Здравствуйте, меня зовут Пименов Глеб Андреевич, я хочу представить Вам свой проект «Создание цифрового приложения».

Что это такое? Под цифровым приложением подразумевается игра. Для её создания был использован язык программирования – python.

Для чего это было сделано? Я хотел изучить python и попрактиковаться в нём. Кроме этого мой проект может быть полезен учащимся, которые интересуются информатикой, ведь они могут увидеть возможности для реализации своих навыков и идей, попробовать себя в чём-то новом.

Кроме того, ввиду стремительного развития сферы информационных технологий и постоянно снижающихся требований для входа в эту отрасль, начать обучаться и пробовать себя могут даже ученики средних или старших классов. В современном мире существует потребность в профессионалах среднего и высокого уровня в сфере информационных технологий. В связи с этим, появляется возможность для популяризации и раскрытия возможностей учеников через это направление, таким образом, будет сделан вклад в развитие и образование будущих поколений.

Одному из членов жюри я предлагаю испытать моё приложение.

Позвольте мне раскрыть Вам технологическую структуры данной работы. Как уже говорилось, я использовал python, сегодня очень популярный язык программирования, выигрывающий за счёт простоты синтаксиса и широкой области применения. Для создания игры была использована библиотека pygame – она позволяет взаимодействовать с графическим интерфейсом. Библиотека выбиралась исходя из того, что конечным продуктом будет игра. Изображения, аудио сопровождение и звуковые эффекты были созданы не мной, я взял их с открытых источников, преимущественно с opengameart.com, не для коммерческого использования, все авторские права соблюдены и представлены на странице с источниками.

При написании приложения была использована парадигма объектно-ориентированного программирования. Это очень популярный способ написания приложений из-за хорошей читаемости кода и лёгкости его улучшения или отладки в дальнейшем. Программа представляет собой совокупность объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

Кроме того, я старался соблюдать основные принципы программирования, следования таким принципам заставляет подумать в начале, но зато значительно упрощает процесс разработки в дальнейшем. Например, вся программа состоит из большого количества файлов – модулей, которые подключаются по необходимости. Обычно один файл – один класс, некоторые небольшие классы, наследующие друг друга и использующеюся обычно вместе, объединены в один модуль, как пример, модуль анимаций. Такой подход позволяет быстро находить интересующий метод или класс и даёт возможность тестировать модули по отдельности, не затрагивая структуру всей системы.

После выбора темы и технологий для проекта я задумался над концепцией игры. Выбор был сделан на платформере с видом сбоку, в котором нужно защищаться от всякой нежити. За этим было создано несколько пробных версий игры и решено, как примерно будет создаваться приложение. Было спроектировано три варианта карты классов. Вы можете видеть их на экране. В финальной версии была использована первая система. Основное различие между первой и второй – это то, что во второй класс существ не является наследником, и использует собственные методы и атрибуты. Всё дело в том, что у персонажей есть две статичные картинки – правая и левая, а, например, у камня она только одна, общий же родитель имел только один атрибут с картинкой, что не позволяло его использовать для существ. Но было решено, что лучше будет наследовать все классы от одного родителя, а данную проблема устранена с помощью добавления атрибута исключительно для существ, содержащего ссылку на вторую картинку. Третья же карта классов предполагала разделение между самой игрой и приложением. Дело в том, что изначально планировался только один режим игры – арена, режим, в котором игроку необходимо уничтожить всех врагов во всех волнах. В этом случае сама карта и игрок загружались сразу с загрузкой приложения, а главное и побочное меню просто накладывались сверху. Из-за того, что карта и режим был один, то и загрузить все необходимые ресурсы было достаточно один раз, а потом просто убирать монстров и обнулять монетки и здоровье. Но в конце разработки, было принято решение добавить второй режим – режим сюжетной кампании, который состоит из нескольких уровней. Если бы он был запланирован в начале, то была бы использована третья карта классов, так как загружалось бы сначала главное меню, а потом уже выбранный режим. Но в заключительной версии сначала запускается режим арены, а если в меню выбирается другой, то и запускается другой (по сути – создаётся новая карта), а если выбирается арена, то игра снимается с паузы и меню закрывается. В результате, такое различие никоим образом не отражается на внешнем виде игры.

Кроме карты классов была создана ментальная карта приложения, в ходе разработки она была дополнена, но основа была заложена. На ментальной карте отображены использованные механизмы и технологии. Как Вы можете видеть, при запуске приложение сначала инициализируется – запускаются циклы: цикл игровых вычислений, который просчитывает столкновения, гравитацию и подобные вещи, цикл отрисовки объектов, который перерисовывает изображение, чтобы казалось, что объекты на экране совершают действия. Каждый цикл срабатывает 60 раз в секунду, такая частота приемлема для человеческого глаза. Кроме циклов есть и процедура, срабатывающая при определённых событиях, например, так сделано управление – при нажатии кнопки срабатывает событие, событие запускает процедуру, она определяет, что за событие произошло, и после этого запускается другая процедура, к примеру, ходьбы. Дальше загружаются ресурсы, подключаются библиотеки и так далее. После этого всё включается, и мы видим главное меню. Далее можно нажать на одну из кнопок, это приведёт к определённым последствиям. Некоторые события можно повторить, например, сначала запустить режим арены, потом вызвать побочное меню, а потом перейти оттуда в главное и снова вызвать режим арены.

Таким образом, в конечной версии игры были реализованы режим арены и режим сюжетной кампании, выбор полноэкранного и оконного режимов, изменение сложности игры, возможность выбрать язык (русский или английский), небольшие титры, ну и выход из этой самой игры. Стандартный набор для игры, можно было найти движок, который бы включал подобные функции, хотите сказать Вы, но хочу напомнить, что целью было создать всё самому и показать процесс разработки, к тому же, с каждым заготовленным шаблоном сужаются возможности для реализации и фантазии.

Вы можете видеть, что в процессе работы программы используется технология – JavaScript Object Notation (JSON в сокращении). Она позволяет создавать файлы с расширением .JSON, которые позволяют хранить не только текст, но и другие типы данных, например числовые переменные или словари. Используется это у меня для записи всех текстов на двух языках и данных об анимациях. В других версиях игры была реализована возможность сохранения, которая записывала весь прогресс игрока в такой файл. Это гораздо удобнее чем использовать обычный текстовый документ. Но несмотря на удобство такого подхода, для составления карты игры мне хватило и простого текстового документа, так как кроме одиночных символов в ней ничего не используется. Карту игры составить или изменить очень просто – она состоит из блоков, которые можно ставить друг на друга. В файле представлена проекция настоящей карты, но каждый объект заменён знаком. У каждого объекта, будь то трава или декоративное дерево, есть определённый номер. В результате, программа считывает файл, видит цифру и устанавливает на это место соответствующий цифре объект.

Для сюжетной кампании я создал четыре уровня, то есть четыре карты, последнюю я использовал также для арены. Перед их созданием, разумеется, сначала был придуман и написан сценарий, после этого было начато проектирование условных карт уровней и в итоге их получилось четыре. На каждой карте изображён уровень в полном объёме, а также начальное положение игрока и положения, в которых срабатывают определённые монологи.

Для написания кода использовался eclipse, это интегрированная среда разработки, в ней легко производить отладку приложения, можно быстро переключаться между папками с ресурсами, автоматически запускать программу без ввода полного пути и так далее. Все продемонстрированные карты, создавались в приложении draw.io, в нём удобно проектировать подобные схемы или таблицы.

Можно также предположить, что некоторые могли никогда не видеть, как выглядит код программы и среда разработки. На этом примере Вы можете видеть класс гравитации. Здесь описано название класса, дальше идёт конструктор, который задаёт атрибуты, в данном случае один – ускорение свободного падения, после него мы видим процедуру, принимающую объекты, которые будут подвергаться гравитации, а потом добавляющая к вертикальной скорости этих объектов ускорение. Вы также можете заметить, что ускорение свободного падения в реальной жизни работает немного не так, но для игры нам достаточно такого.

Хотелось бы подвести итог. Я спроектировал и написал приложение и хочу сказать, что проектирование – это очень важная часть, гораздо проще писать интуитивно понятное и логичное приложение чем придумывать всё по ходу. Кроме того, важным было соблюдение определённых стандартов, например, один метод – одно действие. Не всегда сразу получается написать идеальный класс или метод, поэтому приходиться заниматься так называемым рефакторингом (refactoring), то есть перепроектированием кода, преобразованием некоторых алгоритмов, с целью упрощения понимания работы программы. Принимая во внимание всё вышесказанное уместно сделать вывод, что при должном уровне заинтересованности игру может создать даже ученик 10 класса.