Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

«Игра «Танчики» под Windows»

по дисциплине

«Системное программное обеспечение вычислительных машин»

Выполнил:
студент гр. 450502
Лучина Сергей Ильич

Руководитель:
Лавникевич Д. А.

Минск, 2016

ЛИСТ ЗАДАНИЯ

Название темы:

«Игра «Танчики» под Windows»

Основные требования к программе:

- Наличие текстур
- Рандомная генерация уровня
- Искусственный интеллект
- Система HP
- Наличие геймплея, знакомого по оригинальной игре
- Режим игры – «выживание»

Рекомендуемые системные требования:

- Операционная система Windows 10
- 16ГБ ОЗУ
- Разрешение экрана 1920x1080 или выше
- Несколько гигабайт свободного места на SSD
- Клавиатура

Для сборки из исходных кодов потребуются:

Microsoft Visual Studio (желательно версия 2015 с update 2 или новее)

Наличие следующих библиотек:

- Glut или freeglut
- glew

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С каждым годом возрастает количество различных переизданий старых игр. Их цель - вернуть интерес ностальгирующих геймеров, а также заинтересовать новых игроков. Переиздания могут адаптировать игру под новые архитектуры ПК и консолей, расширяют список поддерживаемых платформ, улучшать геймплей и визуальную часть игры. Также большим спросом пользуются различные эмуляторы старых консолей.

Мною была выбрана цель сделать ремейк популярной игры «Battle City» с некоторыми геймплейными модификациями используя современные технологии. Для разработки была выбрана OS Windows, язык программирования C++ и кроссплатформенная графическая библиотека OpenGL.

Разработанная в ходе курсового проектирования игра имеет простое управление, соревновательный характер, который понравится современным геймерам по всей планете. Графика имеет приятный 8-ми битный стиль. В игре были использованы спрайты из NES версии игры. Игра имеет очень низкие системные требования, что позволяет запустить ее даже на очень старых ПК. Современные игроки легко освоят новый геймплей.

Следующие версии программы могут быть направлены на исправление существующих ошибок, добавление нового функционала, улучшение кроссплатформенности.

Приоритетные задачи:

- Переход на современнейший игровой движок Unreal Engine (4.12.1 (версия от 7.6.2016) или новее)
- Переход от 2D в 3D
- Улучшение логики AI
- Переход на формат отображения 16:9
- Улучшение управления
- Добавление графических эффектов
- Добавление текстур сверхвысокого разрешения
- Добавление звуковых эффектов

Наиболее интересными улучшениями игры, которые могут быть реализованы после приоритетных задач и позволят выделить данный проект среди аналогов, являются:

- Добавление поддержки геймпадов
- Добавление многопользовательской игры на одном компьютере
- Сетевая игра
- Поддержка 4К разрешения
- Введение микротранзакций
- Редактор уровней

Опыт в разработке данного программного продукта помог мне пополнить знания по разработке игр, разработке приложений с использованием библиотек низкоуровневой работы с графикой, а также в целом улучшить понимание некоторых алгоритмов игрового программирования.

ОБЗОР ИСТОЧНИКОВ

1.1 *Обзор программных аналогов*

Ниже приведено описание типичного для данной игры геймплея.

Battle City (с англ. – *Город битв*) - компьютерная игра для игровых приставок Famicom и Game Boy. В России и странах СНГ выпускалась на пиратских картриджах как в оригинальном виде, так и в модификации **Tank 1990**, и известна под неофициальным названием «**Танчики**». Её предшественником была аркадная игра Tank Battalion, выпущенная фирмой Namco в 1980 году.

Полигон действий виден сверху. Игрок должен, управляя своим танком, уничтожить все вражеские танки на уровне, которые постепенно появляются вверху игрового поля. Враги пытаются уничтожить базу игрока (внизу игрового поля в виде орла) и его танк. На каждом уровне нужно уничтожить около двадцати единиц бронетехники противника разных видов. Если противник сможет разрушить базу или лишит игрока всех жизней — игра окончена.

В игре имеется четыре типа вражеских танков, которые различаются скоростью и прочностью:

- обычный танк (100 очков);
- бронетранспортёр, который отличается повышенной скоростью хода (200 очков);
- скорострельный танк (300 очков);
- тяжёлый танк (броневик), уничтожить который можно четырьмя попаданиями (танк меняет цвет в зависимости от оставшейся прочности) (400 очков).

Существует несколько бонусов:

- *Танк* («жизнь»). Прибавляет игроку одну жизнь.
- *Пятиконечная звезда (медаль)*. Улучшает танк игрока.
- *Ручная граната* («бомба»). Взрывает танки противника на карте, за их уничтожение очки не начисляются — только 500 очков за взятие бонуса как такового.
- *Часы*. На некоторое время останавливает врагов и их стрельбу.
- *Штыковая лопата*. Временно делает кирпичную стену штаба бетонной, что защищает его от вражеских снарядов. Если кирпичное

ограждение вокруг штаба было уничтожено, после прекращения действия лопаты (бетонная стена) вокруг штаба восстанавливается кирпичная стена.

- *Каска*. Временно делает танк игрока неуязвимым.

Рис. 1 – оригинальная версия игры, 21 уровень

1.2 *Основные технологии, использованные при разработке, и их краткий обзор:*

C++ - компилируемый статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции.

OpenGL (**Open G**raphics **L**ibrary) - спецификация, определяющая платформонезависимый (независимый от языка программирования) программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику. Последняя версия на данный момент – 4.5.

Существует ряд библиотек, созданных поверх или в дополнение к OpenGL. Например, библиотека GLU, являющаяся практически стандартным дополнением OpenGL и всегда её сопровождающая, построена поверх последней, то есть использует её функции для реализации своих возможностей. Другие библиотеки, как, например, GLUT и SDL, созданы для реализации возможностей, недоступных в OpenGL. К таким возможностям относятся создание интерфейса пользователя (окна, кнопки, меню и др.), настройка контекста рисования (область рисования, использующаяся OpenGL), обработка сообщений от устройств ввода-вывода (клавиатура, мышь и др.), а также работа с файлами. Обычно, каждый оконный менеджер имеет собственную библиотеку-расширение для реализации вышеописанных возможностей, например, WGL в Windows или GLX в X Window System, однако библиотеки GLUT и SDL являются кросс-платформенными, что облегчает перенос написанных приложений на другие платформы.

Библиотеки GLEW (The OpenGL Extension Wrangler Library) и GLEE (The OpenGL Easy Extension library) созданы для облегчения работы с расширениями и различными версиями OpenGL. Это особенно актуально для программистов в Windows, так как заголовочные и библиотечные файлы, поставляемые с Visual Studio, находятся на уровне версии OpenGL 1.1.

OpenGL имеет только набор геометрических примитивов (точки, линии, многоугольники) из которых создаются все трёхмерные объекты. Порой подобный уровень детализации не всегда удобен при создании сцен. Поэтому поверх OpenGL были созданы более высокоуровневые библиотеки, такие как Open Inventor и VTK. Данные библиотеки позволяют оперировать более сложными трёхмерными объектами, что облегчает и ускоряет создание трёхмерной сцены.

GLM (OpenGL Mathematics) — вспомогательная библиотека, предоставляющая программистам на C++ классы и функции для выполнения математических операций. Библиотека может использоваться при создании 3D-программ с использованием OpenGL. Одной из характеристик GLM является то, что реализация основана на спецификации GLSL. Исходный код GLM использует лицензию MIT.

OpenGL Utility Toolkit (GLUT) - библиотека утилит для приложений под OpenGL, которая в основном отвечает за системный уровень операций ввода-вывода при работе с операционной системой. Из функций можно привести следующие: создание окна, управление окном, мониторинг за вводом с клавиатуры и событий мыши. Она также включает функции для рисования

ряда геометрических примитивов: куб, сфера, чайник. GLUT даже включает возможность создания несложных всплывающих меню.

OpenGL Extension Wrangler Library (**GLEW**) - кроссплатформенная библиотека на C/C++, которая упрощает запрос и загрузку расширений OpenGL. GLEW обеспечивает эффективные run-time механизмы для определения того, какие OpenGL расширения поддерживаются на целевой платформе. Все расширения OpenGL размещаются в одном заголовочном файле, который автоматически генерируется из официального списка расширений.

freeglut — открытая альтернатива OpenGL Utility Toolkit (GLUT). GLUT (и, следовательно, freeglut) позволяет пользователю создавать окна, предоставляющие контекст OpenGL на широком спектре платформ, и управлять ими, а также взаимодействовать
с мышью, клавиатурой и джойстиком. freeglut предназначена для полной замены GLUT, и имеет очень немного отличий от неё.

С того времени, как оригинальный GLUT прекратил развитие, freeglut начал развиваться с целью улучшения предоставляемого инструментария. Он выпущен под лицензией X Consortium.

1.3 *Функциональные требования к проекту:*

Игра является ремейком игры «Battle City».

Список требований к проекту:

- Наличие искусственного интеллекта (врагов)
- Режим игры – «выживание». Подразумевает бесконечную игру на максимальное количество уничтоженных танков противника.
- Наличие рандомно генерируемой карты
- Система HP
- Наличие текстур
- Логика перемещения из оригинальной игры

1.4 API Vulkan и его преимущества над API OpenGL

Vulkan - кроссплатформенный API для 2D и 3D графики, впервые представленный Khronos Group в рамках конференции GDC 2015.

Vulkan API изначально был известен как «новое поколение OpenGL» или просто «glNext», но после анонса компания отказалась от этих названий в пользу названия Vulkan. Как и OpenGL, Vulkan позволяет с высокой производительностью отображать в реальном времени различные приложения с 3D графикой, такие как игры или интерактивные книги, на всех платформах, а также обеспечивает более высокую производительность и меньшую нагрузку на процессор, аналогично Direct3D 12 и Mantle. Vulkan основан на технологиях AMD в Mantle.

Летом 2014 года Khronos Group начала проект по созданию следующего поколения графического API. В 2014 на SIGGRAPH, проект был публично анонсирован с призывом к участию.

Согласно организации США по патентам и товарным знакам, фирменный знак «Vulkan» был зарегистрирован 19 февраля 2015 года.

Vulkan был официально назван и анонсирован на Game Developers Conference 2015, хотя спекуляции и слухи вокруг нового API существовали заранее. Один из вариантов названий был «glNext».

3 марта 2015 года, Valve анонсировала Source 2, игровой движок с поддержкой графического API Vulkan.

В начале 2015 года, LunarG (финансируется Valve) разработан и представлен драйвер Linux для Intel, который позволил Vulkan иметь совместимость с интегрированной графической системой HD 4000 серии, которая, несмотря на открытый драйвер Mesa, не полностью совместима с OpenGL 4.0. Существует еще возможность поддержки Sandy Bridge, так как он поддерживает Direct3D11.

10 августа 2015, Google объявила о будущей версии Android с поддержкой Vulkan.

18 декабря 2015 года, Khronos Group объявила о том, что спецификация версии Vulkan 1.0 практически завершена и будет выпущена, когда будут доступны совместимые драйверы.

16 февраля 2016 года, выпущена публичная спецификация версии Vulkan 1.0 и экспериментальные драйверы для видеокарт Nvidia и AMD

Vulkan предназначен для обеспечения различных преимуществ по сравнению с другими API, а также его духовного предшественника OpenGL. Вулкан предлагает более низкие накладные расходы, более непосредственный контроль над GPU, и с меньшей нагрузкой на CPU. Vulkan имеет предполагаемые преимущества:

- OpenGL использует язык высокого уровня для написания шейдеров GLSL. Это заставляет каждого производителя OpenGL драйвера реализовать свой собственный компилятор для GLSL, который выполняется во время выполнения приложения, чтобы перевести шейдерные программы в исполняемый код для целевой платформы. Vulkan вместо этого обеспечивает промежуточный двоичный формат под названием SPIR-V (Standard Portable Intermediate Representation), аналогичный двоичному формату в который компилируются HLSL шейдеры на платформе DirectX. Это снимает бремя с поставщиков драйверов, позволяя производить компиляцию шейдеров на этапе разработки. Также позволяет разработчикам приложений писать шейдеры на других языках кроме GLSL.
- Кроссплатформенный API поддерживается на мобильных устройствах и высокопроизводительных видеокартах.
- Улучшенная поддержка современных систем, использующих многопоточность.
- Снижение нагрузки на ЦП в ситуациях, когда процессор является узким местом, что позволяет достичь более высокой пропускной способности для GPU-вычислений и визуализации.

Начальная спецификация утверждает, что Vulkan будет работать на оборудовании, которое в настоящее время поддерживает OpenGL ES 3.1 или OpenGL 4.x и выше. В качестве поддержки Vulkan потребует новых графических драйверов, но это не обязательно означает, что все существующие устройства, которые поддерживают OpenGL ES 3.1 или OpenGL 4.X будут иметь доступные драйверы с поддержкой Vulkan.

СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ

2.1 *Описание структуры классов*

В курсовом проекте содержатся 8 классов:

- Painter
- Map
- Entity
- Player
- Shell
- Enemy
- Spawner
- DetectHit

Краткое описание классов:

- Entity - родительский класс для классов Enemy, Player и Shell. Содержит обобщённые методы, параметры, характерные для всех вышеперечисленных классов
- Player - наследник класса Entity. Отвечает за перемещение главного игрока в заданном направлении, проверка столкновений с объектами игрового окружения.
- Класс Enemy - еще один наследник класса Entity. Отвечает за перемещение врагов в заданном направлении. Содержит алгоритмы случайного поворота в случайную сторону по истечении заданного времени (направление может совпадать с предыдущим направлением движения). Отвечает за таймер стрельбы врагов, проверка столкновений с объектами игрового окружения.
- Класс Shell - ещё один наследник класса Entity. Отвечает за перемещение снарядов в заданном направлении. Проверка на столкновение с объектами игрового окружения, разрушение кирпичных блоков.
- Класс Painter - класс, отвечающий за отрисовку объектов игрового мира, наложение текстур по заданным координатам. Также данный класс производит загрузку изображения и его парсинг.
- Класс Map - производит чтение карты с файла. Из полученных данных генерирует совершенно случайную карту по заданному алгоритму. Производит передачу необходимых данных для рендера карты.
- Класс Spawner - класс, отвечающий за случайное появление ботов по истечению заданного промежутка времени.

- Класс DetectHit - класс, отвечающий за логику столкновения снарядов с игроками/врагами.

Графически отношения между классами программы представлены на структурной схеме (приложение А) и диаграмме классов (приложение Б).

ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В данной главе будут представлены строение программы и её основные методы.

Одной из задач при написании приложения было создание методов, каждый из которых выполняет строго определенную функцию. Для этой цели было найдено решение в виде инкапсуляции данных в отдельные классы и их методы. Так же было использовано наследование для унификации кода. Это помогло существенно увеличить наглядность исходных кодов, уменьшить время, потраченное на отладку программы.

3.1 *Описание методов классов программы*

В функции main происходит инициализация окна, настройка осей. Функции timerRender и timerUpdateLogic отвечают соответственно за вызов функции отрисовывающую сцену и за вызов геймплейных функций, таких как проверка на столкновение снарядов, вызов функции update на каждый объект. Функции keyboard и keyboard2 служат соответственно для изменений направления движения главного игрока. Функция display производит отрисовку игровой сцены и смену буферов, хранящих данные об изображении.

Объекты некоторых классов необходимо создавать сразу при входе в игру.

3.1.1 *Описание работы класса Entity*

Обобщённый класс, от него наследуются классы Player, Enemy, Shell. Хранит обобщённые методы, переменные, необходимые для корректной работы алгоритмов в наследуемых классах

3.1.2 *Описание работы класса Player*

Наследуется от класса Entity.

Основные методы класса Player – update и Collision. В конструкторе задаются начальные параметры, такие как стандартное направление выстрела, количество очков прочности, начальные координаты (место спауна), количество заработанных очков и, так называемое, кодовое имя персонажа.

В методе update происходит перемещение по осям X и Y, и вызов функции Collision после каждого перемещения. Увеличивает счётчик, отвечающий за время от последнего выстрела, а также проверяет количество очков прочности и текущее направление движения

В методе Collision проверяется столкновение танка со статичными объектами игрового окружения, такими как кирпичные и бетонные блоки

3.1.3 *Описание работы класса Enemy*

Наследуется от класса Entity.

Основные методы класса Enemy – update и Collision. В конструкторе задаются начальные параметры, такие как стандартное направление движения танка, количество очков прочности, скорость движения, начальные координаты (место спауна), время перезарядки орудия и, так называемое, кодовое имя врага.

В методе update происходит перемещение по осям X и Y, вызов функции Collision после каждого смещения, наращивание счётчик для поворота в случайном направлении, в случае достижения определённой границы запускается рандом от 0 до 3, каждая цифра отвечает за новое направление движения, однако новое направление движения может совпасть с предыдущим. Также происходит проверка целостности боевой машины, наращивание счётчика, отвечающего за стрельбу. В случае достижения определённой границы происходит выстрел и снаряд заносится в конец списка снарядов.

В методе Collision проверяется столкновение танка со статичными объектами игрового окружения, такими как кирпичные и бетонные блоки. В случае столкновения также увеличивается счётчик, отвечающий за случайное изменение направления, для уменьшения эффекта «холостого хода в стену».

3.1.4 *Описание работы класса Shell*

Наследуется от класса Entity

Основные методы класса Shell – update и Collision. В конструктор класса передаются начальные координаты левого верхнего угла, имя стрелявшего и направление полёта снаряда.

В методе update происходит перемещение по осям X, Y, вызов функции Collision после каждого смещения,

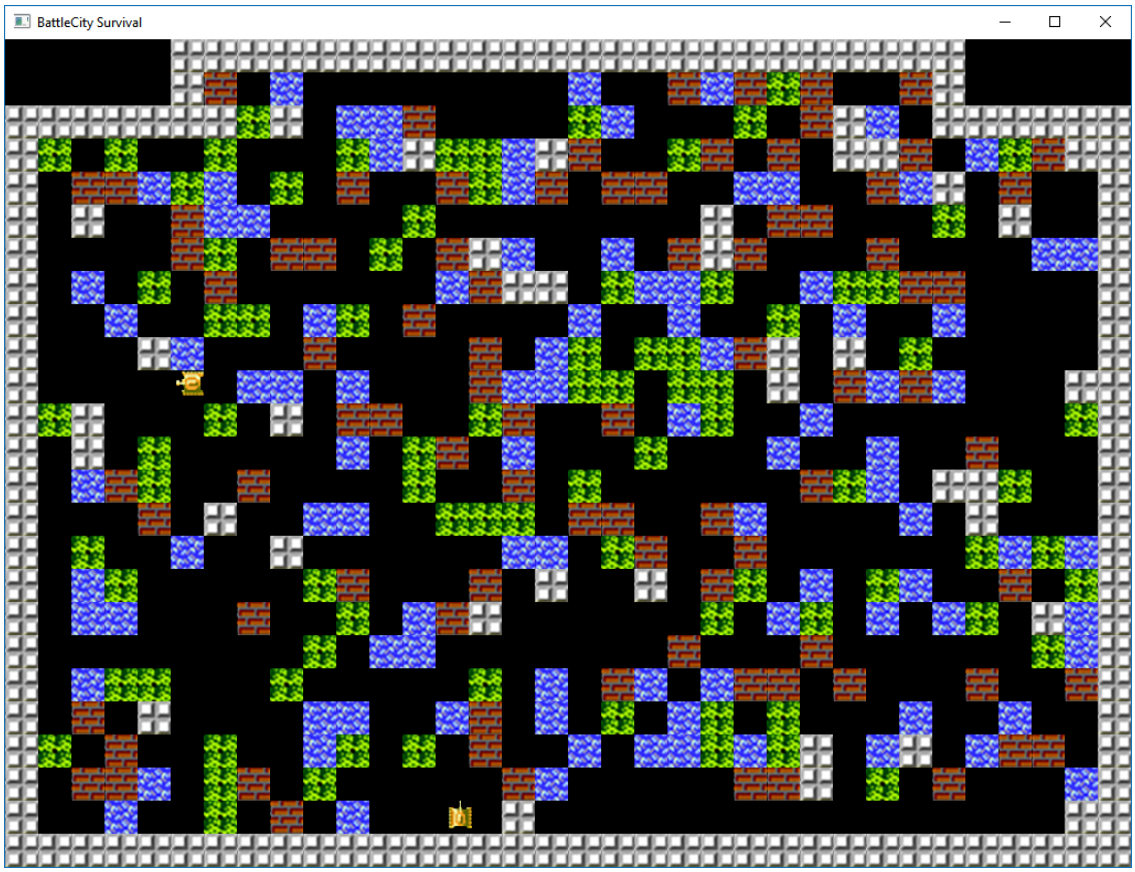
В методе Collision проверяется столкновение снарядов со статичными объектами игрового окружения, такими как кирпичные и бетонные блоки. В случае попадания снаряда в кирпичный блок происходит уничтожение данного кирпичного блока.

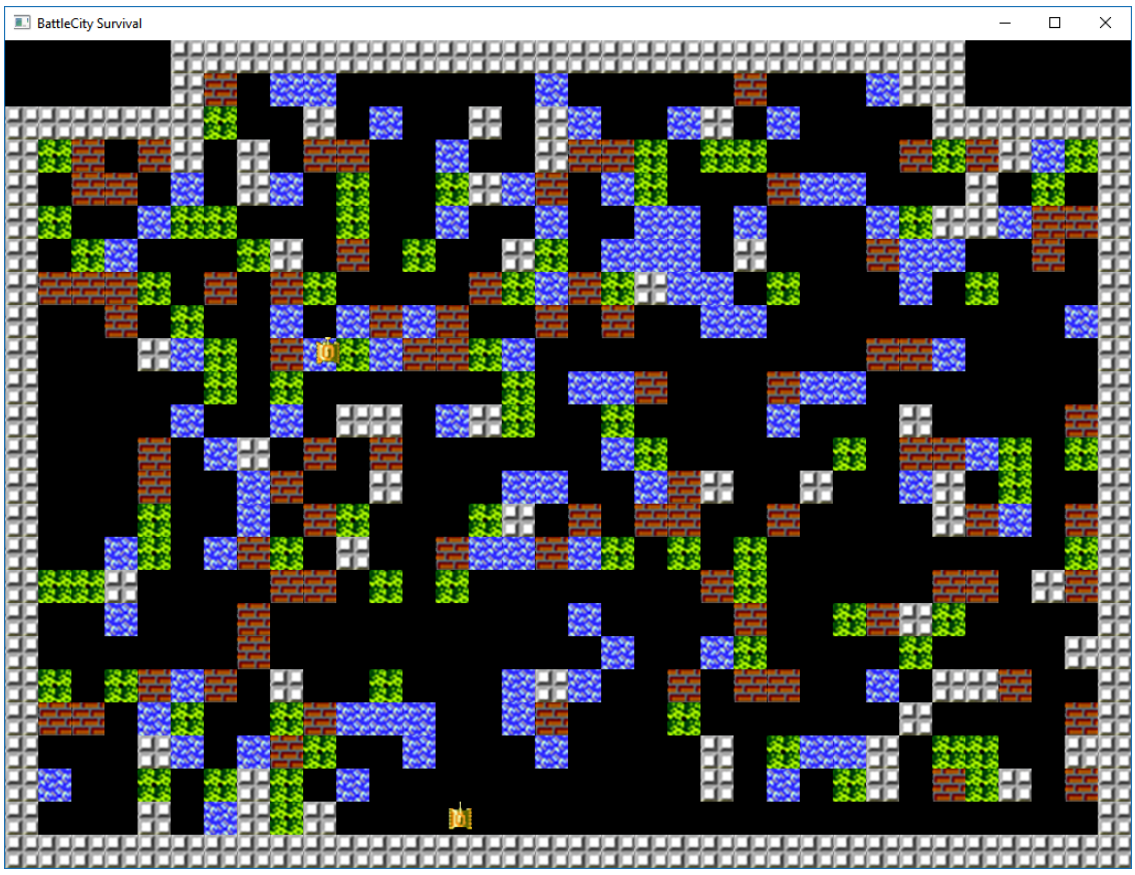
3.1.5 *Описание работы класса Map*

Один из первых классов, создаваемых при запуске игры. Содержит в основном только конструктор и метод renderMap, отвечающий за отрисовку нужного квадрата. Передаёт в метод square класса Painter координаты углов квадрата и координаты левого верхнего угла нужной текстуры в ранее прогруженном спрайт-листе.

Конструктор данного класса Map считывает карту из файла, по заданным алгоритмам при обнаружении определённого символа в файле генерирует новый блок. Блоки расставлены по определённым весам, для большей комфортности игры. Генерируемая карта заносится в ранее созданный двумерный массив.

Примеры генерируемых карт:





3.1.6 *Описание работы класса Painter*

Класс Painter управляет отрисовкой игровой сцены. Для этого у него имеются следующие методы: imageLoad, loadGLtextures, square.

Метод imageLoad производит загрузку и парсинг 32битного bmp файла (с альфа-каналом) с целью получения из него информации о пикселях. Полученные данные используются при создании текстуры.

LoadGLtextures производит создание текстуры из данных, полученных с помощью imageLoad.

Square производит отрисовку квадрата с добавлением поверх него соответствующего спрайта из текстуры по заданным заранее координатам. Координаты передаются при вызове функции square.

3.1.7 *Описание работы класса DetectHit*

Содержит методы intersects и, собственно, сам detectHit, взаимодействующий с тремя списками (врагов, игроков и снарядов). Метод detectHit вызывается в самом конце таймера обновления логики (timerUpdateLogic), после обновления позиций всех объектов игровой сцены. Булева функция intersects служит для проверки пересечения квадратов игроков/врагов со снарядами по заданным координатам левого верхнего угла.

3.1.8 *Описание работы класса Spawner*

Основные методы – setSpawnTimer и addNewEnemy. setSpawnTimer вызывается на каждом проходе таймера timerUpdateLogic, увеличивает счётчик внутри себя. В случае достижения определённой границы вызывается метод addNewEnemy, которые и решает добавить ли нового игрока или нет посредством генерации случайного числа от 0 до 2 включительно. Числа 1 и 2 означают, что добавить нового врага можно. Новый враг (в случае выполнения предыдущих условий) заносится в список врагов в текущем проходе таймера timerUpdateLogic

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

4.1 Базовые сведения по установке и запуску программы

Данное приложение является представляет собой ремейк популярной игры «Battle City».

Минимальные системные требования для ПК:

- Система: 32-битная Windows 7, Windows 8.1 и Windows 10
- Процессор (Intel): Intel Core 2 Duo E6300 или аналогичный
- Процессор(AMD): AMD Phenom II X2 560 или аналогичный
- Память: 1Гбайт ОЗУ
- Место на жестком диске: не менее 30 Мбайт
- Видеокарта (NVIDIA): NVIDIA GeForce GT 610 или выше
- Видеокарта (AMD): AMD Radeon R7 240 или выше
- OpenGL: видеокарта, совместимая с версией 2.1 или аналогичная

Рекомендуемые системные требования для ПК:

- Система: 64-битная система Windows версии 10 и выше
- Процессор (Intel): i7-5820K или аналогичный
- Процессор(AMD): AMD FX-9590 или аналогичный
- Память: 16 Гбайт ОЗУ
- Место на жестком диске: не менее 30 Мбайт
- Видеокарта (NVIDIA): NVIDIA GeForce GTX 970 или выше
- Видеокарта(AMD): AMD Radeon R9 390 или выше
- OpenGL: видеокарта, совместимая с версией 4.5 или аналогичная

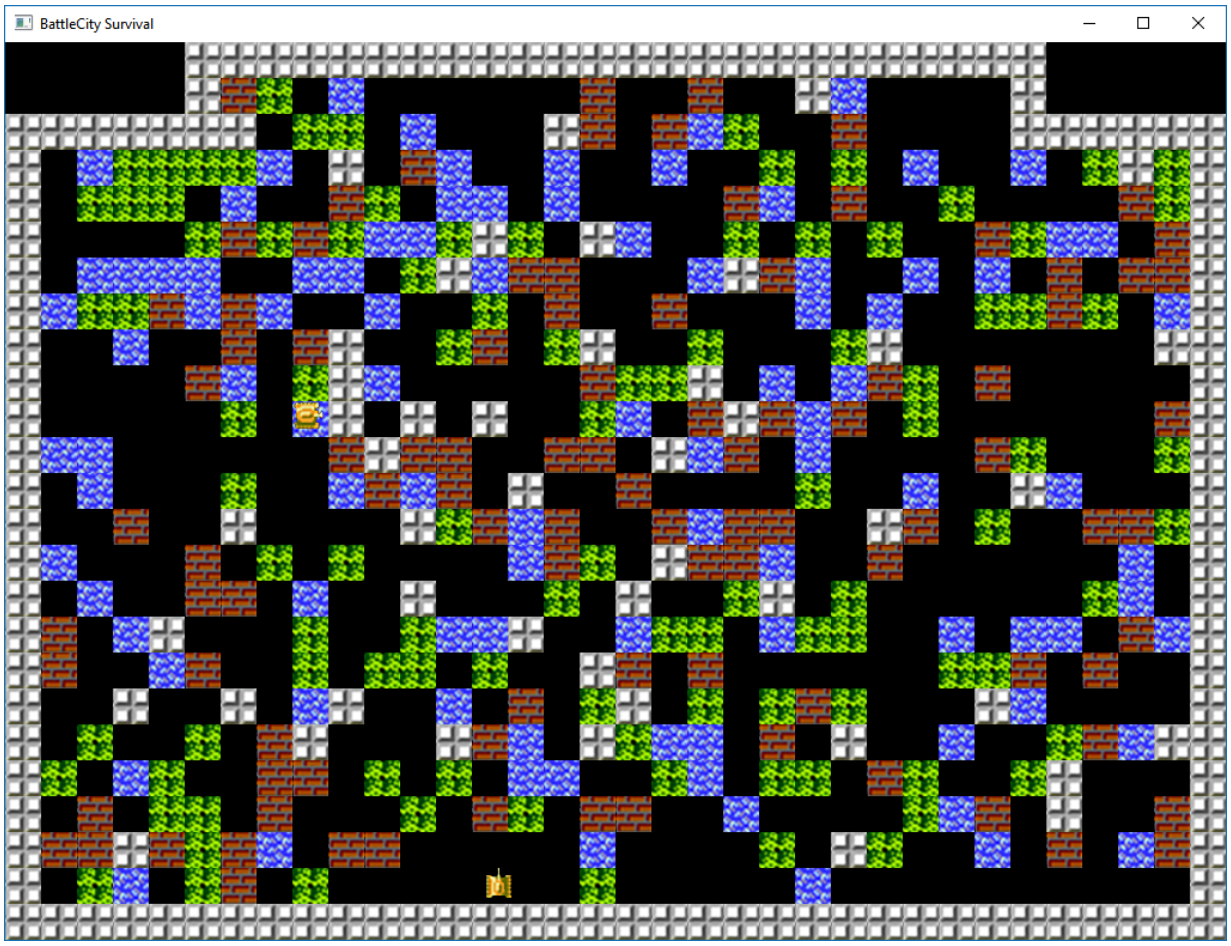
Для сборки из исходных кодов потребуются:

Microsoft Visual Studio (желательно версия 2015 с update 2 или новее)

Наличие следующих библиотек:

- Glut или freeglut
- Glew

4.2 Обзор основных элементов интерфейса программы



При запуске приложения, происходит генерация карты и запуск игры. Появляется главный игрок и 1 враг. Для управления игроком используются клавиши со стрелками и пробел для выстрела.

В отличии от оригинальной игры “Battle City” моя игра “BattleCity Survival” отличается геймплейной частью, однако сохраняет графическую атмосферу тех времён.

Основное отличии от оригинала, это то, что данная игра основана на режиме «выживание», это подразумевает то, что вам не нужно больше прогружать множество мельчайших уровней, а теперь вы будете сражаться на случайно сгенерированной карте за право быть лидером по количеству очков. На данном этапе ваши рекорды никуда не сохраняются.

Ещё одно отличие от оригинала заключается в том, что здесь нет ограничения по одновременно находящимся врагам на карте. Вы можете сражаться в одиночку против сотней, тысяч врагов (зависит от конфигурации вашего компьютера).

ТЕСТИРОВАНИЕ

При успешном запуске игры пользователь увидит окно со сгенерированной картой, одним игроком и одним врагом.

В ходе разработки было выявлено огромное множество проблем, связанных с геймплеем. Многие из этих проблем были решены полностью, для решения других пришлось немного изменить логику игры.

Одна из проблем, это не самая лучшая логика поведения искусственного интеллекта.

Другая проблема, это стрельба врагов, которая происходит по истечению определённого таймера.

ЗАКЛЮЧЕНИЕ

На момент сдачи записки в игре работает базовый геймплей и имеются основные текстуры. Игра имеет простой и удобный интерфейс.

На момент сдачи не удалось наделить ботов алгоритмами для поиска оптимального пути, а также более «умную» стрельбу врагов. Это связано с нехваткой времени на разработку проекта и трудностями, возникшими при освоении API OpenGL.

В дальнейшем в программе будут решены описанные выше проблемы, а также будут добавлены много нововведений:

Приоритетные задачи:

- Переход на современнейший игровой движок Unreal Engine (4.12.1 (версия от 7.6.2016) или новее)
- Переход от 2D в 3D
- Улучшение логики AI
- Переход на формат отображения 16:9
- Улучшение управления
- Добавление графических эффектов
- Добавление текстур сверхвысокого разрешения
- Добавление звуковых эффектов

Полученный в ходе курсового проектирования опыт пригодится в дальнейшем при работе с графикой, написании игр.

СПИСОК ИСТОЧНИКОВ

1. Battle City - Википедия [Электронный ресурс] – Электронные данные. – Режим доступа: https://ru.wikipedia.org/wiki/Battle_City

2. BMP - Википедия [Электронный ресурс] – Электронные данные. – Режим доступа: https://ru.wikipedia.org/wiki/BMP

3. Writting Snake game in 10 minutes [Электронный ресурс] – Электронные данные. – Режим доступа: https://www.youtube.com/watch?v=GiZGEFBGgKU

4. Tetris game in 10 minutes [Электронный ресурс] – Электронные данные. – Режим доступа: https://www.youtube.com/watch?v=q1bHWvpMqtI

5. Writting Breakout game in 10 minutes [Электронный ресурс] – Электронные данные. – Режим доступа: https://www.youtube.com/watch?v=4jMSVaQfoEQ

6. GameDev.net Community Forums [Электронный ресурс] – Электронные данные. – Режим доступа: http://www.gamedev.net/page/resources/_/technical/opengl/rendering-efficient-2d-sprites-in-opengl-using-r2429

7. NeHe Productions: Texture Mapping [Электронный ресурс] – Электронные данные. – Режим доступа: http://nehe.gamedev.net/tutorial/texture_mapping/12038/