достижения поставленной цели предполагается решение следующих задач:

1. Изучить и проанализировать теоретические и методические основы обучения школьников программированию.
2. Рассмотреть программные средства для обучения визуальному программированию
3. Систематизировать основные типы компьютерных игр и сред их разработки для обучения программированию;
4. Проанализировать среду для разработки технологии создания игр для обучения учащихся;
5. Разработать методику обучения программированию на основе создания динамической компьютерной игры.

Для решения поставленных задач применялись следующие методы исследования:

– теоретические: системный анализ отечественной и зарубежной психолого-педагогической, научно-методической литературы по педагогике и информатики; анализ существующих подходов к обучению информатике и программированию.

– эмпирические: обобщение опыта преподавания информатики в старшей школе; анализ содержания программ и учебно-методических комплексов по вопросам обучения программированию.

Выпускная квалификационная работа состоит из введения, двух глав, заключения, списка использованных источников и электронного приложения.

1 Теоретические аспекты обучения визуальному программированию на основе создания компьютерных игр

1.1 Основные подходы к обучению программированию в школьном курсе информатики

Программирование является одним из фундаментальных компонентов информатики, важность которого для современного общества не подлежит сомнению. Вопрос о его месте в школьной программе и эффективных методиках обучения программированию по-прежнему открыт.

В современном программировании сформировались четыре основных способа разработки алгоритмов. Такие способы в специализированной литературе получили название парадигм программирования. Выделяют четыре парадигмы: процедурная, объектно-ориентированная, логическая и функциональная. Под эту классификацию подходят все известные на сегодняшний день языки программирования. У реализации каждой парадигмы есть свои положительные и отрицательные стороны.

Как правило, выбор способа обработки информации определяется спецификой предметной области решаемой задачи. Важно отметить, что использование различных парадигм программирования существенно влияет на эффективность процесса создания компьютерных программ. Так, например, процесс разработки экспертной программной системы, как правило, существенно упрощается при использовании логической парадигмы (вместо традиционно применяемой процедурной или объектно-ориентированной). В то же время большинство компьютерных программ до сих пор разрабатывается без учета этого столь важного фактора.

В определенной степени парадигмы использовались в качестве основы для обучения программированию.

Процедурная парадигма являлась основой обучения в большинстве курсов программирования. Опыт этой работы отражен в работах таких

исследователей, как А.П. Ершов, А.Г. Гейн, В.М. Монахов и др [10].

Парадигма объектно-ориентированного программирования частично реализована в работах Е.Г. Андросовой, Н.Д. Угриновича, Н.Н. Истоминой и ряда других исследователей [20].

Построение курса обучения на основе логической парадигмы программирования реализовано в ряде работ С.Г. Григорьева, А.Г. Щеголева, Д.П. Федюшина [5].

В литературе практически не проработаны вопросы использования функциональной парадигмы в качестве основы для обучения программированию.

Объектно-ориентированное программирование (ООП) занимает на текущий момент центральное место среди парадигм программирования. Это связано с несколькими аспектами:

- большинство современных пользовательских сред программируется на объектно-ориентированных языках;
- ООП более естественно для человека, чем структурное программирование, будь то программист или пользователь, работающий с программой, написанной на объектно-ориентированном языке;
- объектный подход дает возможность работать с накопленными библиотеками классов, на концепции которых основывается построение современных средств разработки.

Традиционно в школе преподается процедурное программирование. Считается, что первый язык программирования (это, как правило, Паскаль) должен иллюстрировать именно алгоритмические структуры, а затем уже можно осваивать более сложные, объектно-ориентированные языки. Более того, в последние годы, в связи с попытками унификации и переходом к ЕГЭ в компьютерной форме, снова активное распространение получает учебный алгоритмический язык (система «КУМир»). Несмотря на его несомненные преимущества, прежде всего, близость к родному языку учащихся и родственные отношения с Паскалем, следует заметить, что этому языку уже

более 50 лет. Современным требованиям к средствам разработки он не отвечает и совместимости с современными языками не имеет.

Такой подход к обучению программированию представляется спорным. Последующий переход от процедурной парадигмы к объектной может быть связан у учащихся со значительными когнитивными затруднениями.

Появившаяся в школьных учебниках тенденция использования объектно-ориентированных языков типа Object Pascal и Visual Basic [12], которые эволюционировали из процедурных языков, еще больше затрудняет понимание объектно-ориентированной парадигмы.

Рассмотрим факторы, определяющие целесообразность изучения ООП в школе.

Первым фактором является возможность повышения интереса учащихся к изучению программирования, так как появляется возможность более наглядного и разнообразного представления процесса и результатов своей деятельности с помощью различных объектов, с которыми учащийся может непосредственно работать.

Следующим фактором становится предпрофессиональная ориентация части учащихся, так как ООП на современном этапе – ведущее средство в профессиональном программировании.

Еще одним фактором является необходимость более глубокого понимания идеологии современных пользовательских сред (MS Office, графические редакторы и т.д.), реализованных на основе объектно-ориентированной парадигмы.

Четвертым фактором становится сама сущность ООП, которая позволяет в процессе обучения большее внимание уделять процессу использования готовых программ и классов, созданных в объектно-ориентированной среде. Такой подход способствует формированию у учащихся понимания механизмов работы программы, и как следствие, обеспечивая повышение интереса учащихся к предмету и их успеваемости.

И последним фактором стала необходимость освоения принципов

объектно-ориентированного программирования (особенно объектной декомпозиции), которая способствует развитию системного подхода к наблюдению и исследованию объектов окружающего мира.

Использование объектно-ориентированного языка в качестве основы обучения школьников программированию не отрицает, но дополняет использование традиционных алгоритмических структур. Все изучаемые в школьном курсе программные конструкции (условное ветвление, цикл, массивы и т.д.) в полной мере реализуются на любом языке ООП.

Также в программах на объектно-ориентированном языке могут быть частично представлены учащимся и реализованы элементы логической и функциональной парадигм.

Изучение современных информационных технологий в целом, и ООП в частности, в настоящее время затруднено нехваткой специальных учебно-методических материалов. Следовательно, актуальной является задача разработки методик, которые помогли бы учителю в преподавании, а учащимся в успешном овладении современными средствами программирования.

Рассмотрим традиционный дидактический подход к обучению программированию в целом, и в общеобразовательной школе в частности.

Начиная с самых первых учебников по программированию, появились неизменные до настоящего времени каноны преподавания. Вначале рассматриваются переменные, типы данных, основные встроенные функции и синтаксические конструкции (операторы), затем структурные типы и средства работы с ними. Такой подход оправдывает себя, если речь идёт об обучении специалиста, да и то лишь в случае хорошо подготовленного, мотивированного на изучение языка программирования. В школе изложение материала в этом ключе малоэффективно. Оно не несет смысловой нагрузки для ученика, поскольку ему заранее даются ответы на ещё незаданные вопросы, а предлагаемые задания не являются личностно значимыми.

В учебниках программирования существуют два различных подхода: описание языка и описание алгоритмов.

Современные школьные учебники информатики, включающие раздел «Алгоритмизация и программирование», рассчитанные на базовый уровень, работают с описанием алгоритмов и комплексом исполнителей, изучение которых может происходить и без участия вычислительной техники, такие как Н.В. Макаровой. Учебники, рассчитанные на профильный уровень, например, Н.Д. Угриновича и А.Г. Гейна, последовательно излагают структуру языка программирования и предлагая поэтапное ее изучение с помощью репродуктивного метода [12].

Средством адаптации к общеобразовательной школе служит «задачный» подход, когда изучение каждой языковой конструкции предшествует постановке задачи, решаемой затем с применением этой конструкции. Задачи эти, как правило, математические, редко связанные друг с другом. Большинство из них короткие, ограниченные и предусматривают «нахождение» решения в течение одного урока. Некоторые из них связаны в комплекс практикумов, как, например, в учебнике Н.Д. Угриновича, каждый из которых реализуется на нескольких уроках, связанных общей темой.

Результатом решения этих задач являются небольшие прикладные программы, способные либо находить решения математических выражений с разными данными, либо заполнять диалоговые формы и интерпретировать полученные данные, либо выполнять некоторый ряд логических действий. Классическая методика, воплощенная в этих учебниках, выглядит следующим образом: учащемуся предлагается интересная, по мнению авторов, мотивирующая на учение задача, которую невозможно реализовать имеющимися в его распоряжении, уже усвоенными средствами.

Предложив в теории, альтернативный оригинальный способ решения и обнаружив его неэффективность, учащийся приступает к усвоению лекции об очередной программной структуре, которая призвана помочь в решении задачи. Получив необходимые теоретические знания, учащийся старается найти их приложение к решению поставленной задачи. И, возможно, с помощью учителя и учебника, находит решение, получив в результате программный продукт.

Несмотря на то, что в этой схеме формально присутствуют частично поисковый и проектный методы, из практики известно, что этапы поиска решения уже имеющимися средствами и поиск приложений конструкций к задаче в подавляющем большинстве случаев на уроках отсутствуют. Сложность предмета, ограниченность учащихся во времени, нежелание большинства учителей отклоняться от темы не позволяют использовать активные методы обучения. Таким образом, структура урока сводится к этапам постановки задачи, лекция, нахождения решения, то есть применяется репродуктивный метод.

Для успешного обучения программированию, нужны дополнительные методы обучения, подходы упрощения подачи материала, а также дополнительной мотивации учащихся. Одним из средств упрощения подачи материала является его визуализация. Рассмотрим существующие средства для обучения визуальному программированию.

1.2 Программные средства для обучения визуальному программированию в курсе информатики

Для человека, живущего в рамках современной цивилизации характерно стремление к визуальному восприятию информации. Данное явление приводит к тому, что в процессе информационной коммуникации зрительный знак преобладает над текстовым. Предмет информатика не является исключением. Применение в процессе обучения визуализации процесса, способствует частичному решению данной проблемы.

Визуальное программирование (от лат. visualis – зрительный) – это технология программирования, предусматривающая создание приложений с помощью наглядных средств.

Следует отметить, что концепция визуального программирования реализуется во многих современных инструментальных средах разработки приложений. Большинство известных компаний – лидеров на рынке

современного программного обеспечения, программных средств разработки и конструирования имеют в своем арсенале системы, поддерживающие технологию визуального программирования. Графический визуальный интерфейс абсолютного большинства современных программных приложений на сегодняшний день стал обычным явлением, поскольку он делает «взаимодействие» пользователя с программным продуктом понятным и удобным. Графические изображения на элементах управления позволяют пользователю интуитивно разбираться в назначении этих элементов. Для визуализации интерфейсов программного обеспечения существует целый ряд специально разработанных элементов интерфейса – визуальных компонент, позволяющих отображать различную информацию и осуществлять управление программой в целом.

Тема «Алгоритмизация и основы объектно-ориентированного программирования» входит в содержание учебника «Информатика и ИКТ» автора Угриновича Н. Д., для 10 класса профильного уровня. На углубленном уровне обучения информатике тема, связанная с изучением визуального программирования, также изложена в учебнике «Информатика» для 11 класса авторского коллектива Семакина И. Г. и др. и в учебнике «Информатика» для 11 класса авторов Полякова К. Ю. , Еремина Е. А. [27], а в качестве средства программирования выступает Delphi. Отметим, что в Visual Basic и Delphi выполняется визуальное построение интерфейса программы, но не самого кода.

В качестве определяющих элементов процесса визуализации выделяют:

– визуализируемую модель – модель, которая подвергается отображению с целью возможности изменения ее структуры или ее параметров (либо параметров ее отдельных частей);

– панель элементов – окно, содержащее набор элементов, из которых строится визуальная модель. Обычно элементы разделяются по их назначению на отдельные группы, размещающиеся на отдельных закладках окна инструментов;

– окно свойств – окно, в котором отображаются параметры (свойства)

выбранного элемента визуальной модели.

Визуализируемой моделью в Visual Basic и Delphi является окно (форма, диалог) Windows, а не код программы. Обычной практикой является визуализация работы с элементами интерфейса, когда в качестве объектов визуализации рассматриваются визуальные компоненты, из которых состоят формы (окна и диалоги) интерфейса программы. Но и операторы программы можно рассматривать как объекты визуализации. В этом случае параметры операторов и функций программы можно настраивать при помощи окна свойств.

Итак, визуальное программирование обладает достоинством наглядного представления информации и гораздо лучше соответствует природе человеческого восприятия, чем методы традиционного, текстового программирования. Концепция визуального программирования реализована во многих современных средах разработки программных систем. Изучение основ визуального программирования также находит отражение в школьном курсе информатики на базовом, профильном и углубленном уровнях.

Рассмотрим основные среды разработки с применением технологии визуального программирования, которые используются в школе:

1. Визуальная объектно-ориентированная среда Kodu. Среда предоставляет возможность показать учащимся, что программирование – это вовсе не скучное занятие, оно может быть интересным и увлекательным. И любой учащийся сможет проявить творческие навыки в процессе создания своей игры. Все, что требуется для разработки игры – создать игровой мир, в котором будут обитать добавленные пользователем персонажи, и взаимодействовать с внешним миром по установленным пользователем правилам, конечно, не забывая про законы физики. Причем составляются эти правила весьма незамысловатым способом: в среде используется конструкция «когда X делать Y». А методами и полями в этой конструкции служат примитивные «карточки» с действиями или состояниями объекта [1].

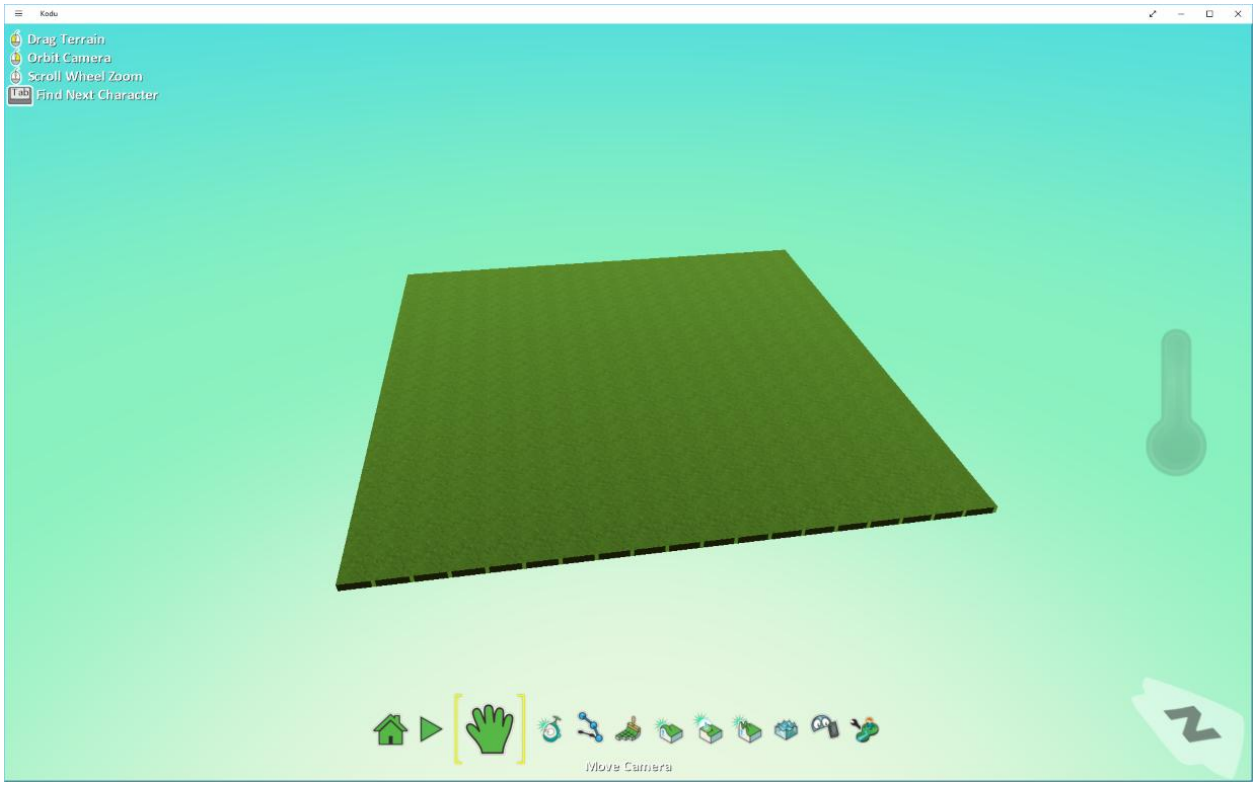


Рисунок 1 – Среда Kodu Game Lab

При первом запуске Kodu Game Lab (Рисунок 1) можно выбрать один из уже существующих демонстрационных миров, в которые входят уровни с начальным обучением программирования в данной среде, а также создать игру, начав с пустого мира (Рисунок 1). Внутри мира могут жить абсолютно разные объекты: «коду», яблоки, монеты, деревья, камни, черепахи, корабли, пушки и т.д. Изначально объекты уже имеют некое поведение. Так, например, пушка, добавленная в игровой мир, может «улыбаться», опрокидываться на спину и обратно. Также действия объекта могут быть добавлены вами – например, при нажатии на клавишу «пробел» можно заставить пушку стрельнуть, а при нажатии на клавиши W, A, S, D – передвигаться и поворачиваться вокруг своей оси. Снаряд при столкновении с каким-либо объектом взрывается, нанося ущерб «здоровью» объекта, в которого он попал. Когда показатель «здоровья» у объекта достигнет нуля, последний уничтожается. Все это и многое другое уже изначально заложено в ряд функциональных возможностей. Поэтому, чтобы получить работающий вариант своей игры, надо не так уж и много – разместить объекты и наделить их хоть каким-то минимальным поведением.

Перед тем, как писать правила нашим объектам и размещать их, необходимо создать и настроить ландшафт. Для этого используются соответствующие параметры, включающие большое разнообразие текстур: от гор до равнин, от водоемов до дыр в земле, в которые объекты могут упасть. Все это настраивается различными видами кисточек: квадратная, круглая, их аналоги, позволяющие рисовать прямые линии, а также волшебная кисть, с помощью которой можно изменить состояние всей области одинакового цвета, части которой контактируют друг с другом. Огромный набор текстур предоставляет возможность каждому пользователю придумать уникальный ландшафт для будущей игры. Используя разнообразные цветные текстуры для создания воды, начинающий программист может создать виртуальный мир, в ко действие может происходить и на суше, и на воде, и под водой. В этом плане Kodu Game Lab является очень хорошей площадкой для воплощения своих творческих способностей и воображения.

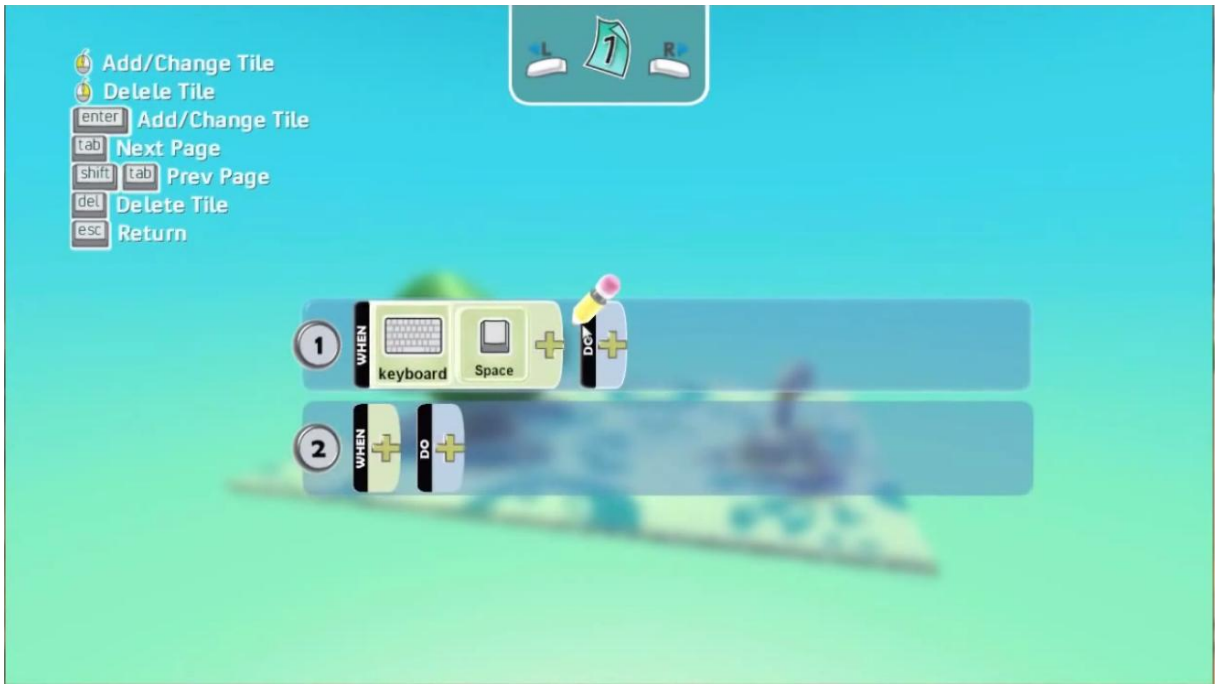


Рисунок 2 – Реализация многоуровневых условий

После создания ландшафта будущей игры, начинается этап размещения и программирования поведения объектов. Окно с программой объекта, в которой описывается его поведение, имеет целых 12 страниц – 12 состояний объекта. Kodu Game Lab позволяет создавать многоуровневые условия и действия

(Рисунок 2). Для этого необходимо подвинуть строчку с поведением правее, чем та, что над ней, организовав многоуровневый порядок действий. Самое простое, с чего можно начать – научить объект ходить.

И уже на этом этапе понятно, что игру можно сделать даже на несколько человек, которые будут играть на одной клавиатуре, так как действия можно задать почти на все клавиши. Уже только эта возможность открывает огромные границы для творчества. Далее можно научить созданный объект еще какому-либо действию, например, эмоциям при каких-то определенных ситуациях. Отдельно стоит отметить такой параметр у объектов, как «родитель». При его активации появляется возможность создать до ста экземпляров данного объекта с точно такой же программой поведения, с точно такими же параметрами самого объекта (скорость, урон, количество здоровья и т.д.). Это способствует уже какому-то введению в объектно-ориентированное программирование. Учащийся будет понимать, что не обязательно создавать десятки клонов, достаточно просто создать экземпляры одного объекта. Также присутствует возможность копировать программы поведения, но, к сожалению, только по одной строчке (одному поведению), либо же по одному многоуровневому условию с подчинением. При создании мира, в правой области экрана находится термометр. Данный индикатор показывает объем используемых ресурсов создаваемой игры. Как только объектов станет очень много, или карта будет чересчур большая, термометр оповестит, что выбран лимит ресурсов – ничего нового уже нельзя будет добавить, пока что-нибудь не удалить. Еще одна очень полезная возможность среды – организация переходов с уровня на уровень (переходить из одного мира в другой). При достижении игроком пяти игровых очков, загрузится следующий уровень игры. Визуальная объектно-ориентированная среда Kodu Game Lab позволяет создавать не только обычные шутеры от перового лица, но и многоуровневый сюжетный квест, в котором может быть все, что угодно.

В заключении стоит отметить, что в нашем веке быстро прогрессирующих технологий никуда без навыков мышления, а среда Kodu

Game Lab позволяет начать формирование алгоритмического мышления уже с раннего возраста.

2. *Lego Mindstorms NXT робототехнического комплекса среда NXT-G, EV3-G.*

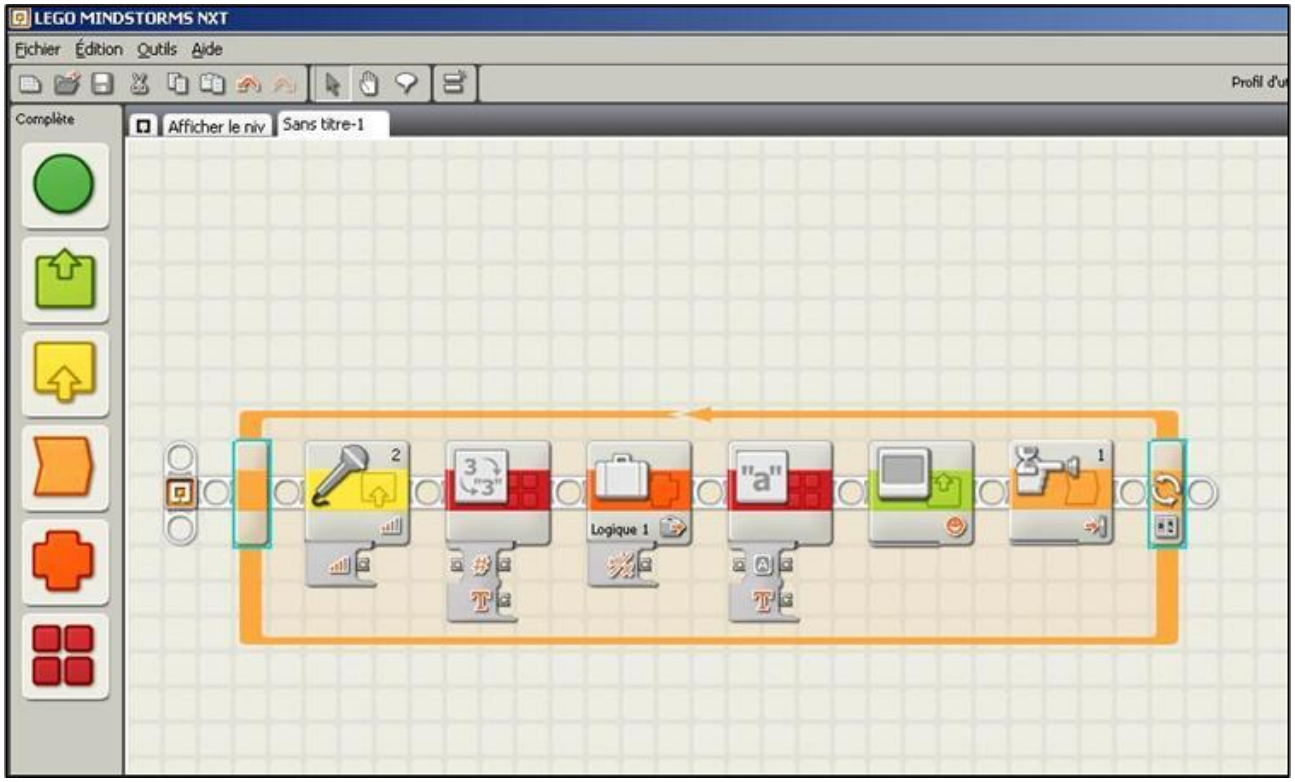


Рисунок 3 – Программная среда NXT-G

В школьной информатике активно используется понятие «Исполнитель», как некая сущность, которая выполняет команды, описанные в программе. В качестве исполнителя в российских школах активно внедряются робототехнические конструкторы, самым популярным из которых является конструктор Lego Mindstorms NXT. Он позволяет из блока управления, моторов, сенсоров и соединительных деталей собирать роботов, способных под управлением программы сложным образом взаимодействовать с окружающим миром. Существует довольно много систем, позволяющих программировать такие роботы, как текстовых, так и визуальных.

Идея использовать роботов при обучении информатике родилась неслучайно. Кроме того, даже представить себе программу не так просто – каждый человек «видит» программу по-разному. Современные технологии

позволяют создавать недорогие механические устройства, управляемые загружаемой в них программой, либо непосредственно с компьютера, поэтому идея использования материальных исполнителей в школьной информатике получила второе рождение, из-за чего получил распространение конструктор Lego Mindstorms NXT.

Для преподавания информатики с использованием этого конструктора существуют методические пособия. Робототехнический конструктор довольно сложно программировать: из набора деталей могут быть собраны самые разные конструкции, поэтому программировать приходится в терминах оборотов моторов, подключённых к определённым портам управляющего блока, а не в терминах движений и поворотов.

Это, безусловно, делает процесс обучения более творческим, поскольку школьники могут собрать своего собственного исполнителя, но и более сложным с точки зрения написания для этого исполнителя программ. Проблема сложности программирования преодолевается использованием наглядных визуальных языков и удобных графических редакторов для составления программ из блоков, представляющих элементарные команды, такие как «включить мотор», «гудок» и т.д. (Рисунок 4)

Рисунок 4 – Алгоритм программирования в среде NXT-G

Таким образом, начинающие учащиеся работают с графическими языками программирования, а более опытные постепенно переходят на текстовые языки. В комплекте с конструктором поставляется графическая среда программирования NXT-G, поэтому визуальные языки среди использующих Mindstorms NXT весьма популярны.

3. Scratch. В российском школьном образовании Scratch уже занял достойное место. Scratch - это разработка группы Lifelong Kindergarten MIT Media Lab, специально созданный при поддержке правительства США для поддержки обучения школьниками визуального программирования. «Этот пакет позволяет «собирать» сценарий игры или анимации из блоков, но не из пиктограмм, как в большинстве конструкторов, а из стандартных текстовых фраз на формализованном естественном языке (английском) (рисунок 5) [22].

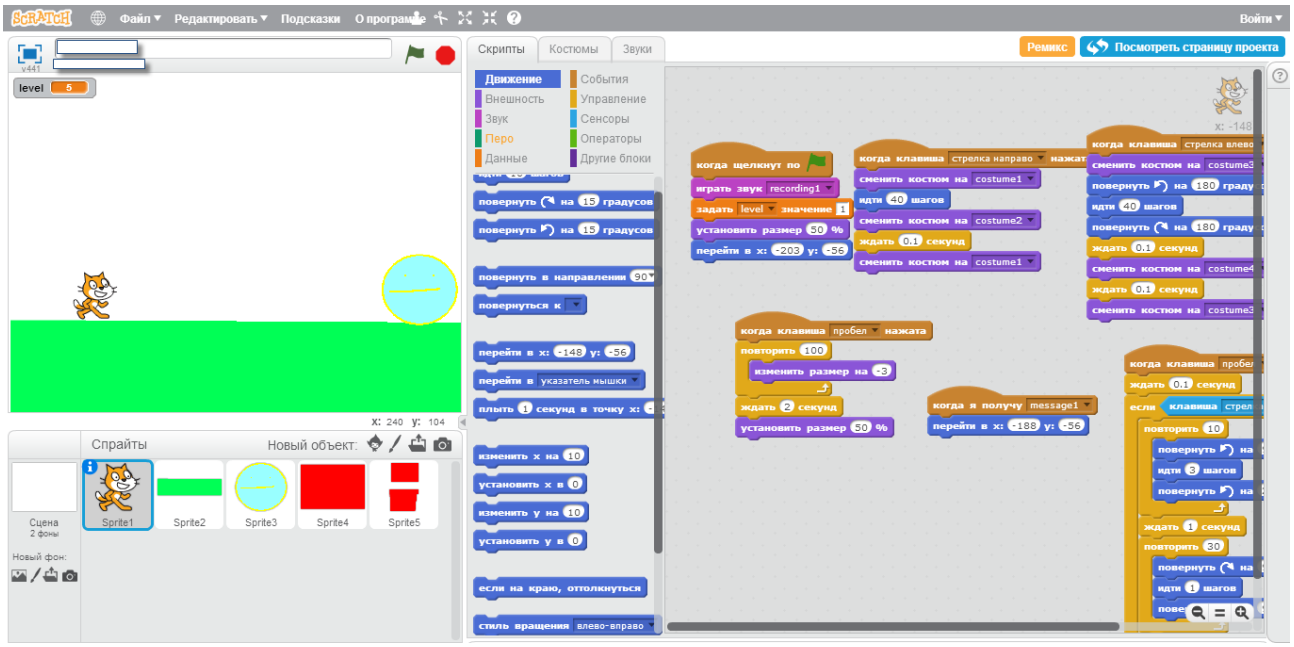


Рисунок 5 – Среда Scratch

Эта интересная концепция имеет хорошую методическую поддержку разработчиков, широко применяется в школах США, Европы и даже ряде программ дополнительного образования в отечественной педагогике. Однако данный конструктор в настоящее время не поддерживает ряд важнейших программных конструкций (процедуры и функции, передачу параметров, рекурсию, наследование и т.д.), что делает его использование в обучении школьников программированию весьма ограниченным.

Вышеизложенное позволяет сделать вывод о том, что необходимо определить принципы и критерии отбора компьютерных игр и сред их разработки; определить технологию и этапы создания игры в процессе

обучения программированию; разработать методику обучения программированию.

1.3 Создание компьютерных игр как средство обучения программированию

В последние три десятилетия множество зарубежных исследователей занималось изучением того, как естественная увлеченность учащихся цифровыми (в том числе компьютерными) играми может быть использована для достижения образовательных целей. Компьютерные игры - объект серьезных исследований в психологии, педагогике, информатике и социологии. Все эти науки пытаются связать компьютерные игры с обучением.

Основным фактором, влияющим на эффективность использования компьютерных игр при решении образовательных задач, является мотивация.

Мотивация - весь комплекс факторов, направляющих и побуждающих поведение человека.

Термином «мотивация» могут обозначаться два явления:

– система мотивов определенного человека;

– система действий по активизации мотивов определенного человека.

В аспекте создания компьютерных игр особенно интересна концепция внутренней (англ. «intrinsic») мотивации Э. Деси. Также в литературе ее обозначают как «мотивацию, проистекающую из процесса» или преоперациональную в теории развития Пиаже. Это «стремление совершать деятельность ради нее самой, ради награды, которая содержится в самом процессе деятельности» [8].

Рассмотрим основные аспекты обучения программированию на основе создания компьютерных игр:

1. Идентифицируемость заданий – степень, в которой работа требует завершения целостной и определенной задачи или этапа работы от начала и до конца с видимым результатом. Исходя из этой характеристики, компьютерные игры хорошо подходят в качестве учебной задачи, если создание их четко разделено на этапы, каждый из которых имеет видимый, осознаваемый в начале этапа выполнения, законченный результат.

2. Значимость задания – степень важности результата (получаемой игры) для учащегося. Этот показатель, как правило, очень высок. Создание собственной игры «с нуля» является для подростка важнейшим и значимым достижением.

3. Уникальность задачи по созданию компьютерной игры, помимо высокой внутренней мотивации на результат, заключается в том, что учащиеся детально знакомы с требованиями к конечному продукту, могут в каждый момент разработки проекта предположить результат следующего этапа. Это связано с их предыдущим опытом знакомства с компьютерными играми в качестве потребителей.

В создании компьютерных игр учащиеся легко могут рефлексировать, опираясь на свой значительный опыт в качестве игроков в компьютерные игры.

Использование компьютерных игр в обучении программированию в настоящее время можно условно выделить несколько подходов:

– создание дополнительных уровней к играм, созданным профессионалами, в том числе к играм с образовательным содержимым (обучающим играм);

– использование визуальных конструкторов игр, в том числе со встроенным языком программирования;

– введение небольших логических игр как элементов курса обучения какому-либо языку программирования.

Для определения условия и критерия отбора компьютерных игр и сред разработки необходимо рассмотреть существующие классификации компьютерных игры по разным основаниям.

Наиболее известной и использующейся является классификация компьютерных и мобильных игр по жанрам.

«Жанр - это совокупность произведений, объединяемых общим кругом тем или предметов изображения; авторским отношением к предмету, лицу или явлению: способом понимания и истолкования».

Рисунок 6 – Классификация жанров

Постоянное развитие отрасли производства компьютерных игр привело к появлению множества игр комбинированного типа, сочетающие в себе несколько жанров и видов деятельности. Современные игры часто содержат элементы игры-симулятора, квеста, логической игры и охватывают множество аспектов психической и моторной деятельности человека.

Объектная модель игры, наглядная реализация множественности отношений и событий полностью могут быть реализованы только в динамическом типе игр. Динамичность, в связи с этим, станет одним из критериев отбора типа компьютерной игры для обучения программированию.

Рассмотрим классификацию компьютерных игр, разработанной в области педагогики-психологии, автора А.Г. Шмелева [15].

Рисунок 7 – Классификация А.Г. Шмелев

Основанием данной классификации является тип мышления, стимулируемый в процессе игры, доминирующее свойство психики. Например, азартные игры требуют от игрока интуитивного, иррационального мышления. Влияние компьютерных игр на субъекта, психологические особенности процесса игры в данном исследовании имеют опосредованное значение. Следовательно, данная классификация не имеет прямого отношения к определению критериев отбора компьютерной игры для разработки. Отметим лишь, что естественными рамочными условиями при отборе типа игры и, впоследствии, сюжетов игр учащихся должны стать толерантность и ограничение деструктивных влияний на психику.

Рассмотрим классификацию игры по цели процесса и наличию движения объектов (Рисунок 8).

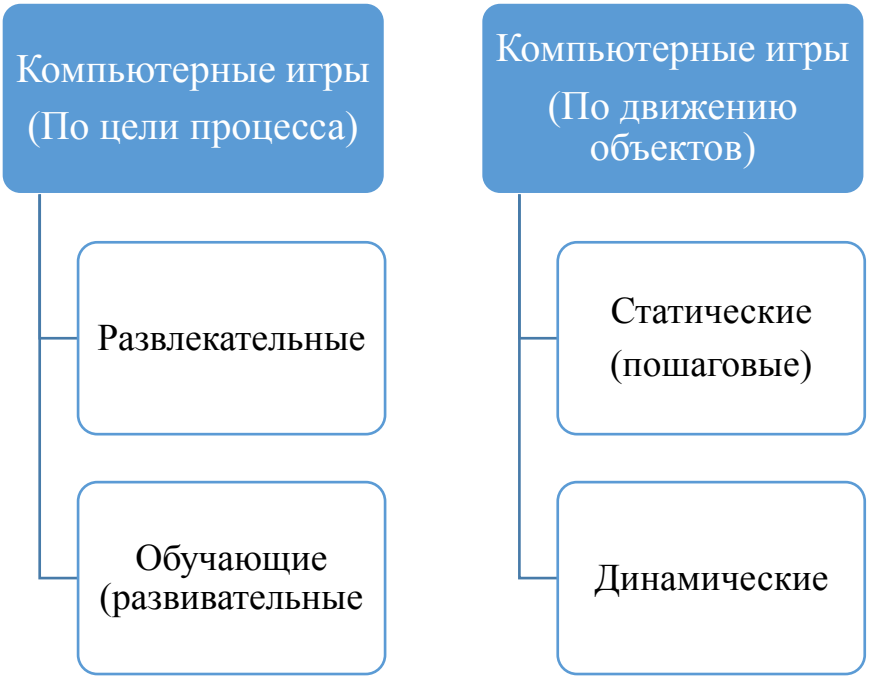


Рисунок 8 – Классификация игр

Классификация игры по наличию движения объектов (Рисунок 8). В этом аспекте в целях формирования высокой внутренней мотивации следует выбрать развлекательный тип игр.

Объектная модель игры, наглядная реализация отношения объектов и событий могут быть реализованы только в динамическом типе игры.

Для того что бы подойти к обоснованию компонентов методики обучения программированию, необходимо рассмотреть, какие условия и критерии лежат в основе отбора компьютерных игр и сред для обучения школьников программированию.

Основные условия при отборе игр:

1. Условие унификации типа игры и индивидуализации учащихся. На основании критериального отбора должен быть определен единый тип игры, наиболее подходящий для обучения школьников программированию. Многие типы игр по структуре, содержанию, схеме взаимодействия объектов не позволяют логично включить в процесс разработки все необходимые программные конструкции. Другие типы, напротив, требуют избыточно

сложного процесса разработки, что отрицательно влияет на понимание материала, зачастую превращая процесс обучения программированию в процесс конструирования из «черных ящиков».

2. *Условие актуальности среды разработки.* Данное условие обозначает ориентацию на современные средства, результаты использования которых учащиеся могут наблюдать в интересующих их сервисах и приложениях. Демонстрация и анализ учителем аналогичных или близких к планируемым результатов является важным элементом мотивации и целеполагания учащихся. В то же время, работа в устаревших приложениях с неактуальным интерфейсом резко снижает заинтересованность учащихся программированием. Это отчетливо прослеживается при работе с традиционными «школьными» языками программирования, необходимыми для подготовки к ЕГЭ на современном этапе [9].

Определим и обоснуем важнейшие критерии отбора компьютерных игр для обучения школьников программированию.

Критерии отбора компьютерных игр:

1. Возможность реализации всех программных конструкций, предусмотренных стандартом;
2. Минимальная сложность реализации игры нужного типа;
3. Наличие системы отладки программы;
4. Наличие объектно-ориентированного языка;
5. Простота и логичность синтаксиса языка программирования;
6. Возможность реализации всего проекта на языке программирования;
7. Дружественность интерфейса;
8. Актуальность, применимость;
9. Популярность среди детей данной возрастной группы;
10. Возможность демонстрации результата в сети Интернет.

Оценим перечисленные среды разработки компьютерных игр по данным критериям, используя для каждого критерия шкалу от 0 до 2 (таблица 1).

Таблица 1 – критерии отбора компьютерных игр

№	Среда разработки	Критерий										Итого
		1	2	3	4	5	6	7	8	9	10	
1.	Kodu	2	2	1	1	2	2	2	1	1	1	15
2.	Scratch	1	2	0	0	2	2	2	1	0	1	11
3.	GameMaker	2	0	2	2	1	2	2	1	1	2	15
4.	JavaScript	2	1	1	2	1	0	0	2	1	1	11
5.	Flash	2	1	1	2	1	0	2	1	1	1	12
6.	Visual Basic	2	1	1	2	1	1	1	1	0	0	10
7.	PascalABC.NET	2	0	1	1	1	1	1	0	0	0	7
8.	Basic	2	1	1	1	1	1	1	0	0	0	8
9.	C++	2	1	1	2	1	1	1	2	2	2	15
10.	UnrealEngine	2	2	2	2	2	2	2	2	1	0	17
11.	Unity 3D	2	1	0	2	1	1	0	2	1	0	10

Можно сделать вывод о том, что для обучения школьников в старших классах программированию наиболее подходящим является среда Kodu, GameMaker, C++ и UnrealEngine.

В данном параграфе были сформулированы условия отбора компьютерных игр и сред их разработки. Исходя из этих условий, исследованы имеющие классификации игр и сред разработки, критерии отбора. Выявлен наиболее подходящий тип игры – трехмерная игра от третьего лица и среда для создания игр школьниками: Unreal Engine 4.

Далее, основываясь на результатах отбора, необходимо разработать методику обучения и описать этапы создания школьниками в процессе обучения программированию компьютерной игры.

2 Методические особенности обучения программированию, основанная на создании учащимися компьютерных игр

2.1 Особенности технологии визуального программирования в среде Unreal Engine 4

Unreal Engine 4 – игровой движок, разрабатываемый и поддерживаемый компанией Epic Games.

Игровой движок – центральный программный компонент компьютерных и видеоигр или других интерактивных приложений с графикой, обрабатываемой в реальном времени (Рисунок 9).

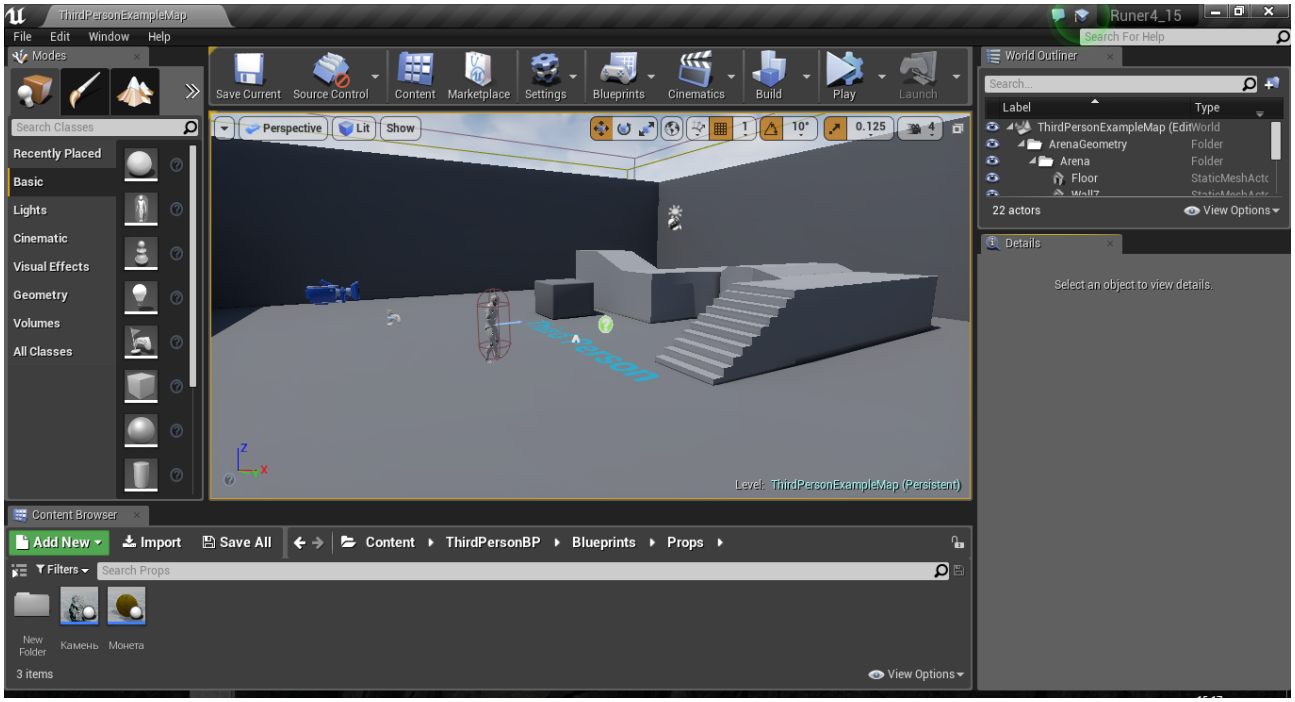


Рисунок 9 – Редактор Unreal Engine 4

Blueprint (Блупринты) – это скриптовая система в Unreal Engine 4, которая представляет собой визуальный интерфейс для создания элементов игрового процесса. Система очень гибкая и очень мощная, и позволяет использовать концепцию и почти полный потенциал программирования (Рисунок 10).

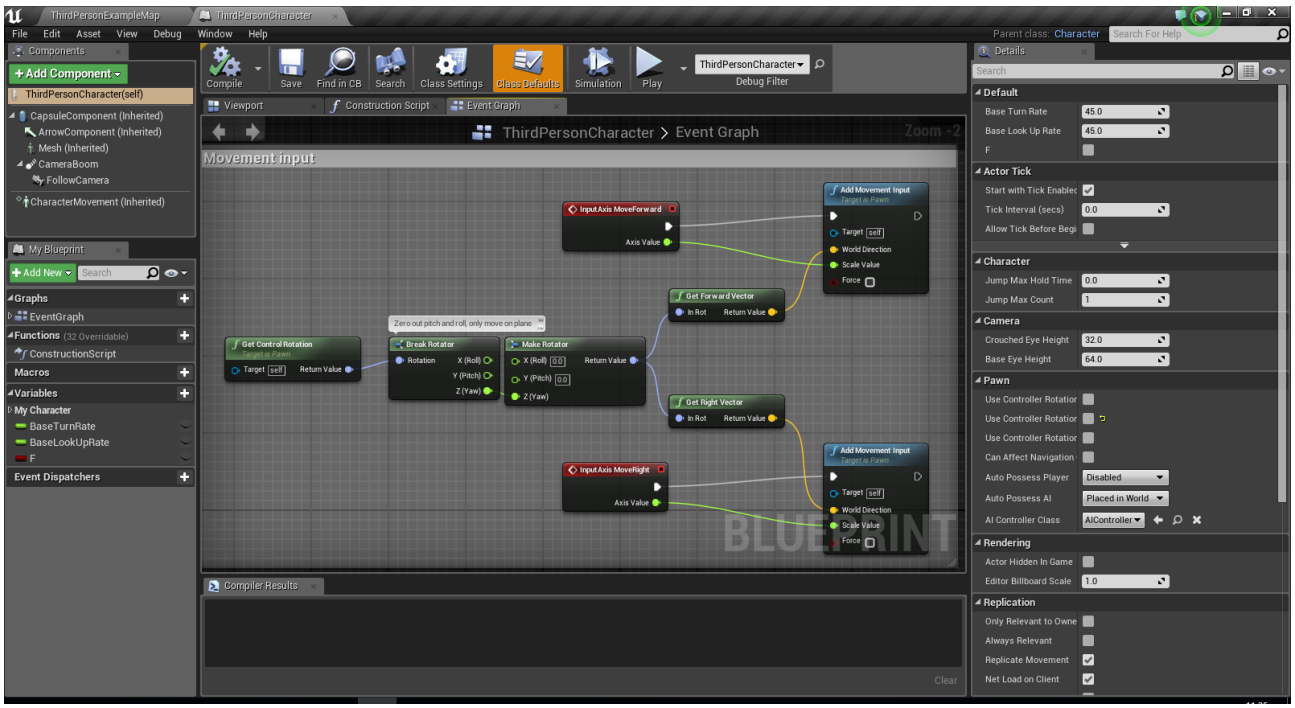


Рисунок 10 – Редактор Blueprint

Блупринты являются типом объектов, которые предоставляют пользователю интуитивную систему для создания специальных типов объектов, которые вследствие можно поставить на сцену и используются для создания игровой логики уровня или логики уровня, при этом не написав ни строчки кода, что облегчит написание логики не только программистам, но и дизайнерам, кто не силен в программировании.

Блупринты используют встроенную в редактор систему для визуального построения логических последовательностей. Соединяя блоки, события, функции и переменные, создавать алгоритм исполнения программы.

Основные типы Блупринтов: Level Blueprint и Class Blueprint.

Level Blueprint используется для манипулирования объектами на сцене в процессе игры. Level Blueprint’ы так же могут взаимодействовать с Class блупринтами, которые имеются на сцене.

Class Blueprint позволяют создать сложные объекты для последующего размещения на сцене, такие как открывающиеся двери, ящики с предметами, кнопки и т.п. На изображении сверху напольная кнопка и дверь являются разными блупринтами и содержат определенный скрипт для взаимодействия с игроком, проигрывания анимации и звука, открытии дверей и так далее. Class

Blueprint могут быть полностью индивидуальными, а значит, для их работы не обязательно воздействовать извне, и они могут работать и производить какие-либо действия сами по себе.

Классовый Блупринт (Class Blueprint) позволяет хранить в себе набор объектов, а также логическую последовательности, которая будет выполняться внутри данного Блупринта в процессе игры. Блупринты создаются в специальном редакторе, а логика пишется на визуальной основе, вместо написания кода вручную. После создания Блупринта и написания всей логики внутри него, Блупринт может быть помещен на сцене как объект.

Редактор Blueprint состоит из таких компонентов как:

1. Components (Компоненты)

Компоненты – объекты, которые могут быть добавлены в Блупринт (Рисунок 11). Этим объектами могут быть статические объекты, источники света, динамические объекты и т.п. Компоненты можно добавить через меню Components во вкладке с таким же именем. Использовать компоненты в логике блупринта можно с помощью панели переменных.

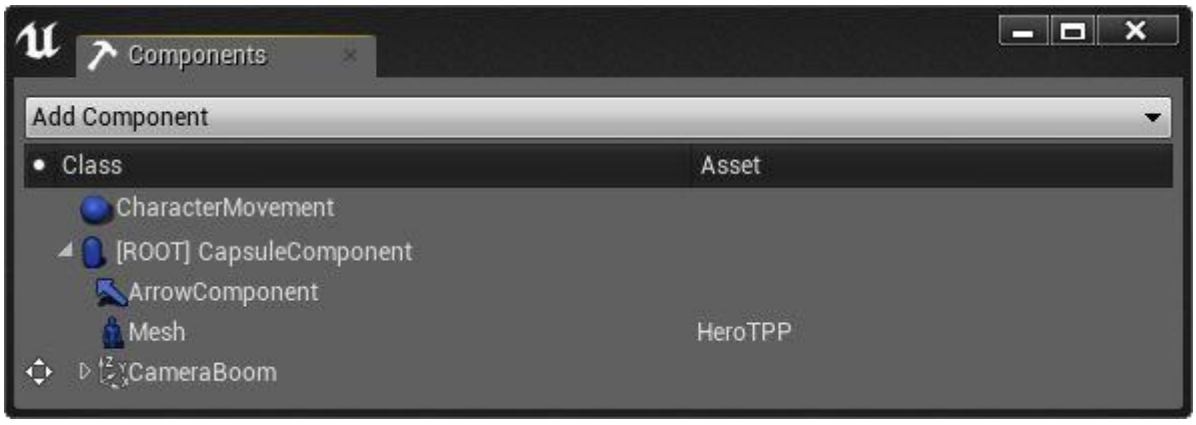


Рисунок 11 – Панель Components

2. Construction Script.

Construction Script представляет собой систему скриптов, использующихся исключительно в целях настойки Блупринта. Данный элемент не играет никакой роли на игровой процесс и служит лишь для изменения параметров или взаимодействия с блупринтом из редактора. Вся логика Construction Script’а активна только в самом редакторе. Данная система может

быть невероятно полезна при настойке блупринта, уже находящегося на уровне. Например, имеется блупринт, в компонентах которого есть источник освещения. Construction Script позволяет ещё на стадии проектирования уровня выставить цвет данного источника света в конкретной копии данного блупринта на сцене (Рисунок 12).

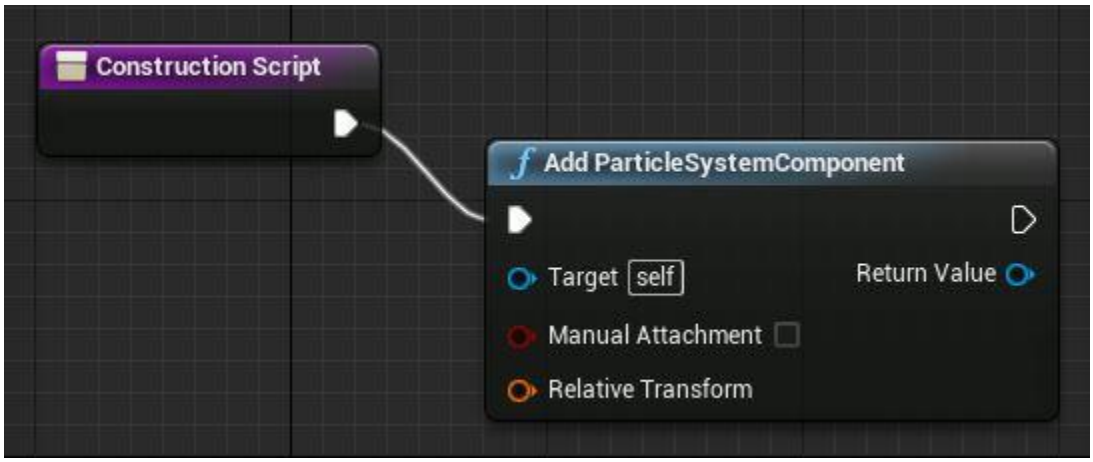


Рисунок 12 – Пример Construction Script

3. График (EventGraph)

EventGraph – Самый главный элемент редактора Блупринтов. Это место, где вы строите логику вашего Блупринта, используя события и функции, соединяя это все в последовательность действий.

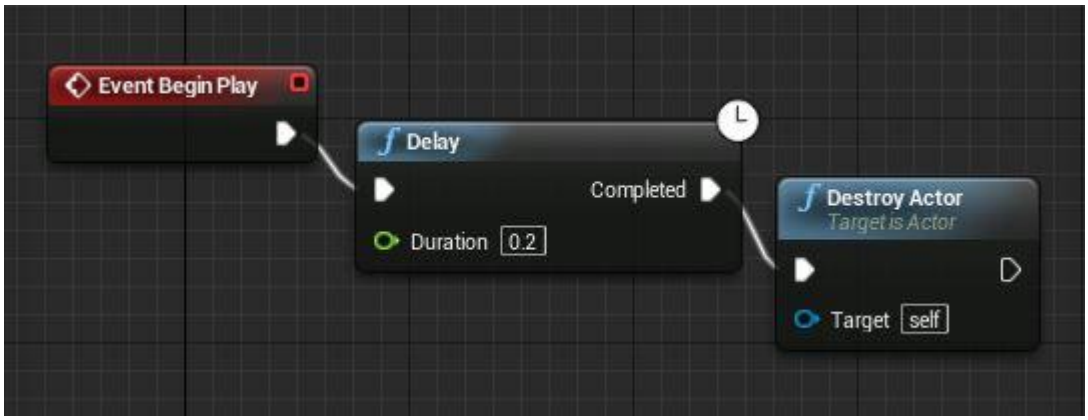


Рисунок 13 – Пример EventGraph

EventGraph (Рисунок 13) содержит график, который так же использует события и функции, для того что бы создавать какую-либо реакцию на поведение пользователя. Однако данный график, в основном, служит для

создания каких-либо действий непосредственно на текущей сцене, и используется для взаимодействия с конкретными объектами на уровне.

График отображает визуальную структуру логики определенного класса объектов вместе со всеми блоками и соединениями между ними. Так же он предоставляет возможность строить данную структуру, посредством вызова блоков функций и соединения их между собой.

4. Функции (Functions)

Функции могут принадлежать только конкретному Блупринту и используются подобно Макро функциям, однако различие в том, что в функции позволяют создавать внутри себя логические последовательности, состоящие из исполняемых блоков (Рисунок 14).

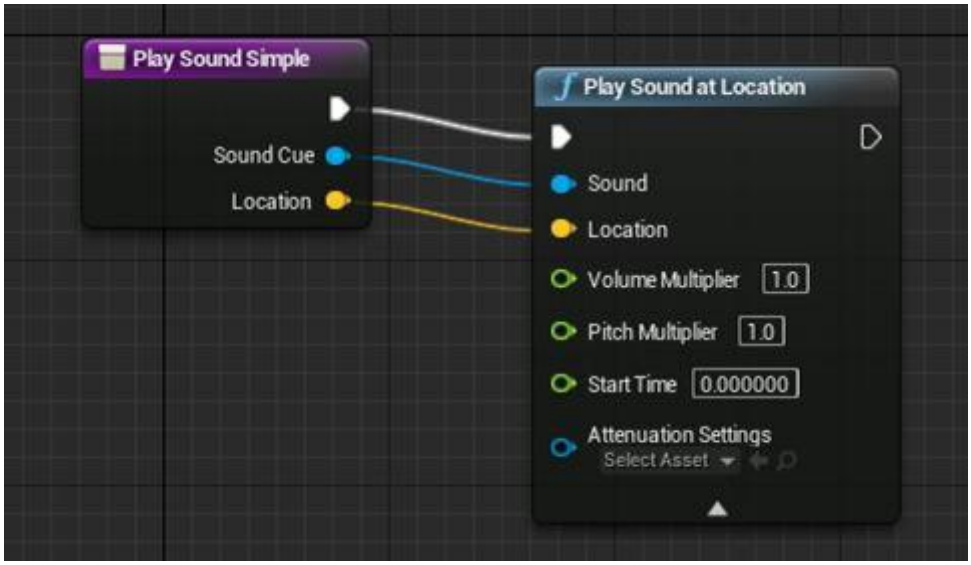


Рисунок 14 – Пример использования функции

5. Переменные(Variables)

Панель My Blueprint содержит список всех переменных, добавленных в блупринт, а также список компонентов, для отсылки к ним (Рисунок 15). Через данную панель можно так же добавить переменные.

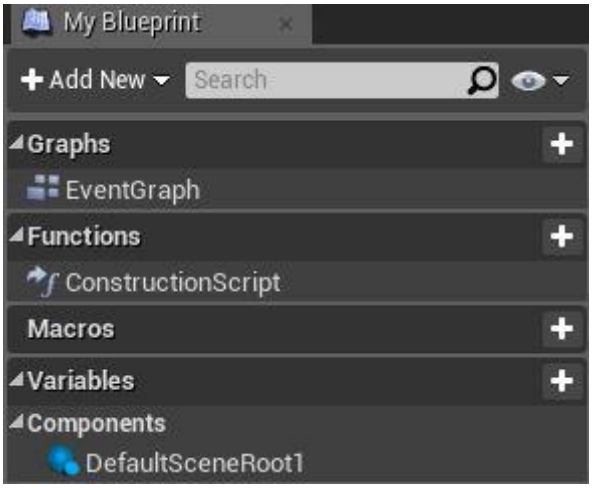


Рисунок 15 – Панель MyBlueprint

Переменные – специальный тип данных, который хранит в себе какое-либо значение. Переменные используются для назначения значений и последующего доступа в Блупринтах. Новые переменные могут быть добавлены нажатием на кнопку (Рисунок 16).

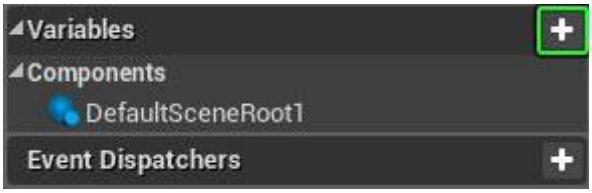


Рисунок 16 – Пример создания

После создания, можно переименовать переменную, а также изменить дополнительные параметры на панеле Details. (Рисунок 17)

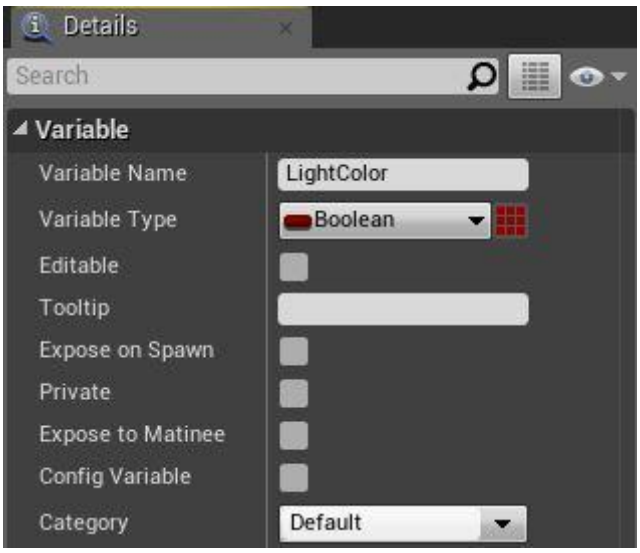
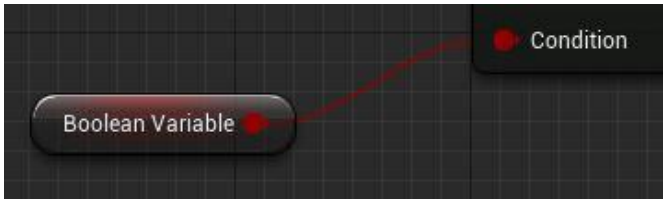
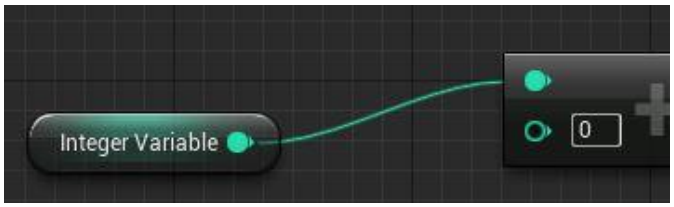
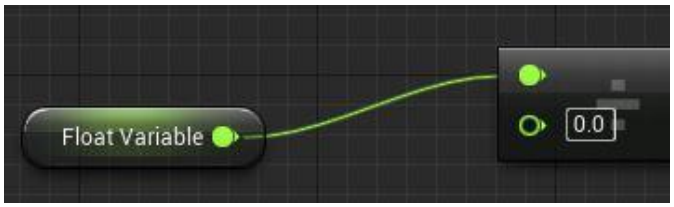
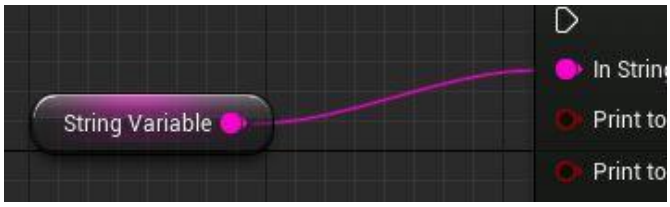
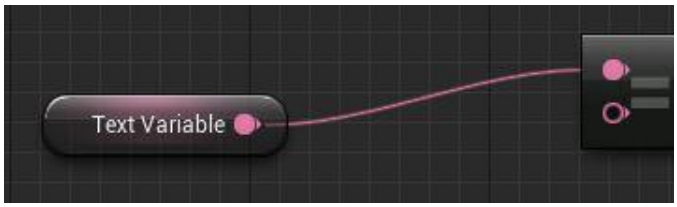
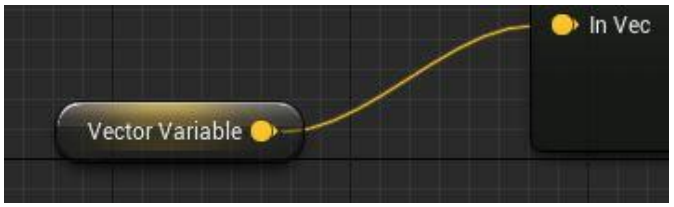
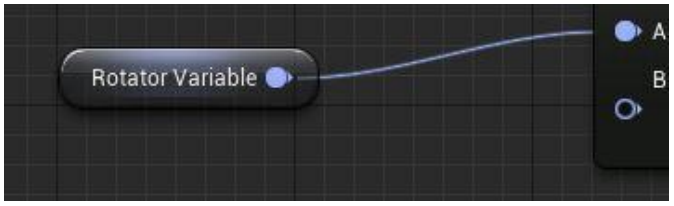
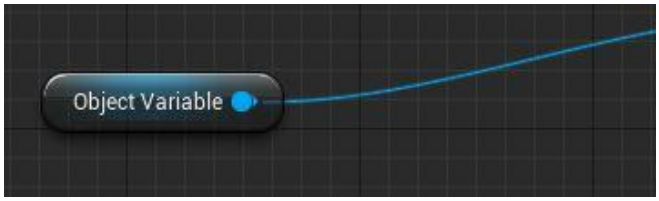


Рисунок 17 – Панель Details

Рассмотрим основные типы переменных, используемые в blueprint.

Таблица 2 – Основные типы переменных

Пример	Тип	Использование
	Boolean	Булевая переменная. Хранит в себе значения «Правда» или «Ложь»
	Integer	Переменная, которая хранит в себе целые числа.
	Float	Переменная, которая хранит в себе дробные числа.
	String	Строчная переменная для хранения небольших буквенных данных
	Text	Текстовая переменная для хранения локализированного текста.
	Vector	Переменная, которая сожержит векторные данные с 3мя элементами: X, Y и Z. Может так же подходить как RGB вектор.
	Rotator	Переменная с данными, которые определяют ориентацию в пространстве

	Transform	Переменная, включающая в себя векторное значение (тип Vector), поворот (тип Rotator) и размер(тип Vector)
	Object	Переменная, хранящая в себе отсылку к определенному объекту. Например источнике света, объекта на сцене.

Переменные так же могут быть изменены по ходу выполнения последовательности. Самый легкий способ установить переменную или получить её в графике, это перетащить переменную напрямую с панели My Blueprint на график. Откроется небольшое меню, предлагающее, хотите ли вы получить переменную или установить её (Рисунок 18).

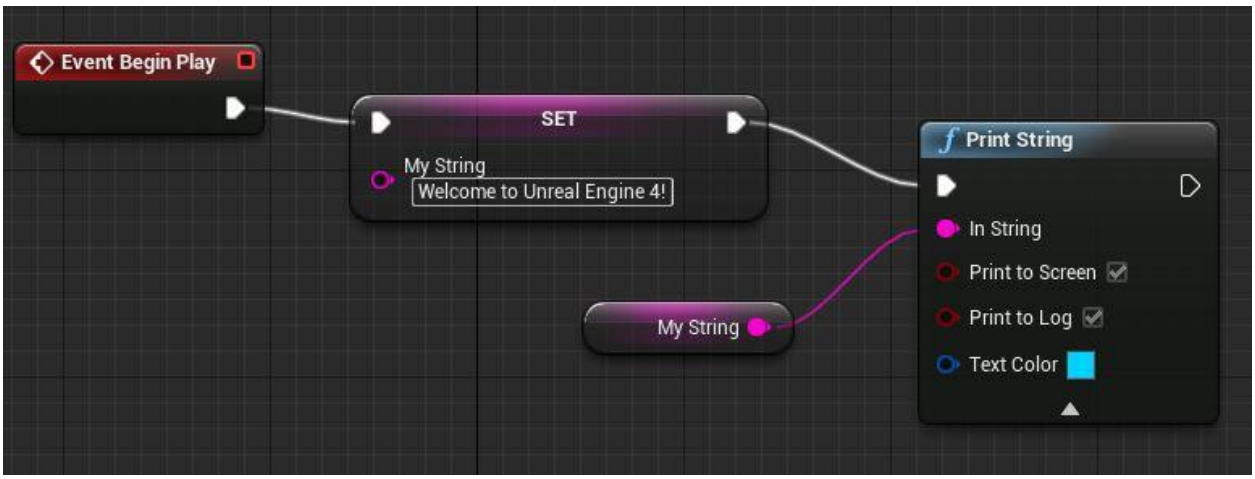


Рисунок 18 – Установка и получение значений переменных

Блок Get позволяет получить значение переменной в графике. После создания, переменная будет иметь выход для соединения с другими функциями или блоками. (Рисунок 19)

Рисунок 19 – Получение значения

Блок Set устанавливает переменную на входящее или введенное значение. Обратите внимание, что во вход можно присоединить лишь один контакт, когда из Get можно вытягивать соединения в множество блоков (Рисунок 20).

Рисунок 20 – Запись значения

2.2 Особенности организации обучения на основе создания компьютерной игры

В процессе обучения визуального программирования на основе создания динамических игр каждый ученик научится:

1. Создавать собственной компьютерной игры.
2. Получение удовольствия от процесса создания игры
3. Освоение технологии создания игр
4. Изучения языка программирования

Технология обучения программированию игр в подавляющем большинстве случаев рассчитана на практикующих программистов, преподается из расчета на знание специальных приемов и понимание сложных конструкций. Поэтому создание игры является одним из заключительных разделов методики обучения программированию, на основе освоения большей части конструкций языка программирования решается комплексная тренировочная задача по созданию игры, нет параллельности обучения программированию и создания игры.

Элементы объектно-ориентированного программирования считаются излишними в школьной программе, но их обучение важно для мета предметных умений и профессиональной ориентации. Наглядность объектов и визуального программирования в динамической игре станет существенным фактором для

усвоения сложных программных конструкций объектно-ориентированного программирования.

Была составлена логическая последовательность модулей для освоения учащимися алгоритмических и программных конструкция в процессе создания динамических игр. Первые два модуля посвящены анализу игры и работе с созданием объектов, т.е. программные конструкции в них не изучаются.

Таблица 3 – Программные конструкции относительно модулей

№ Модуля	Конструкции
2	Создание и модификация пользовательский объектов
3	Переменные. Операции присваивания. Функция. Наследование свойств. Полное условие. Комментирование программ. Операции сравнения. Цикл Со счетчиком. Массивы. Структуры с составным условием.
4	Обработка событий клавиатуры и мыши. Пользовательское событие. Переменные. Полное условие. Работа с логическими переменными.
5	Обработка событий объекта, глобальные, локальные переменные. Обращение к другим объектам.
6	Функция и процедура. Динамическое текстовое поле.

Была смоделирована работа учащегося по созданию компьютерной игры в данной последовательности.

Рассмотрим набравшуюся популярность на мобильных платформах жанр бесконечный раннер. Основа стандартная динамическая игра от третьего лица. Персонаж бесконечного раннера бежит сквозь локации игры, набирая очки и собирая монеты, преодолевая препятствия, прыгая и уклоняясь. Бег происходит автоматически и управлять игроку не придётся, а вот чтобы вовремя уклоняться и прыгать, вам понадобятся хорошая скорость реакции. Цель игры набрать максимальное возможное количество монеток. Если игрок столкнулся с препятствие конец игры и перезапуск уровня. В правом верхнем углу экрана ведется подсчет собранных монет.

Следует подчеркнуть, что раньше школьники уже видели десятки подобных игр, но рассматривали их как цельный, неделимый объект или продукт. Правила декомпозиции и композиции, данные им как средство дают,

возможность увидеть составные части игры, разложить их на простые составные части, достаточно легкие для дальнейшей реализации.

Самостоятельно составленная по итогам анализа модель служит отправной точкой, для своего описания собственной игры.

В разработанной методики преподавания, основной формой обучения являются практические занятия, проектная деятельность.

Самостоятельная работа учащихся подразделяется на три вида в зависимости от этапа обучения и степени освоения материала:

- работа по образцу: разработка конструкций, аналогичных конструкциям игры-прототипа, с индивидуально разработанными объектами и сюжетом;

- вариативная работа: разработка на основе конструкций игры-прототипа игр с оригинальным движением и взаимодействием объектов;

- творческая работа: разработка (на основе освоенных конструкций) новых этапов игры, игры другого типа, новых аспектов содержания игры (например, обучающих), не рассматриваемых в игре-прототипе.

В качестве средств обучения программированию на основе создания игр используются:

1) учебные пособия и материалы:

- раздаточный материал к занятиям; - материалы тестирований;

2) средства наглядности:

- проектор;

- презентации по теме занятий;

- маркерная доска.

3) средства осуществления практических действий:

- персональный компьютер;

- специальное программное обеспечение

- инфраструктура кабинета информатики общеобразовательной школы.

Обозначенные модули рассмотрим с точки зрения методики преподавания визуального программирования, проиллюстрировав результаты каждого из них.

Первое занятие включает знакомство с пакетом программного обеспечения Unreal Engine 4. Разъясняется структура и назначение программного обеспечения в современном мире и в индустрии развлечения. Подчеркивается актуальность.

Учащиеся делятся своими знаниях игровых платформ и играх, сделанных на технологиях Unreal Engine 4 (UE4).

Учитель приводит примеры проектов и игр в среде UE4, а так же игру прототип созданную учителем.

Учитель описывает и иллюстрирует примерами назначения элементов окна и панелей программы для дальнейших ссылок для создания игры.

1. Панель вкладок и Меню.

2. Панель инструментов ToolBar.

3. Перечень основных объектов (классов) Modes.

4. Встроенный файловый браузер, Контент Браузер.

5. панель, где вы просматривайте и редактируете ваш уровень напрямую. Вьюпорты (По умолчанию развернут режим перспективы).

6. Scene Outliner отображает список всех объектов на вашем уровне в древовидной систем.

7. Подробные настройки объекта или свойства Details.

Учащиеся записываются в тетради назначение панелей, работая синхронно с учителем, на произвольном проекте тренируются в использовании Viewporta и управления в нем, так же использования панелей программы; тренируются разворачивать, перетаскивать, скрывать панели по мере необходимости.

Модуль 1. Анализ

Учащиеся научаться: Анализировать, разбивать на структурные элементы, объекты; описывать взаимодействия объектов.

Применяемые методы: Логические (Анализа, сравнения, обощения, классификации); эвристические (эвристических вопросов, мозгового штурма, синектики, образной картины)

Продолжительность: 1 занятие

Результат: файл с описанием игры прототипы, файл с моделью игры-прототипа.

Учитель объясняет объектную модель на примере игровых приложений. Вводит понятия свойства, метода и события объектов. Приводит примеры и предлагает учащимся привести примеры и провести аналогии между реальным и виртуальными объектами, их свойствами.

Учитель предлагает проанализировать с точки зрения объектной модели несложную игру в жанре Endless Runner (Бесконечный бег) (Рисунок 21)

Рисунок 21 – Пример Endless Runner

Учащимся предлагается примерная форма модели и список условных объектов (классов) обязательных для игры данного типа. Результаты анализа (модели игры) излагается в устной или текстовой форме и затем структурируется учителем.

Пример анализа. Игра состоит из динамически случайно-генерируемой карты, предметов, взаимодействующих с игроком (различных препятствий, монеты, стены) и персонажа, который бежит постоянно вперед, а также счетчик

монеток (Рисунок 22). При сборе монеток меняется счетчик на количество собранных монет, при столкновении с препятствием происходит конец игры, сбрасывается счетчик собранных монет. Цель игры собрать максимальное количество монеток.

Далее требуется схематическое представление объектов на экране с указанием свойств объекта.

На следующем этапе учитель должен предложить опытный образец (прототип) игры, которая будет состоять из всех основных объектов, описанных в анализе (Рисунок 22).

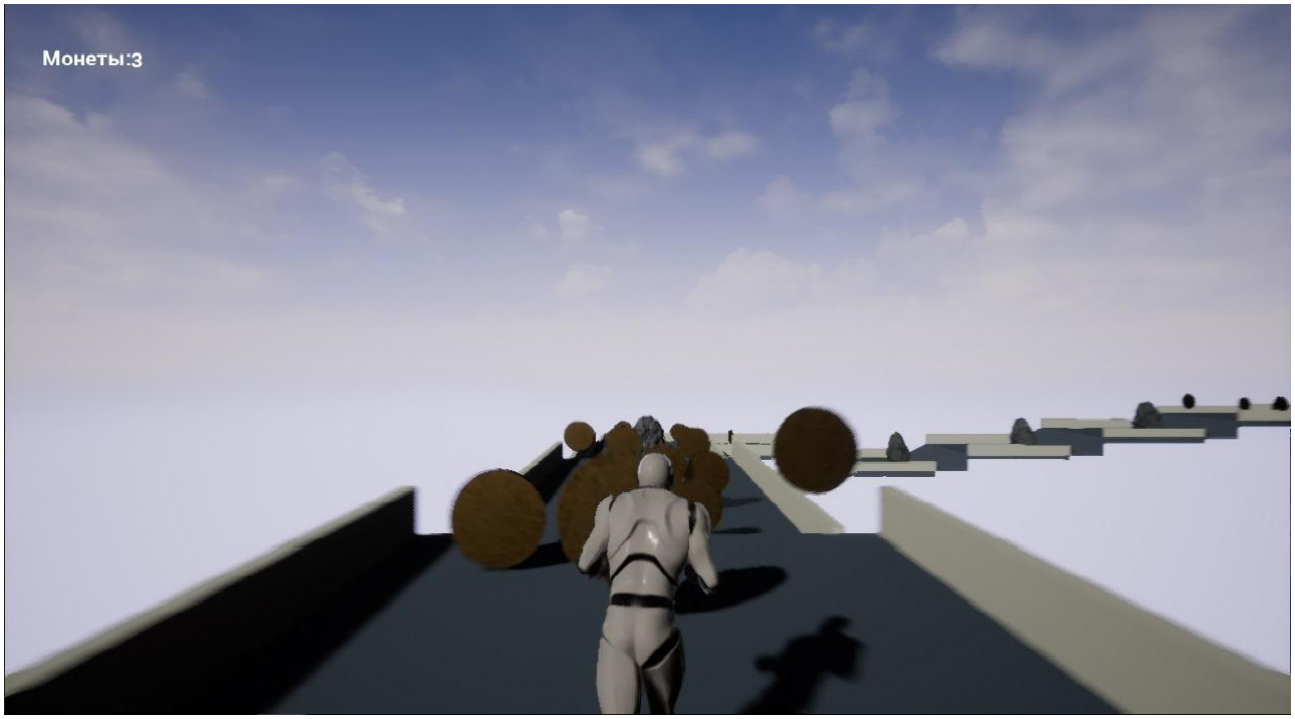


Рисунок 22 – Прототип игры

Основные объекты:

1. Случайно-генерируемая карта;
2. Персонаж;
3. Препятствия;
4. Монеты;
5. Счетчик монет.

Модуль 2. Подготовка объектов.

Учащиеся научаться: Ориентироваться в среде Unreal Engine 4. Создавать трехмерные объекты из примитивов. Манипуляции с объектами. Применять материалы к выбранным статическим объектам. Использовать концепцию объектно-ориентированного программирования – наследование классов.

Применяемые методы: логические (анализа, сравнения); эвристические (эвристических вопросов, образной картины); специфические (сквозной задачи, контроля освоения материала по промежуточным результатам, взаимообучения в группе).

Продолжительность: 1-2 занятия.

Результат: Трехмерные объекты, элементы из которых будет состоять игра.

Учитель знакомит учащихся с игровым движком Unreal Engine 4. А так же знакомство со встроенным языком визуального программирования Blueprint.

Редактор Blueprint состоит из таких элементов как:

1. Componets;
2. MyBlueprint;
3. ViewPort;
4. Eventgraph;
5. Toolbar;
6. Details.

На примере создания объектов, из которых будет состоять проект игры.

Это модуль нацелен на знакомство с редактором Blueprint в среде Unreal Engine 4. Учитель демонстрирует набор объектов для игры прототипа и дает учащимся задания создать аналогичный набор для своих игр. Учащиеся находят стандартные примитивы, объекты, находящиеся в библиотеке Unreal Engine 4.

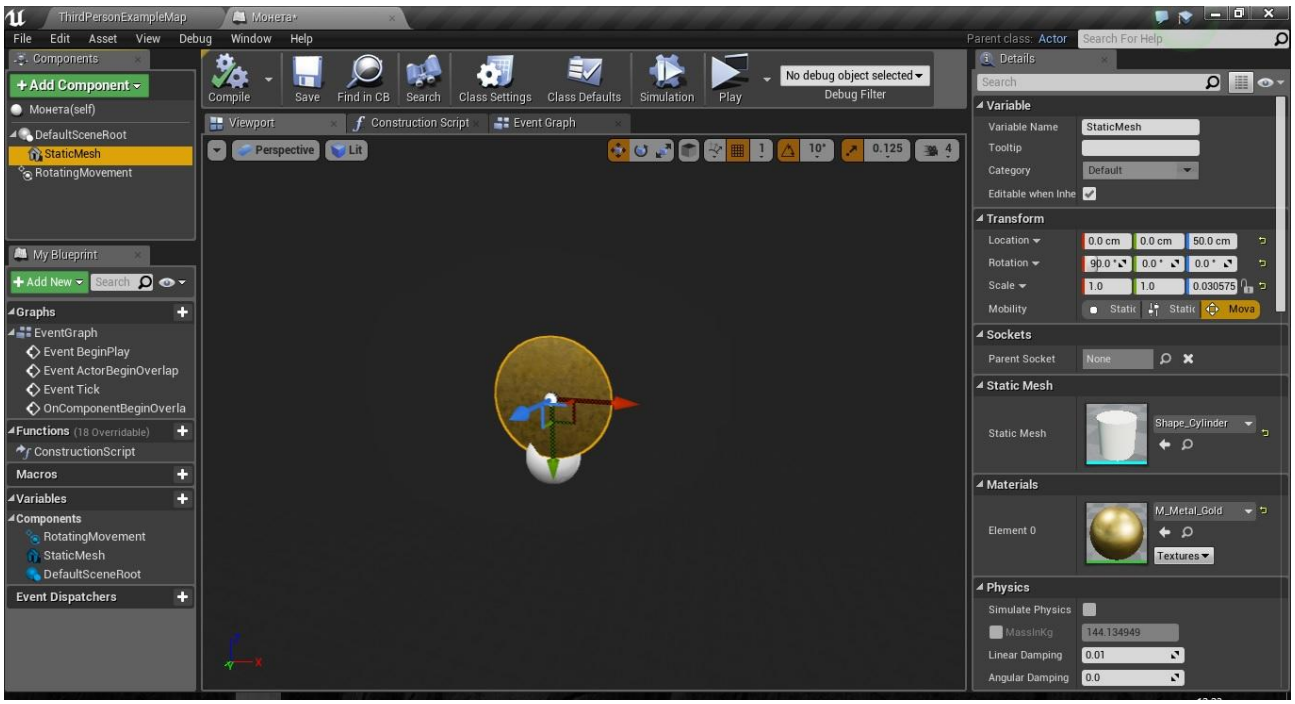


Рисунок 23 – Редактор Blueprint. Пример монеты

Учитель демонстрирует работу со стандартными объектами, изменяя их свойства (положение, размер, координаты, материал) (Рисунок 23)

Стандартная библиотека содержит в себе стандартный набор часто используемых материалов в играх.

Учащиеся самостоятельно создают заданные объекты и сохраняют их в Content Browser

Визуальные результаты модуля представлены на рисунке 24.

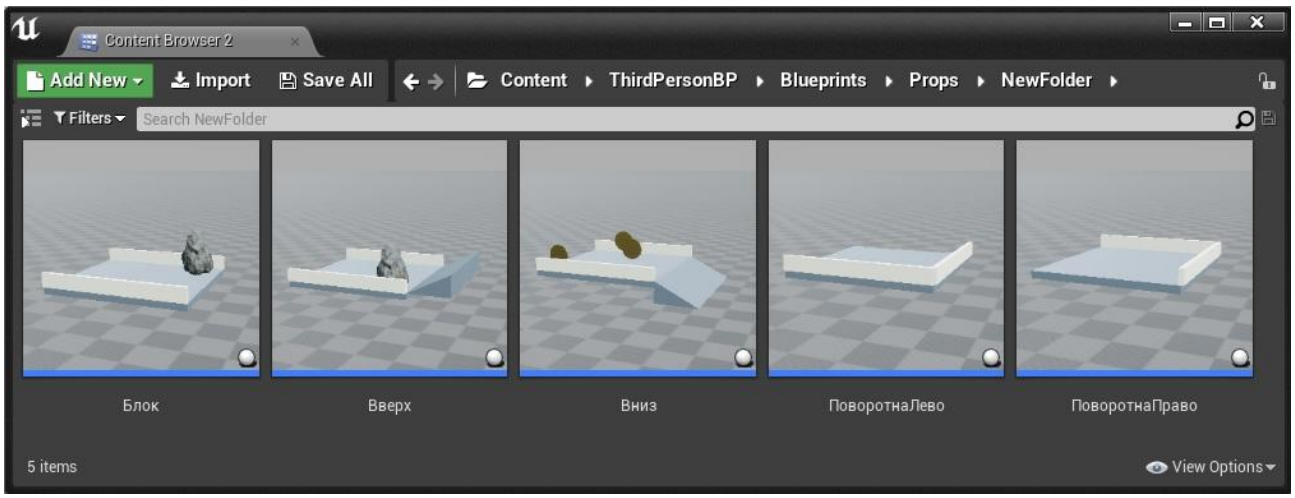


Рисунок 24 – Результаты модуля 2

Модуль 3. Случайно-генерируемая карта.

Учащиеся научаться: Основным понятиям объектно-ориентированного программирования (объект, свойства, методы, события); построению алгоритмов; генерации случайного числа; использовать цикл со счетчиком; пользоваться системой координат; работы с типами переменных; использованию полного условия.

Применяемые методы: логические (анализа, сравнения, обобщения, классификации, дедукции и индукции); эвристические (эвристических вопросов, мозгового штурма, синектики, образной картины); специфические (сквозной задачи, адаптации задачи для поэтапного освоения программных конструкций, контроля освоения материала по промежуточным результатам, взаимообучения в группе).

Продолжительность: 3 занятия

Результат: Проект с случайно генерируемой картой.

В этом модуле Учитель объясняет алгоритм создания случайно-генерируемой карты, а учащимся предстоит настроить карту проекта с ранее созданными объектами.

Объясняется алгоритм появления частей карты на сцене игры.

Разобьем на 3 этапы:

1. Использование основного блока;
2. Использование блока подъема-спуска (вверх-вниз);
3. Использование блока поворота.

На первом этапе объясняется принцип создания алгоритмов программирования с помощью визуального языка программирования Blueprint. Создают первый класс Blueprint Gamemod который отвечает за логику мира или игровые условия:

Учитель рассказывает о событийно-ориентированном программировании основой которого происходит различные события.

Событие – действие, которое может быть инициировано или пользователем, устройством, взаимодействием объектов или вызовом другой логики.

Учитель приводит пример логики с помощью события Event-begin-play, которая вызывается, когда запускает игра.

Обозначим у наших составных блоков начало и конец. За начало будет принимать центральную точку откуда начинает движение наш персонаж. А конечную центральную точку от куда персонаж выходит с блока.

Для построение карты нужно, в конец предыдущего блока установить начало нового блока. На этом будет построена логика, создания уровня.

Для того что бы расположить объект на сцене потребуется знать его координаты. Так как на сцене находится множество одинаковых объектов, потребуется получать множество раз координаты объекта и для того что бы оптимизировать алгоритм, логично использовать функцию. Функция — это поименованная часть программы, которая может вызываться из других частей программы столько раз, сколько необходимо. Функция, в отличие от процедуры, обязательно возвращает значение.

А так же создадим функцию добавления элемента карты на сцену.

Алгоритм построение карты на первом этапе будет, состоять из добавления на сцену последовательно 10 основных блоков карты. Так как будем часто использовать в логике мира, одно действие такое как добавление блока карты на сцену, необходимо создать функцию добавления блока.

После этого учащиеся запустят мир, и у них на сцене должен будет появится один основной блок. С помощью цикла со счетчиком вызовем данную функцию 10 раз. С помощью этого действия учащиеся познакомятся с алгоритмической конструкцией цикл со счетчиком.

В итоге первого этапа учащиеся должны получить на сцене, коридор (Рисунок 25).

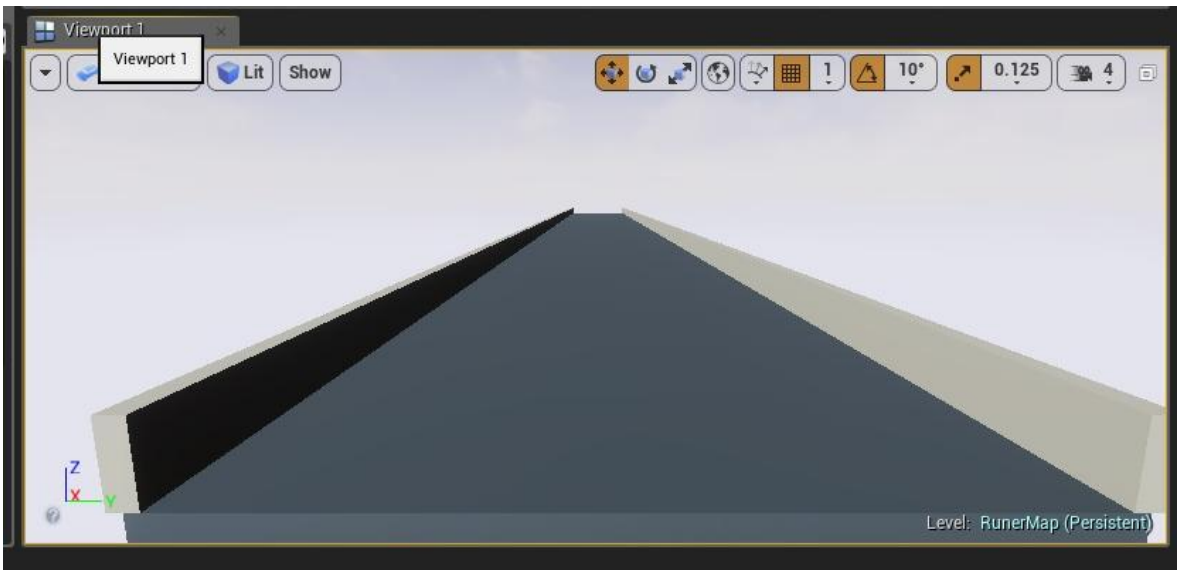


Рисунок 25 – Результат первого этапа модуля 3

На втором этапе учащиеся должны будут добавить разнообразие в линейный коридор, с помощью подъемов и спусков.

Для этого в этих объектах подъема и спуска потребуется переместить вектор отвечающий за конец нашего блока, потому что длинна, а, следовательно, и конец нашего объекта изменится. Для того что бы объект случайным образом создавал на карте подъем или спуск, создадим массив из наших объектов. То есть упорядоченные группировка объектов с присваиванием индекса. С помощь этого индекса будет осуществлен выбор случайного элемента из массива.

В итоге второго этапа должен получиться коридор с перепадом высот (Рисунок 26).

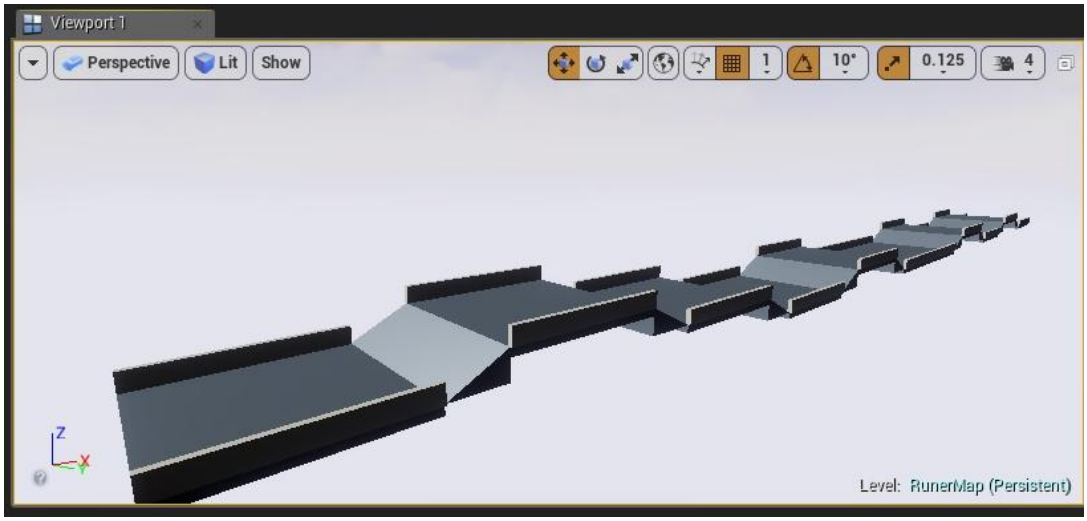


Рисунок 26 – Результат второго этапа модуля 3

На третьем этапе учащиеся добавляют к карте, блоки поворота направо и налево. Учитель объясняет, что если просто добавить к имеющему массиву блоки поворота, то случайным образом, траектория карты может зациклиться в друг друга, что способствует затруднению прохождения игры и приведет к окончанию игры. Для этого нужно добавить условие, что блоки поворота должны использоваться после построения 7 блоков прямого движения. Следовательно, нужно будет создать новую функцию для размещения на карте блоков поворота, а также добавить условие появление блоков на карте с использование полного условия, и счетчик блоков. После появления каждого прямого блока, переменная счетчика считает +1, если счетчик больше 7, то размещается случайным образом поворот на право или на лево, после чего счетчик сбрасывается.

В конечно итоге при запуске проекта учащиеся должны видеть на сцене 10 различных участков, а после 7 блоков устанавливается случайный поворот направо или налево (Рисунок 27).

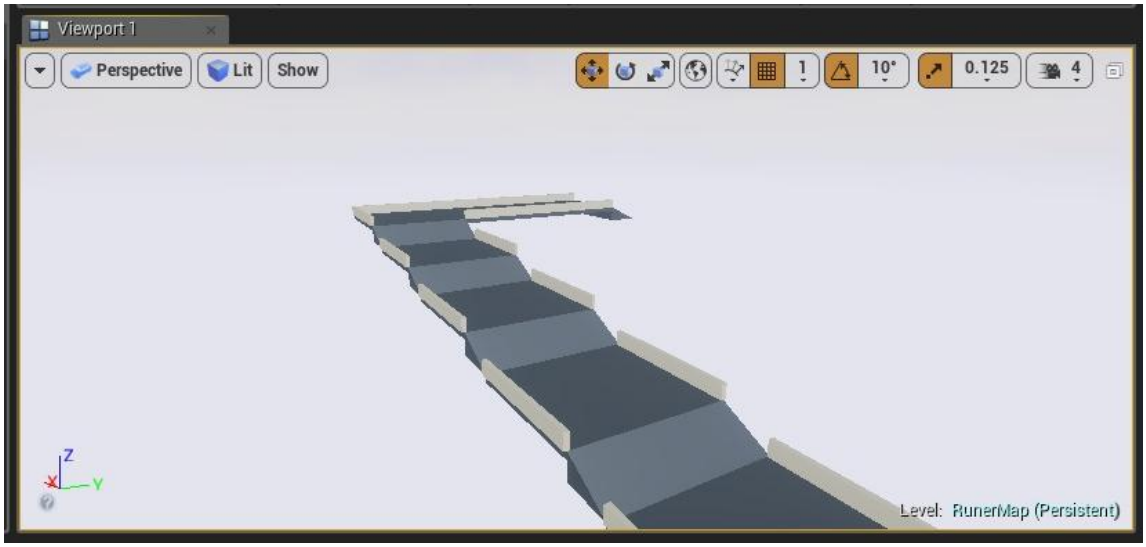


Рисунок 27 – Результат Модуля 3

Модуль 4

Учащиеся научаться: считывать события с клавиатуры; создавать события на пересечения персонажа и объекта; использовать неполное условие; применять пользовательские события;

Применяемые методы: логические (анализа, сравнения, обобщения, дедукции и индукции); эвристические (эвристических вопросов, образной картины); специфические (сквозной задачи, адаптации задачи для поэтапного освоения программных конструкций, контроля освоения материала по промежуточным результатам, взаимообучения в группе).

Продолжительность: 2 занятия

Результат: Готовый персонаж с полностью настроенными элементами управления.

Модуль посвящен созданию персонажа, которым будет управлять игрок (учащийся). Учитель ставит задачи исходя из требований к персонажу.

1. Игровой процесс происходит без управления камеры, камера статична относительно персонажа;
2. Постоянный бег вперед;
3. Управление поворота направо и налево, изменение траектории;
4. Поворот на 90 градусов.

Модель и анимация персонажа заранее приготовлены учителем, для того что бы не акцентировать внимание на этом, а сконцентрироваться на программирование самого персонажа.

Для создания управления, был взят стандартный персонаж уже с готовыми функциями движения. Учитель объясняет принцип работы управления персонажем.

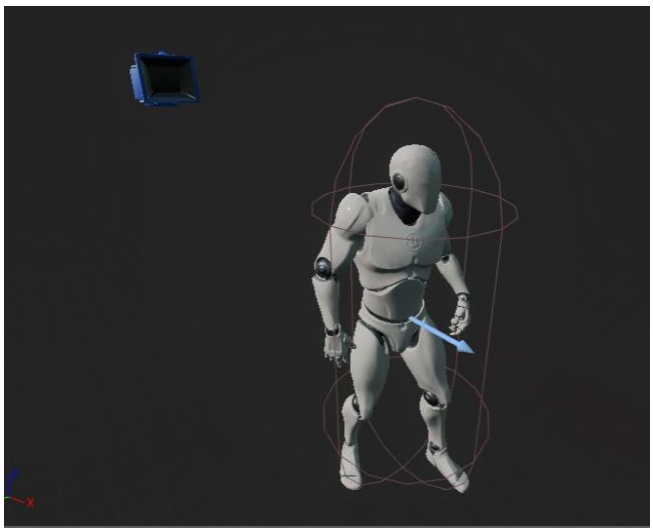


Рисунок 28 – Пример персонажа

1. Статичная камера относительно игрока.

Так как стандартный персонаж управляется динамической камерой зависящей от положения мышки, то потребуется удалить это событие.

Далее надо пройти во ViewPort и поставить координаты камеры чуть выше персонажа, для того что бы игрок видел путь перед персонажем, впереди стоящие препятствия и монеты.

2. Постоянный бег

Для реализации постоянного бега, учитель предлагает убрать событие управления бегом вперед - назад и поставить событие которое не зависит от клавиш и вызывает событие каждую секунду (EvenTick)

3. Управление поворота направо и налево.

Реализуется автоматически после добавления и использования одновременно двух событий на управление персонажем.

4. Поворот на 90 градусов.

Для реализации этой задачи учащиеся знакомятся с понятием BoxTriger, который создает событие, когда с ним пересекается другой объект.

Т.к. возможность поворота на 90 градусов нам нужна лишь в определенных участках, то потребуется поставить в эти участки BoxTriger, для активации резкого поворота. Алгоритм самого поворота состоит в следующем, если активна переменная поворота, то можно поворачивать, если нет, то

управление остается неизменным на события нажатия клавиши A и D что соответствуют повороту налево или направо.

После чего это событие добавляется в событие которое управляется движением прямо. Учащиеся переходят в блок элемента поворота и создают в нем box размерами с самого блока. После чего создают событие, которое обращается к нашему персонажу и записывает переменную поворота на истину, т.е. может поворачивать.

После всех этапов ученики размещают на сцене персонажа и у них должен появится уровень на котором бежит персонаж, откланяется от траектории влево и вправо, а также при резких поворотах поворачивается на 90 градусов.

Визуальные результаты модуля приведены на рисунке 29.

Рисунок 29 – Результат модуля 4

Модуль 5. Препятствия.

Учащиеся научаться: создавать события на пересечения персонажа и объекта (столкновение); создавать пользовательские события; использовать ConstructionScript для настройки класса; создавать и использовать функции;

Применяемые методы: логические (анализа, сравнения, обобщения, дедукции и индукции); эвристические (эвристических вопросов, образной картины); специфические (сквозной задачи, адаптации задачи для поэтапного освоения программных конструкций, контроля освоения материала по промежуточным результатам, взаимообучения в группе).

Продолжительность: 2

Результат: Уровень игры с добавлением препятствий.

Этот модуль посвящен обучению создания препятствий на пути персонажа. Учащиеся должны будут модернизировать объект препятствия, условно будет изображать камень. Для того что бы персонаж не проходил сквозь объект для него нужно создать сетку коллизии, от которой персонаж будет останавливаться и в итоге проигрывать.

Первым шагом создание событие окончание игры. В персонаже создаем пользовательское событие, которое называем «Конец игры». Создаем переменную типа Boolean(Истина-Ложь) переменную окончания игры, для того что бы потом модернизировать движение. Если событие сработало то, мы не можем управлять персонажем. Далее мы должны отключить управление и убрать персонажа с карты. После этого нам нужно вызвать «Конец игры» в событии столкновения с камнем, т.е. когда с нашим камнем сталкивает персонаж то мы обращаемся к персонажу и вызываем событие «Конец игры».

Следующим шагом учащиеся должны добавить в основной блок точки для появления камня. После этого создаем функцию, отвечающую за появление на участке пути наших препятствий Spawn Blocker. Для этого создаем еще одну функцию для преобразования наших координаты точек, на которых будет происходить появление препятствий и записываем их в массив. После этого мы берем из массива случайную координату точки и устанавливаем камень.

В итоге по окончанию модуля у учащихся должен получится уровень с препятствиями, и событием «Конец игры» (Рисунок 30).

Рисунок 30 – результаты модуля 5

Модуль 6. Подсчет предметов, интерфейс.

Учащиеся научаться: создавать интерфейс игры; создавать функции интерфейса; конвертировать переменные из одного типа в другой; использование программной конструкции выбора.

Применяемые методы: логические (анализа, сравнения, обобщения, дедукции и индукции); эвристические (эвристических вопросов, образной картины); специфические (сквозной задачи, адаптации задачи для поэтапного освоения программных конструкций, контроля освоения материала по промежуточным результатам, взаимообучения в группе).

Продолжительность: 1 занятие

Результат: Готовый уровень с препятствием. Сбором и подсчетом монет. Вывод подсчета монет на экран.

В шестом модуле ученики продолжают создавать предметы с которыми взаимодействует персонаж. Задача данного модуля стоит в создание случайно генерируемых монеток на поле. Персонаж должен подбирать монеты и подсчитывать количество монет.

1. Переменная отвечающего за подсчет монеты.
2. Функция подсчета.

3. Событие сбора монет.

4. Вывод на экран подсчета монет.

Ученики создают переменную собранных монет, а также функцию подсчета в самом персонаже. Она будет отвечать за количество монет, собранных персонажем на карте. Для того что бы изменять переменную создадим функцию подсчета монет. Которая изменяет переменную на +1.

Далее будет использоваться такое же событие, как и при столкновении с камнем, учащиеся сами составляют алгоритм. Учащие должны составить такой алгоритм: После столкновения персонажа и монеты, должна использоваться функция подсчета монеты у персонажа, в следствии чего собранная монета исчезнет, а переменная подсчета изменится на +1.

Следующим этапом учитель объясняет структуру класса Widget, который отвечает за вывод интерфейса игры на экран и напоминает структуру работы с объектно-ориентированным программированием, такие как Visual Basic, Visual Studio.

Структура Widget:

1. Palette – Палитра элементов меню

2. Hierarchy – Иерархия элементов присутствующие на игровом поле (холсте)

3. Details – Свойства выбранного на холсте элемента

4. Designer – Холст для настройки меню. Холстом является размеры экрана монитора.

Создаем horizontal box, и перетаскиваем на холсте где он будет располагаться. Далее добавляя в него два текста, в одном пишем «Монетки», во втором создаем функцию для конвертирования переменной подсчета монеты и вывода его на экран.

Осталось добавить класс widget на экран. Для этого нужно будет добавить widget «интерфейс» в логику мира.

В ходе освоения модуля шесть учащиеся добавят к проекту случайное появление на карте монет, взаимодействие персонажа с монетами, а также отображение на экране подсчета монет.

Рисунок 31 – Результат всех модулей

Результатом всех модулей является один уровень динамической игры (Рисунок 31). Который будет представлять из себя прототип игры, в котором будет случайно-генерируемая карта, с препятствием персонажа, сбора и подсчета монет. Помимо методической составляющей, каждый модуль дает видимую эволюцию демонстрируемого результата по сравнению с предыдущим и позволяет поддерживать мотивацию учащихся на результат на высоком уровне. Конкретное описание цели, для которой необходимо освоить текущую (подчас сложную) тему, улучшает понимание и позволяет учащимся охватить цельную картину процесса программирования. Целью каждого модуля является логичное развитие результата предыдущего.

Игра-прототип является итогом отбора содержания обучения программированию и разработки этапов создания компьютерной игры, она становится основой изучения программных конструкций и демонстрации планируемых промежуточных результатов при обучении программированию на основе создания игр.

ЗАКЛЮЧЕНИЕ

Цель настоящего исследования состоит в разработке методики обучения, учащихся на основе создания компьютерной игры с использованием визуального программирования.

Для достижения поставленной цели были выполнены следующие задачи:

1. Проведен анализ учебной и методической литературы по теме исследования. Выявлены и проанализированы подходы обучения программированию. Были рассмотрены парадигмы программирования и их факторы обучения.

2. Были проанализировали существующие программные средства для обучения визуальному программированию.

3. Рассмотрены аспекты обучения программированию на основе создания компьютерных игр; Проведена классификация компьютерных игр по жанрам, по цели процесса, по движению в реальном времени, по психологическим аспектам; Проанализированы программные средства разработки компьютерных игр для обучения визуальному программированию; выявлена наиболее приемлемая для обучения программированию среда разработки динамических компьютерных игр (Unreal engine 4).

4. Изучена техническая документация и литература, относящееся к среде разработки. Были выявлены особенности разработки в среде Unreal engine4, составлена описательная часть;

5. Были выявлены особенности организации обучения на основе создания компьютерной игры. Разработана технология создания игр учащимися, в том числе игра-прототип, модули с демонстрируемыми результатами, адаптированные типовые программные конструкции.

Таким образом, задачи решены в полном объеме, цель данной работы достигнута – разработана методика обучения учащихся на основе создания компьютерной игры с использованием визуального программирования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Kodu Classroom Kit for Educators [Электронный ресурс] // Режим доступа: http://fuse.microsoft.com /page/ kodu.asp

2. Арсак, Ж. Программирование игр и головоломок [Текст] / Ж. Арсак. - М.: Наука, 1990.

3. Бежанова, М.М. Практическое программирование. Структуры данных и алгоритмы : учебник [Текст] / М.М. Бежанова, Л.А. Москвина, И.В. Поттосин. - М. : Логос, 2001.

4. Бежанова, М.М. Современные понятия и методы программирования [Текст] / М. М. Бежанова, И. В. Поттосин. - М. : Науч. мир, 2000.

5. Воровщиков, С.Г. Учебно-познавательная компетентность старшеклассников: состав, структура, деятельностный компонент [Текст] / С.Г. Воровщиков. - М.: АПК и ППРО, 2006.

6. Выготский, Л. С. Мышление и речь: психика, сознание, бессознательное [Текст] / Л. С. Выготский. - Москва: Лабиринт, 2001.

7. Гейн, А.Г. Информатика и ИКТ (базовый и профильный уровни) [Текст]/ А.Г. Гейн [и др.].-М.: Просвещение, 2008.

8. Дьюи, Д. Психология и педагогика мышления [Текст] / Д. Дьюи ; пер. с англ. Н.М. Никольской. - М.: Совершенство, 1997.

9. Жемчужников, Д.Г. Создание компьютерных игр как средство обучения школьников программированию [Текст] / Д. Г. Жемчужников // Информатика и образование. - 2012. -№8. - С. 49-51

10. Зинченко, В.П. Исследование визуального мышления: вопросы психологии: учебник / В.П. Зинченко, В.М. Мунипов, В.М. Гордон. – Москва: Просвещение, 2000. – С. 3.

11. Информатика и ИКТ. Практикум по программированию. 10-11 класс. Базовый уровень [Текст] / под ред. Н.В. Макаровой. - СПб: Питерпресс, 2008.

12. Информатика, 10-11 класс. Учебник для учащихся средней школы [Текст] / под ред. Н.В. Макаровой. - СПб.: Питер, 2000.

13. Информатика, 9 класс. Учебник для учащихся средней школы [Текст] / под ред. Н.В. Макаровой. - СПб.: Питер, 2000.

14. Кнут, Д. Искусство программирования [Текст] / Д. Кнут; под общ. ред. Ю. В. Козаченко; пер. с англ. В. Т. Тертышного, И. В. Красикова. — М.: Вильяме, 2000.

15. Компьютерные игры без программирования : создание открыток, мультфильмов, фото-, видеоальбомов, мелодий для мобильных телефонов [Текст] / отв. ред. А. В. Сидорович. - СПб.: Лениздат, 2006

16. Лаптев, В. В. Методическая теория обучения информатике : Аспекты фундаментальной подготовки [Текст] /В.В. Лаптев, Н.И. Рыжова, М.В. Швецкий; Рос. гос. пед. ун-т им. А.И. Герцена. - СПб. : Изд-во СПетерб. ун-та, 2003.

17. Лапчик, М.П. Методика преподавания информатики: Учеб. пособие для студ. пед. вузов / И.Г. Семакин, Е.К. Хеннер; Под общей ред. М.П. Лапчика. – Москва: Издательский центр «Академия», 2001.

18. Любимский, Э. 3. Программирование [Текст] / Э.З. Любимский, В.В. Мартынюк, Н.П. Трифонов.- М.: Наука. Главная редакция физико-математической литературы, 1980.

19. Малев, В.В. Общая методика преподавания информатике: Учебное пособие / В.В. Малев. – Воронеж: ВГПУ, 2005.

20. Матросов, В.Л. Теоретические основы информатики : учеб. пособие для студентов вузов, обучающихся по специальности 030100 (050202) - информатика / В. Л. Матросов [и др.] ; Моск. пед. гос. ун-т. Москва : МГЛУ, 2005.

21. Методика преподавания информатики: учеб. пособие для студ. пед. вузов [Текст] / М.П. Лапчик, И.Г. Семакин, Е.К. Хеннер; под общей ред. М.П. Лапчика. - М.: Издательский центр «Академия», 2001

22. О программе Scratch [Электронный ресурс]//Режим доступа: http://info.scratch.mit.edu/ru/About_Scratch.

23. Окулов, СМ. Программирование в алгоритмах [Текст] / СМ. Окулов. - М.: БИНОМ. Лаборатория знаний. 2002.

24. Пейперт, С. Переворот в сознании: дети, компьютеры и плодотворные идеи [Текст] / С Пейперт : пер. с англ. / под ред. Беляевой А.В., ЛеонасаВ.В. - М.: Педагогика, 1989.

25. Резник Н.А. Технология визуального мышления: учебник // Н.А. Резник. – Санкт – Петербург: Свет, 2007. – С. 68 – 83.

26. Ткаченко, О.Н. Развитие визуального мышления в современной культуре / О.Н. Ткаченко // Омский научный вестник. – 2014. – № 4 (131). – С. 198-200.

27. Угринович, Н.Д. Информатика и информационные технологии. Учебник для 10-11 классов. Профильный уровень. [Текст] / Н.Д. Угринович. - 4-е изд. - М.: БИНОМ. Лаборатория Базовых Знаний, 2009.

28. Шабалина, О.А. Компьютерные игры как средство обучения разработчиков программного обеспечения [Текст] : монография / О. А. Шабалина, П. Н. Воробкалов, А. В. Катаев ; М-во образования и науки Российской Федерации, Волгоградский гос. технический ун-т. - Волгоград : ВолгГТУ,2011.

29. Эхерн, Л. Создание компьютерных игр без программирования [Текст] / Л. Эхерн. - М. : ДМК Пресс, 2003.

30. Официальная техническая документация Unreal Engine 4. [Электронный ресурс]. – Режим доступа: http://docs.unrealengine.com