Литература

1 Gosling, J. The Java language Specification. Java SE 7 Edition / J. Goslong, B. Joy. – Oracle America Inc., 2013. – 644 с.

2 Smith, G. Grails in Action / G. Smith. – MindView, Inc. teochew, 2014. – 576 с.

УДК 004.7

А. А. Лаптев, Н. Б. Осипенко

РАЗРАБОТКА ИГРОВОГО ПРИЛОЖЕНИЯ НА ОСНОВЕ UNITY3D

Статья рассказывает об основных этапах и принципах создания игрового приложения «Последний герой» с использованием технологии Unity3D, языка программирования JavaScript, программы Autodesk 3ds max, которое представляет собой 3D игру от первого лица для Windows. Описанная схема создания игровых приложений реализована в двух вариантах: упрощенный и продвинутый игровой интеллект. Упрощенный игровой интеллект умеет двигаться к цели и атаковать врага с определенной задержкой, продвинутый – дополнительно к перебчсленным возможностям еще преследовать цель, потерянную из зоны видимости, обходить и разрушать препятствия.

Благодаря прорыву в области технологий создания мобильных игр, появляется много начинающих разработчиков игр, желающих раскрыть свой потенциал. Создание уже готовых исходных модулей помогает оптимизировать работу по созданию игр, упрощать осуществление рутинных операций, освобождая место для творческих задач, превращая программирование из ремесла в искусство. Разработке такого модуля посвящена настоящая работа.

С целью создания искусственного интеллекта в Unity3D [1] было разработано приложение «Последний герой», которое представляет собой 3D игру от первого лица для Windows. Игра состоит из одного уровня, главного героя (игрока) и монстров (искусственный интеллект). Уровень состоит из одной улицы, зацикленной по кругу; улица – из дороги и тротуара. Вдоль дорог расположены здания, трава и вход в метро. На дороге есть машины, которые перекрывают путь главному герою. В самой дороге есть ямы и трещины, в которые игрок может провалиться. Когда игрок доходит до конца улицы он переходит в начало и продолжает снова проходить уровень. Так происходит, пока игрок не умрет.

Главный герой – это персонаж от первого лица. У игрока есть только одно оружие – автомат с бесконечным количеством патронов. В автомате 30 патронов, когда они заканчиваются, игрок начинает перезаряжаться, для перезарядки используется звуковой эффект. В центре экрана есть прицел, с помощью которого игрок может вести прицельный огонь по своим целям. На стрельбу из автомата наложены эффекты. Для управления игроком используется мышь. У главного героя есть показатель здоровья. Если оно заканчивается игрок умирает и переходит в главное меню игры. За убитого противника игрок вознаграждается монетами.

В игре есть один вид монстров – зомби, это основной противник игрока. Они генерируются автоматически и с увеличением дистанции игрока увеличиваются в количестве, а также растет скорость их передвижения. Зомби реагируют на игрока, подходят к нему и атакуют, отнимая у главного героя здоровье.

Для создания окружения использовалась программа Autodesk 3ds max. Эта программа использовалась для создания объектов, которые составляют основную часть уровня: модель плоскости, с дорогами и тротуарами, ямами и трещинами; модели всех зданий. Для создания плоскости с дорогами использовался примитив Box. Плоскость

создавалась, используя модульное строительство – то есть дорога создавалась по частям. Каждый отдельный модуль (отдельный объект): простая плоскость для земли или травы, дорога, тротуар – в последствие соединялись вместе и потом с помощью свойства объекта *Attach* соединялись и становились единым объектом. В результате моделирования дорога была разделена на несколько частей (объектов) для удобства её дальнейшего использования в Unity3D. В дороге были сделаны ямы и трещины. С помощью примитива *Line* был создан полигон, что-то вроде плоскости, который в последствие был выдавлен с помощью свойства объекта *Extrude*. В итоге получился объект трещина, которая имела нужную глубину. С помощью свойств *Bridge* и *Create* были созданы недостающие полигоны в объекте, чтобы сделать объект замкнутым. После нужно было вырезать трещину в дороге. Это осуществлялось с помощью *Compound Objects*. Используя тип *Boolean* на объекте и *Pick Operand* на другом объекте, из первого объекта вырезался второй объект там, где они пересекались. Таким образом, были сделаны ямы и трещины в дороге. В 3ds max было создано 5 типов зданий и вход в метро. Вход в метро состоит из двух модулей: крыша и сам вход. Вход представляет собой лестницу, опускающуюся вниз, крыша – аркообразная. Для создания входа и лестницы использовался примитив Box и потом с помощью *Extrude* были созданы лестница и плоскость, внутри ограненные стенами. Крыша создавалась с помощью примитива Line. С помощью его была создана арка. Она была выдавлена в длину и перекрыта сзади. После были наложены соответствующие текстуры на все объекты.

Здания создавались, используя модульное строительство. Были отдельные модули стены, стены с окном, стена для угла и стена с дверной прорезью. Так же были модули крыши, пола и дверь. После всё модули соединялись вместе по координатам, и получалось целое здание (рисунок 1).

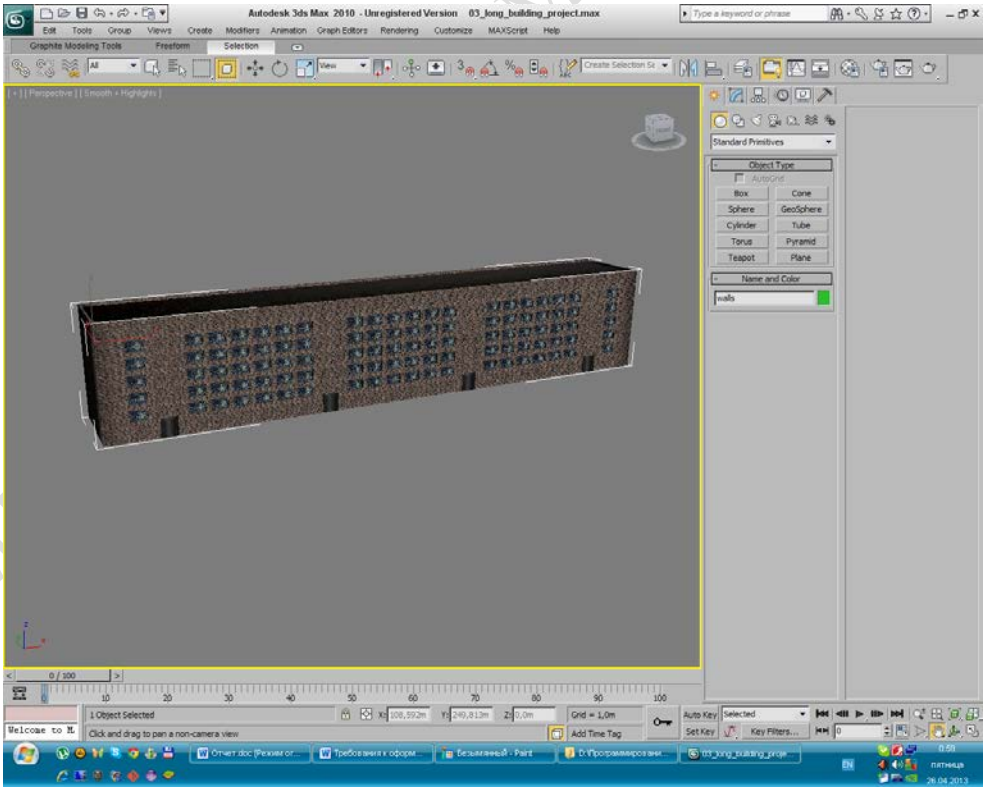

Рисунок 1 – Создание дома

Созданные модели экспортируются в Unity3D. Для начала был создан проект Unity3D. В данном приложении импортировались только пакеты *Standard Assets* и *Skybox*.

Первый нужен для создания персонажа, чтобы можно было побегать по уровню, *Skybox* – для создания неба. Если это первый вход в Unity3D, то программа предложит выбрать, куда сохранять проект и какие пакеты импортировать. Если же уже есть какой-то проект – то Unity3D его запустит и нужно будет нажать *File–>New Project* и появится аналогичное окно, в нем настраивается создание проекта. Можно выбрать путь проекта (*Project Location*) и импортировать нужные стандартные пакеты Unity3D (*Import the following package*s). Далее в настройках *Build Settings* был настроен проект. В *Scenes In Build* можно с помощью *Add Current* добавлять нужные сцены (уровни) в проект. В *Platform* можно выбирать под какую платформу будет компилироваться проект, в данном случае был сделан переход на платформу Windows с помощью *Switch Platform.* После этого было выбрано *PlayerSettings* и появилась вкладка для дальнейшей настройки приложения.

В приложении присутствуют две сцены: в первой сцене сделано меню входа в игру, во второй – сам уровень. Для создания уровня использовалась новая пустая сцена с помощью *File–>New Scene*, а затем импортированы все созданные и скачанные оптимизированные объекты в проект. После простым перетаскиванием и манипулированием объектами был выстроен уровень; наложены текстуры на дорогу, тротуары, землю; создано освещение *Directional Light* – дневной свет; с помощью пустых объектов созданы границы уровня. Создать пустой объект можно с помощью *GameObject–>Create Empty*. После на него была наложена физика, для того чтобы через этот объект нельзя было пройти с помощью *Component–>Phisics–>Box Collider*. С помощью растягивания данных объектов по высоте и длине уровень был ограничен, была создана зона, по которой персонаж может ходить, а вне её – нет. Эта зона в проекте Unity3D в виде зеленых перегородок (рисунок 2). Это пустые объекты с коллайдером.

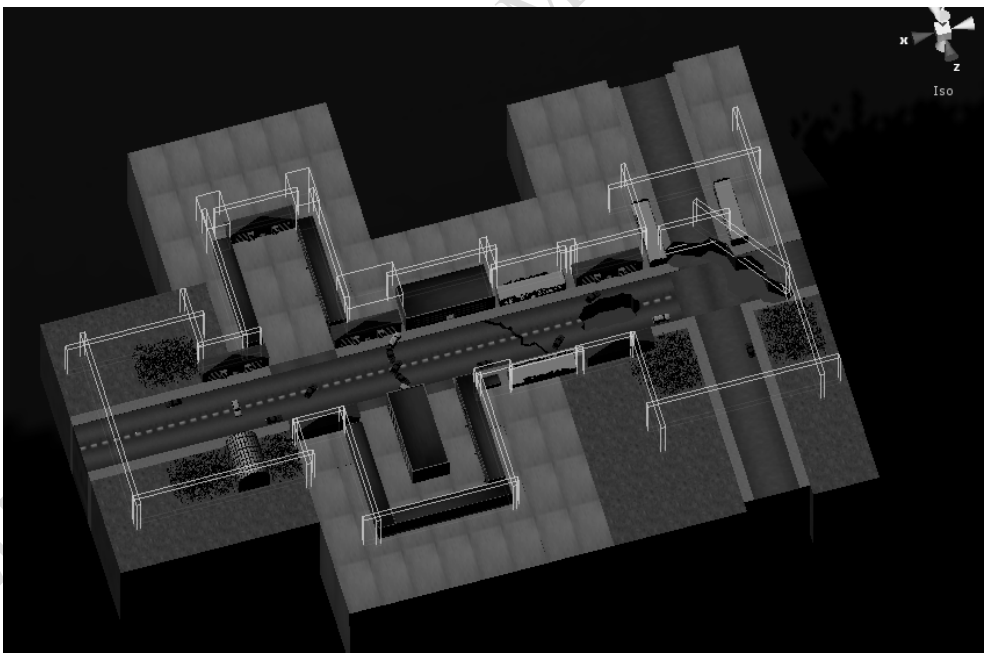

Рисунок 2 – Общий вид уровня с зонами

Для удобства было создано 2 вида меню: главное меню и меню паузы. Главное меню можно увидеть при старте игры. Это первая сцена. С помощью JavaScript скрипта [2] реализовано игровое меню и кнопки. Есть кнопки выхода из игры (Exit) и запуска игры (Start). Меню паузы реализовано с помощью с JavaScript. Оно позволяет поставить игру на паузу в нужный игроку момент. Есть кнопки: Продолжить (Continue), Настройки (Options), Выход в главное меню (Main Menu) и Выход из игры (Exit).

С целью зацикливания уровня были созданы 2 объекта: *Respawn* и *Finish*. Объекты *Respawn* и *Finish* являются триггерами. Для объекта *Finis* создан JavaScript «Teleport.js», который переносит игрока в позицию объекта *Respawn,* если он в него войдет.

Для создания игрока использовался prefab из стандартных пакетов Unity3D. Это *First Person Controls* из пакета *Standard Assets.* В нем уже реализовано перемещение персонажа и управление камерой. К персонажу был прикреплен объект-автомат (3d-модель, найденная в интернете), на него был прикреплен *AudioSource* с целью создания звукового эффекта стрельбы. Скрипт «Fire2.js» был написан для реализации стрельбы из автомата и запуска эффектов стрельбы. Для эффекта огня из дула автомата используется система частиц *Particle System*. Каждый раз, когда игрок стреляет, *AudioSource* воспроизводит звук выстрела, и создается клон объекта системы частиц для графического изображения выстрела. Для создания прицела был написан скрипт «Crosshair.js». С помощью функции *OnGUI()* в центре экрана отрисовывается прицел, в который идет выстрел.

Для показателя здоровья используется скрипт «HP_Hero.js». С помощью *OnGUI()* прорисовывается текстура, отображающая на экране (рисунок 3) показатели текущего и максимального количества здоровья: (*currentHP*) и (*maxHP*). Зеленая полоса – это показатель жизни героя. Она уменьшается, когда герой получает урон. В скрипте есть функция *ApplyDamage(damage: int)*, которая принимает повреждения от других объектов и следит за состоянием здоровья героя. Если оно меньше 0, то игрок умирает, и игра заканчивается переходом в главное меню. Также в нем реализовано воспроизведение звуковых эффектов, когда герой терпит урон и умирает.

Рисунок 3 – Текущее количество здоровья

У игрока есть счетчик заработанных монет – это объект *GuiText*. На него действуют 2 скрипта: «monetaSH.js» и «moneta.js». Первый отрисовывает текстуру монет, а его переменная – *schetchik* следит за количеством монет. Второй скрипт находится на монете, которая выпадает из зомби. Когда игрок подбирает монету, этот скрипт обращается к скрипту «monetaSH.js» и изменяет его переменную *schetchik*.

Есть один тип зомби. Это 3d-модель, которая была скачана из интернета. У неё есть анимация ходьбы. Для генерации зомби используется скрипт «GenerateZombie.js». Он находится на объекте, из которого должен появляться зомби. В скрипте отслеживается расстояние до игрока. Если игрок входит в зону реагирования зомби создается объект зомби. За количеством зомби на определенном расстоянии и их скоростью передвижения следит скрипт «maxZombie.js». Уровень разделен на 6 зон, в каждой из которых свой коэффициент. Когда игрок входит в новую зону, то скрипт увеличивает количество зомби на уровне и их скорость.

Описанная схема создания игровых приложений реализована в двух вариантах: упрощенный и продвинутый игровой интеллект. Упрощенный игровой интеллект умеет двигаться к цели и атаковать врага с определенной задержкой, продвинутый – двигаться к цели и атаковать врага с определенной задержкой, преследовать цель, потерянную из зоны видимости, обходить и разрушать препятствия.

Опыт, полученный от разработки этого приложения, дал возможность создать и опубликовать игровые приложения на GooglePlay на своём аккаунте FreedomGames: https://play.google.com/store/apps/details?id=com. freedomgames.memorygamemaster.

Литература

1 Unity learn documentation [Electronic resource]. – Mode of access : http://docs. unity3d.com/. – Date of access: 25.10.2015.

2 JavaScript [Electronic resource]. – Mode of access : https://ru.wikipedia.org/wiki/ JavaScript/. – Date of access : 20.10.2015.

УДК 372.853

В. В. Лелекова

ДОМАШНИЙ ЭКСПЕРИМЕНТ КАК СРЕДСТВО МОТИВАЦИИ УЧАЩИХСЯ К УГЛУБЛЕННОМУ ИЗУЧЕНИЮ ФИЗИКИ

Статья посвящена одной из форм внеклассной работы по физике. При формировании познавательных интересов учащихся особое место отводится такому эффективному педагогическому средству, как внеурочная работа по предмету. Целью организуемой работы является мотивирование учащихся к более глубокому изучению теоретических сведений и приобретению практических навыков по изучаемому материалу школьного курса физики. Организация внеурочной работы способствует повышению уровня заинтересованности школьников к изучаемому предмету, а также развитию их творческих способностей.

Уже в определении физики как науки заложено сочетание как теоретической, так и практической частей. Важно, чтобы в процессе обучения учитель смог как можно полнее продемонстрировать взаимосвязь этих частей. Ведь когда учащиеся почувствуют эту связь, то они смогут многим процессам, происходящим вокруг них, дать верное теоретическое обоснование. Это может являться подтверждением достаточно полного владения материалом [1, с. 15].

При проведении демонстрационного опыта в классе время ограничивается продолжительностью урока. При этом ведущую деятельность выполняют лишь несколько человек, которые следуют указаниям педагога. Остальные же, всего-навсего, наблюдают за проведением опыта. Зачастую после урока, к столу учителя подходит много учащихся, которые с любопытством рассматривают приборы, каждый из них пытается потрогать либо покрутить предоставленное оборудование. Это всё указывает на то, что многие из них сами хотят выполнять опыты, им это интересно.

Также существуют фронтальные лабораторные работы, в ходе которых учащиеся разделены на группы, часто это два или три человека, и им самим предлагается провести опыт, а после сделать выводы о проделанной работе. Для успешной реализации такой деятельности, требуется большое количество приборов. Однако очень часто в школьных кабинетах физики нет достаточного количества комплектов исправного оборудования для проведения таких работ. Кроме того возникают сложности организационного характера.

Перед каждым учителем стоит проблема стимулирования у обучающихся познавательных интересов, положительной настроенности к обучению и возбуждения внутреннего желания узнавать что-либо новое. Для того чтобы происходило такое стимулирование учителями разрабатываются всё новые и новые средства обучения.

Одним из таких средств является домашний эксперимент. Домашняя экспериментальная деятельность учащихся – это проведение опытов, наблюдений и лабораторных работ, выполняемых самостоятельно в домашних условиях, используя изготовленные ими самими приборы, с целью удовлетворения интереса и в соответствии с логикой