Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Компьютерные системы и сети (КСиС)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

по курсовому проекту
на тему

ПРОГРАММНОЕ СРЕДСТВО ИГРА «Packet Tanks 2D»

БГУИР КР 1-40 01 01 614 ПЗ

Выполнил:
студент гр. 851006 Крот Е. Д.

Руководитель: Шимко И.В.

Минск 2020

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

Лапицкая Н.В.

_____

(подпись)

_____2020 г.

ЗАДАНИЕ
по курсовому проектированию

Студенту *Кроту Евгению Дмитриевичу*

1. Тема работы *Программное средство игра «Packet Tanks 2D»»*

2. Срок сдачи законченной работы *05.06.2020г.*

3. Исходные данные к работе *Среда программирования MS Visual Studio 2019. Документация по фреймворкам MonoGame и Microsoft XNA.*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Введение
1 Анализ предметной области
2 Постановка задачи
3 Разработка программного средства
4 Тестирование и проверка работоспособности программного средства
5 Руководство пользователя программного средства
Заключение
Список использованных источников
Приложения

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)
Схема алгоритма в формате А1

6. Консультант по курсовой работе *Шимко И.В.*

7. Дата выдачи задания *10.02.2020.*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):
Раздел 1. Введение к 28.02.2020г. – 10 % готовности работы;
Раздел 2 к 15.03.2020г. – 30% готовности работы
Раздел 3 к 15.04.2020г. – 60% готовности работы
Раздел 4 к 10.05.2020г. – 80% готовности работы
Раздел 5. Заключение. Приложения к 20.05.2020г. – 90% готовности работы;
Оформление пояснительной записки и графического материала к 01.06.2020г. – 100% готовности работы.
Защита курсового проекта с 01.05.2020г. по 09.06.2020г.

РУКОВОДИТЕЛЬ _____ *Шимко И.В.*
(подпись)

Задание принял к исполнению _____ *Крот Е.Д. 10.02.2020г.*
(дата и подпись студента)

СОДЕРЖАНИЕ

Введение

IT-специалист – широкое понятие, объединяющее в себе представителей многих профессий, работающих в области информационных технологий. Это программисты, разработчики, администраторы сетей и баз данных, модераторы, специалисты по робототехнике, по информационной безопасности, web-дизайнеры и 3D-художники. При этом, с проникновением информационных технологий во всё новые сферы деятельности, появляются новые профессии для IT-специалистов. По данным на сегодняшний день и мнению многих аналитиков [1] специалисты данной области являются востребованными и будут востребованы в ближайшем будущем. В частности, сегодня в мире появляется всё больше новых перспективных игровых разработок и проектов.

GameDev – сокращение от Game Development (разработка игр). В процесс разработки игрового приложения входят: разработка дизайна игрового процесса (геймплея), программирование игрового «движка» или использование готовых решений, разработка визуального концепта и его составляющих, создание графики и моделирование объектов, музыкального и звукового сопровождения, решение множества возникающих в процессе задач из различных сфер.

Игровая индустрия продолжает развиваться и с каждым годом её прибыль на рынке увеличивается. Таким образом, идея отработки навыков разработки игровых проектов является перспективным и интересным направлением.

Целью работы является разработка игры «Tanks 2D» в жанре Аркады для различных платформ персональных устройств, для чего используется достаточно распространённый фреймворк «MonoGame», основанный на платформе Microsoft XNA.

В рамках курсового проекта ставятся следующие задачи:

1. Осуществить постановку игровой задачи;
2. Выполнить обзор существующих аналогов, программных средств разработки компьютерной игры;
3. Спроектировать приложение;
4. Составить документацию на полученное приложение.

1 Анализ предметной области

1.1 Анализ существующих аналогов

На сегодняшний день существует множество игровых приложений жанра Аркада. Основное распространение данный жанр получил в сфере мобильных игр, поскольку позволяет пользователям «убивать время», например, по пути на работу. Рассмотрим самые распространённые 2D аркады.

Space Invaders [2]

Самой популярной аркадой является Space Invaders (рис. 1.1.1), которая была одна из первых в этом жанре. Цель игры – уничтожить всех врагов на экране. Изначально игра имела простую графику и показатель здоровья, при достижение нулевого значение которого игра завершалась, однако позже она усовершенствовалась (были введены новые цветовые схемы и показатель очков) (рис. 1.1.2).

Рисунок 1.1.1 – Скриншот игры Space Invaders

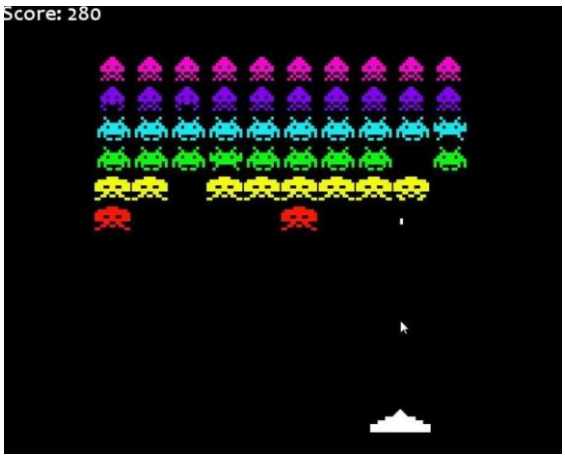

Рисунок 1.1.2 – Скриншот улучшенной Space Invaders

Radiant [3] – Аналог игры Space Invaders выпущенный в 2010 году. Цель игры – уничтожить всех врагов на экране. Главная особенность игры – улучшенная 2D графика. По мере увеличение времени игры увеличивается и сложность – появляются больше врагов, однако, у пользователя нет возможности изменять модели игры.

Достоинства:

– улучшенная 2D-графика;

– присутствует нарастание сложности по мере прохождения.

Недостатки:

– в игре присутствуют только стандартные однотипные модели;

– сложность присутствует, однако, у пользователя нет возможности выбрать ее самому;

– отсутствует возможность выбрать звуковое сопровождение.

1.2 Выбор программной среды разработки

В настоящее время существует множество программного обеспечения, позволяющего разрабатывать собственные проекты различного характера, в том числе и компьютерные игровые приложения. Выделяют две основные группы ПС, характеризующиеся:

– Использованием программного компонента (комплекса), позволяющего создавать игровое приложение (игровой движок);

– Созданием собственного «движка» с помощью языка программирования и специализированных библиотек.

Рассмотрим каждую из групп подробнее.

Игровые движки – базовое программное обеспечение компьютерной игры [4]. Термин «игровой движок» появился в середине 1990-х в контексте компьютерных игр жанра шутер от первого лица, похожих на популярную в то время Doom. Архитектура программного обеспечения Doom была построена таким образом, что представляла собой разумное и хорошо выполненное разделение центральных компонентов игры (например, подсистемы трёхмерной графики, расчёта столкновений объектов, звуковой и других) и графических ресурсов, игровых миров, формирующие опыт игрока игровые правила и другое. Как следствие, это получило определённую ценность за счёт того, что начали создаваться игры с минимальными изменениями, когда при наличии игрового движка компании создавали новую графику, оружие, персонажей, правила игры и тому подобное [5].

Unity 3D [6] – это мощный инструмент для создания приложений и игр под разные платформы. Имеется возможность создавать как 2D, так и 3D проекты. Имеется поддержка DirectX и OpenGL, поддерживает множество различных форматов данных. Unity 3D поддерживает такие языки программирования как C# и JavaScript. Рабочее окно Unity 3D представлено на рисунке 1.2.1.

Достоинства:

– кроссплатформенность;

– большое русскоязычное общество, наличие большого количества уроков, самоучителей;

– простота и эффективность разработки;

– регулярные обновления.

Недостатки:

– требует дорогостоящие технические комплектующие;

– платная коммерческая лицензия;

– неудобная работа при создании 2D – приложения.

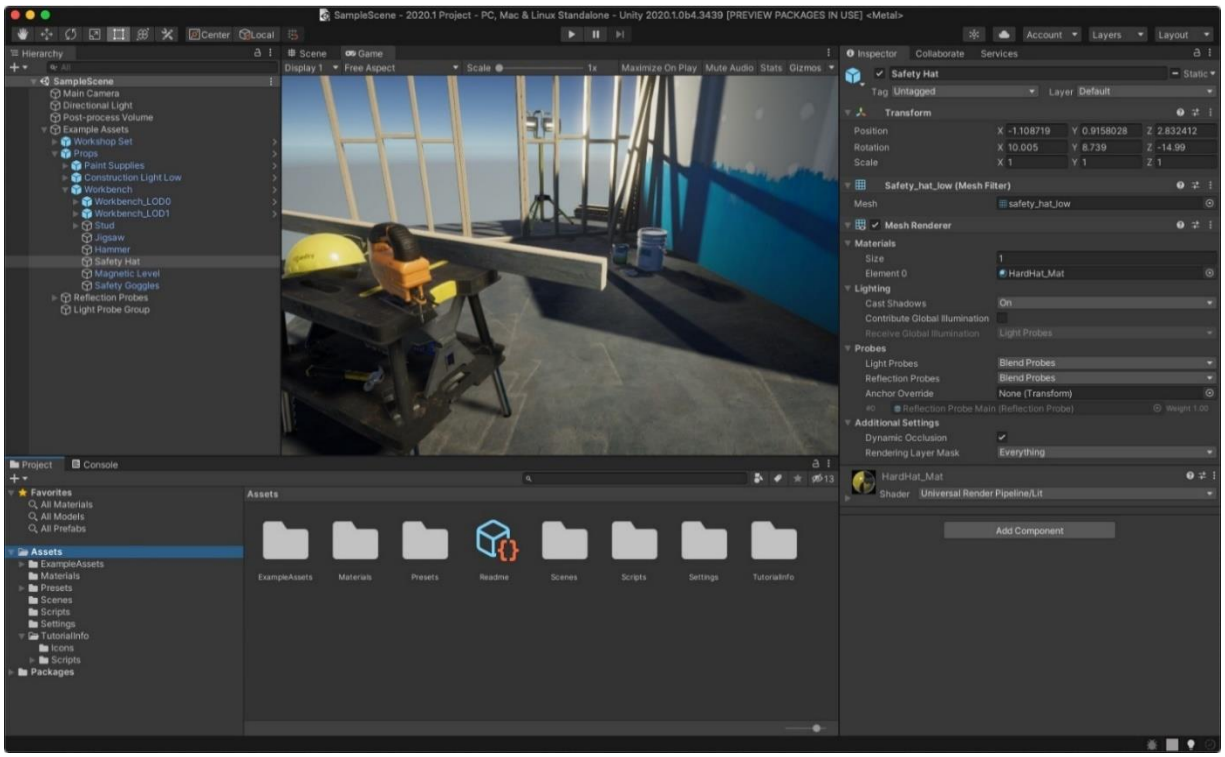

Рисунок 1.2.1 – Рабочее окно Unity 3D

При всех своих достоинствах, 2D редактор среды Unity представляет собой урезанную 3D версию, что в некоторых ситуациях приводит к нестабильной или неоправданно усложнённой работе со средой.

CryEngine [7] – самый мощный из современных игровых движков, обеспечивающий фотореалистичную графику с поддержкой DirectX 12 и шейдеров (теней). Третья версия «движка» создана в 2009 году, распространяется платно. Пятая (V) версия «движка» вышла в конце 2016 года и имеет условно бесплатную лицензию. Основным направлением данного средства является создание игр жанра «Шутер». Рабочее окно CryEngine представлено на рисунке 1.2.2.

Достоинства:

– высокие показатели графики;

– кроссплатформенность.

Недостатки:

– обладает малым набор инструментов;

– направлен на создание определённого жанра игр;

– самый ресурсоемкий из всех представленных.

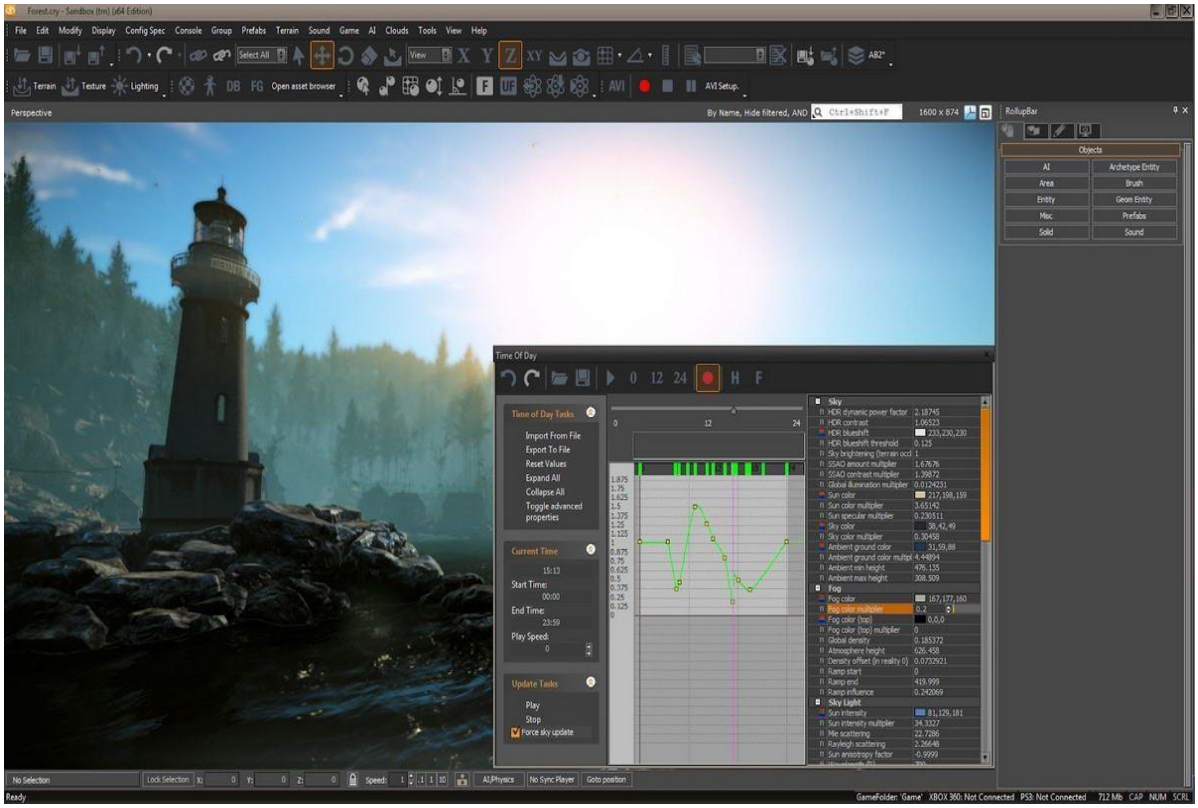

Рисунок 1.2.2 – Рабочее окно CryEngine

Unreal Engine [8] – игровой «движок», разрабатываемый и поддерживаемый компанией Epic Games. Предоставляет большой набор инструментария для создания 2D и 3D проектов. Начиная с четвертой версии – распростра-

няется бесплатно. Однако до тех пор, пока разработчик не выпустит свой первый коммерческий продукт на основе UE4. Является основным движком множества современных игр. Рабочее окно игрового «движка» Unreal Enginge представлено на рисунке 1.2.3.

Достоинства:

– высокие показатели графики;

– кроссплатформенность;

– большие и регулярные обновления;

– разнообразный инструментарий.

Недостатки:

– высокие технические требования;

– сложность в освоение;

– малое количество русскоязычной документации;

– коммерческая лицензия.

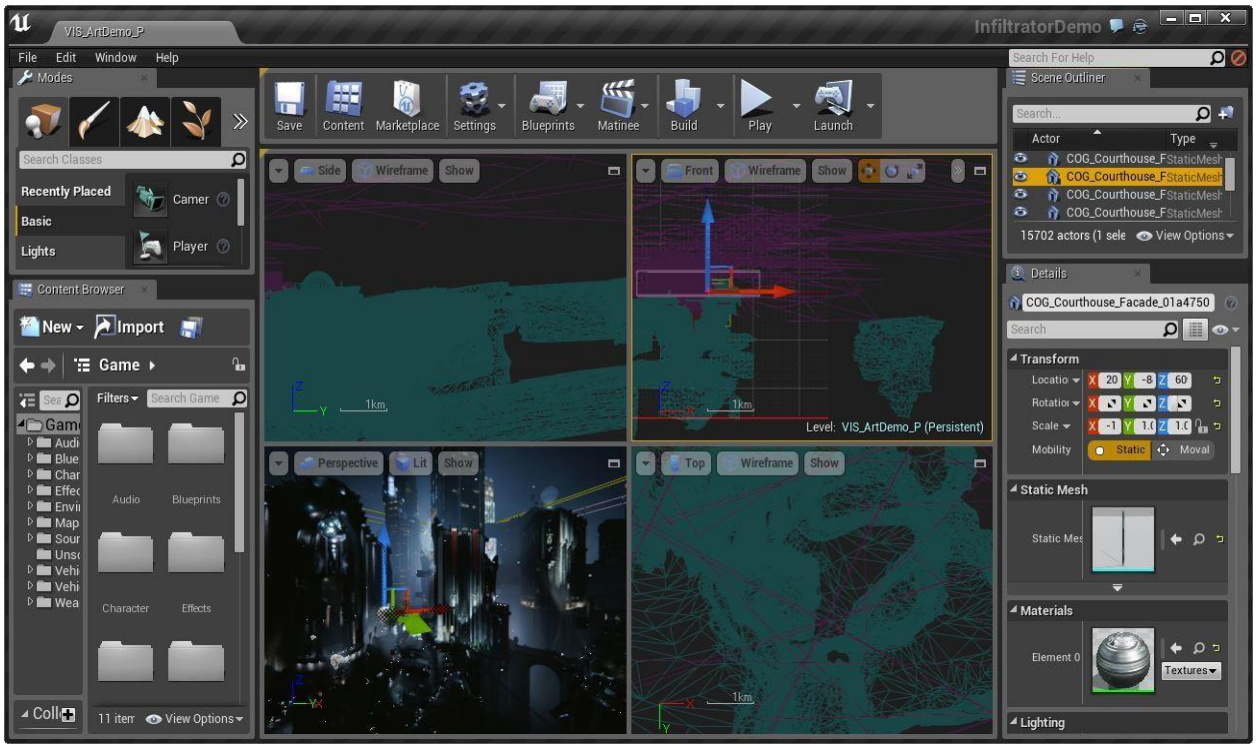

Рисунок 1.2.3 – Рабочее окно Unreal Engine

Все рассмотренные игровые «движки» в основном предназначены для создания 3D приложений и требуют высокие показатели технических требований, поэтому использовать их нецелесообразно (у многих пользователей ПК отсутствуют современные и мощные комплектующие).

Перейдём к рассмотрению идеи создания собственного примитивного игрового движка на основе платформы MonoGame.

MonoGame представляет кроссплатформенную OpenSource-реализацию популярного фреймворка Microsoft XNA 4, который предназначен для работы с графикой и прежде всего для создания игр [9]. К настоящему моменту MonoGame практически полностью внедрил все функции, которые имелись в XNA 4, и кроме того, предлагают кучу возможностей, которые ранее не имелись в XNA. MonoGame частично или полностью поддерживает следующие платформы: Xbox 360, Windows (в том числе последнюю 10-ю версию), Windows Phone, iOS, Android, Mac OS X, Linux.

Основными достоинствами платформы следует выделить направленность на создание не только 3D игр, но и достаточно сложных 2D проектов, низкий вес самой платформы (десятки мегабайт против нескольких гигабайт у указанных выше 3D движков), а также актуальные для оппонентов кроссплатформенность и лёгкость обучения. Таким образом, MonoGame является оптимальным выбором для обучения созданию видеоигр, в особенности для создания простые проектов в жанре «Аркада».

2 Постановка задачи

В рамках курсового проекта необходимо реализовать программное игровое приложение «Packet Tanks». Создать игру необходимо в жанре «Аркада», также выбран стиль 2D. При запуске приложения пользователь должен иметь возможность получить список доступных в сети игроков, начать поединок с любым из них.

Главная цель игры – уничтожить противника. Игрок с помощью выстрелов (атаки) должен уничтожить танк противника, не допустив при это уничтожения собственного танка. Оба игрока имеют полоску здоровья (HP), отображаемую в верхней части экрана. При каждом попадания снаряда уровень здоровья танка уменьшается, игрок, первым потерявший всё здоровье (уровень 0) считается проигравшим.

Для усложнения игрового процесса поле представляет собой ограниченную по периметру область, также частично перекрытую стенами внутри. При попадании снаряду в стену снаряд взрывается, стена, аналогично танку, теряет часть своего здоровья. При достижении стеной нулевого уровня здоровья она исчезает, давая игроку возможность находить неожиданные пути к врагу.

В рамках проектирования приложения необходимо так же предусмотреть возможность дальнейшего расширения функциональности. Следует продумать возможность дальнейшего добавления новых типов танков, карт, спроектировать универсальное решение для поиска столкновений объектов.

3 Разработка программного средства

3.1 Проектирование алгоритма поиска оппонента

Основным элементом многопользовательской игры несомненно является соревновательный элемент. Поскольку на данном этапе игра не подразумевает сражений с игровым искусственным интеллектом, работа приложения будет невозможно без наличия двух игроков. Таким образом, начать проектирование ПС следует именно с проектирования сетевого взаимодействия.

Для реализации сетевого взаимодействия необходимо в первую очередь выбрать протокол передачи сообщений по IP-сетям. Основными протоколами в современном мире являются TCP и UDP, и, несмотря на многие преимущества TCP (гарантия доставки и очередности, разбивка на пакеты) выбор протокола для многопользовательских игр всё же лучше делать в пользу UDP [10].

Сразу после запуска игроку предлагается ввести своё имя на текущую сессию. После проверки введённого ник-нейма приложение позволяет пользователю подключиться к сети. Для установления соединения со всеми находящимися в текущий момент в сети игроками приложение в момент запуска отправляет широковещательный (многоадресный) пакет, содержащий информацию о ник-нейме подключившегося пользователя. Данный пакет имеет тип 0, и любое находящееся в сети приложение при получении такого пакета обязано отправить ответное сообщение типа 1, содержащее его имя. Приложение пользователя сразу после отправки пакета типа 0 переходит к постоянному ожиданию ответных сообщений типа 1, а также сообщений типа 0 от новых пользователей сети (ранее это же было сделано и другими пользователями сети). Таким образом, после получения приложением пакета типа 0 или 1 в списке доступных оппонентов появляется новое имя, а в памяти программы так же хранится IP-адрес данного оппонента.

Когда пользователь решит начать сражение с одним из доступных оппонентов, он выбирает необходимое имя из списка, и приложение отправляет по соответствующему адресу приглашение на поединок (пакет типа 2). Оппонент при получении такого сообщения принимает бой и отправляет соответствующее сообщение (пакет типа 3), после чего сразу же «выходит на арену», где ожидает вызвавшего его игрока. Игрок при получении согласия на поединок так же выходит на арену.

3.2 Проектирование алгоритма синхронизации событий

После подключения первого игрока к игровой комнате он посылает оппоненту пакет типа 4 с сигналом к началу сражения. С этого момента начинается

сражение, и после обработки действий игрока перед отрисовкой следующего кадра пользователь передаёт оппоненту пакет типа 6 с информацией о новом положении своего танка. Соответственно, оба игрока во время обработки своих действий так же ожидают сообщение о позиции оппонента, и применяют при отрисовке и обработке событий уже новые координаты танка противника. Несмотря на возможную задержку при передаче информации через интернет и возможную погрешность из-за отсутствия сервера, скорость передвижения танков является сравнительно низкой, и вероятность несоответствия координат между оппонентами даже на несколько пикселей маловероятна. В то же время, использование однорангового соединения в многопользовательских играх целесообразно только при реализации пошаговых игр, например, стратегий наподобие серии HoMM или игр вроде Pocket Tanks. В противном случае более целесообразным можно считать назначение одной из сторон сервером, предназначенным для синхронизации данных между пользователями.

3.3 Проектирование алгоритма поиска столкновений

Платформа MonoGame имеет встроенные средства для определения столкновения двух объектов. Основным является метод Intersects класса Rectangle, которые определяет, пересекаются ли два прямоугольника. В процессе создания приложения выяснилось, что встроенные средства фреймворка позволяют определять столкновения лишь расположенных под прямыми углами объектов, отслеживание же столкновений с помощью окружностей является недостаточно точным, а потому малопригодно для отслеживания столкновения с препятствиями и абсолютно не подходит для слежения за снарядами.

В результате проблем с использованием встроенных средств платформы было решено реализовать собственный алгоритм проверки столкновения объектов. Поскольку все объекты игрового поля являются прямоугольниками известных размеров, расположенными под известными углами по также известным координатам, было решено отслеживать пересечение фигур по пересечению между собой их граней. Так, с помощью математических вычислений определяются четыре вершины произвольного прямоугольника, после чего алгоритм проверяет на пересечение все возможные пары линий, образующих исходные фигуры. Для упрощения вычислений в начале проверки на пересечения используется свойство проекций отрезков (отрезки пересекаются только если их проекции имеют общую часть). Если данное условие соблюдается, вычисляется точка пересечения отрезков, и если данная точка принадлежит промежутка, на котором расположены отрезки, то фигуры пересеклись.

В дальнейшем при возрастании вычислительной нагрузки или необходимости оптимизировать вычисления под более слабые устройства можно дополнительно использовать указанные в начале раздела методы проверки с помощью окружностей или встроенных методов MonoGame, однако во время тестирования «зависаний» обнаружено не было, что можно объяснить в том числе сильно ограниченным количеством объектов, одновременно находящихся на игровом поле (2-4 снаряда; 20-30 блоков, расположенных под прямым углом, что сильно облегчает вычисления).

4 Тестирование программного средства

После написания программы необходимо сделать проверку различных функций и ситуаций в программе, для убеждения в том, что программа работает исправно. Далее будут рассмотрены различные тестовые ситуации, в которых ожидалось возникновение проблем в ходе эксплуатации программы.

Тестирование производилось на персональном компьютере с установленной операционной системой Windows 10, а также на аналогичной виртуальной машине.

4.1 Тестирование функционала программы

Таблица 4.1 – Тестирование функционала программы

Номер теста	Тестируемая функциональность	Ожидаемый и полученный результаты
1	Движение танка при нажатии соответствующей клавиши	Изменение координат положения объекта, отрисовка по новым координатам
2	Поворот танка на месте при нажатии соответствующей клавиши	Изменение угла поворота объекта, отрисовка по новым данным
3	Поворот танка в движении при нажатии соответствующих клавиш	Одновременное изменение координат и угла поворота объекта, движение спрайта по окружности
4	Поворот башни танка относительно неподвижного при нажатии соответствующей клавиши	Изменение угла поворота объекта, отрисовка по новым данным Прим.: Вращение не относительно центра объекта
5	Поворот башни танка относительно подвижного корпуса при нажатии соответствующих клавиш	Изменение угла поворота обоих объектов, изменение абсолютной скорости вращения башни при разнонаправленном вращении
6	Столкновение танка с другими объектами на экране	Невозможность продолжать движение/вращение при обнаружении препятствия на пути
7	Столкновение снарядов с другими объектами на экране	Уничтожение снарядов, снижение показателей уровня здоровья у объекта

Продолжение таблицы 4.1

8	Снижение здоровья объекта до нулевого уровня	Уничтожение объекта (блока) /присуждение победы игроку при уничтожении вражеского танка
9	Отображение изменений уровня здоровья в Health Bar	При изменении показателей здоровья танков игрока и оппонента изменяется размер отображаемой полоски здоровья
10	Запрет стрельбы при перезарядке орудия	В течение 2 секунд после выстрела возможность повторного выстрела блокируется
11	Запрет стрельбы при зажатой клавише выстрела	Повторный выстрел до отпускания клавиши стрельбы блокируется
12	Получение списка доступных оппонентов	Постепенное заполнение списка игроков после подключения к сети
13	Запуск игры с выбранным оппонентом	Переход в соответствующее окно и начало игры

4.2 Выводы по прохождению тестирования

Приложение успешно прошло все тесты, что показывает корректность работы разработанного программного средства и соответствие функциональным требованиям.

5 Руководство пользователя программного средства

5.1 Системные требования

Для нормальной работы программного средства необходимы следующие минимальные системные требования:

– операционная система: Windows 10, Windows 8.1, Windows Mobile, Linux, Android;

– процессор: с тактовой частотой 1Ггц;

– оперативная память 512 МБ;

– свободное место на жестком диске: 25 МБ.

5.2 Использование

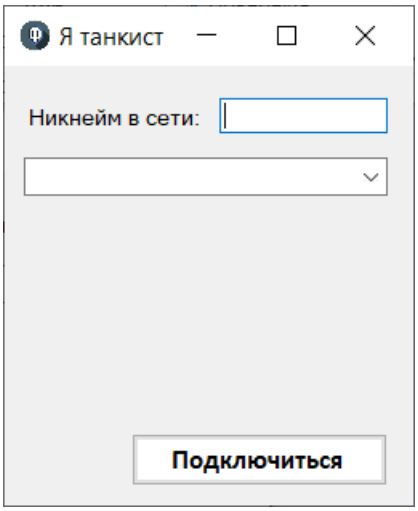

Рисунок 5.2.1 – Окно авторизации

Сразу после запуска приложения пользователь видит окно авторизации (Рис. 5.2.1). Основными компонентами формы являются текстовое поле «Никнейм в сети», выпадающий список «Оппоненты» и кнопка «Подключиться – Сыграть». После ввода пользователем валидного (допустимого) имени и нажатия кнопки происходит изменение текста кнопки, все находящиеся в сети в данный момент оппоненты добавляются в список (Рис. 5.2.2).

После выбора пользователем любого доступного оппонента ему становится доступна кнопка «Сыграть» (Рис. 5.2.3), при нажатии на которую происходит установка соединения с другим приложением и переход к процессу игры.

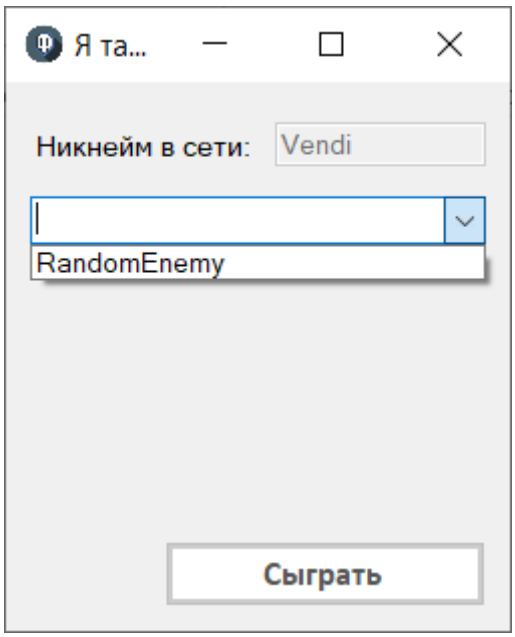

Рисунок 5.2.2 – Окно авторизации со списком пользователей

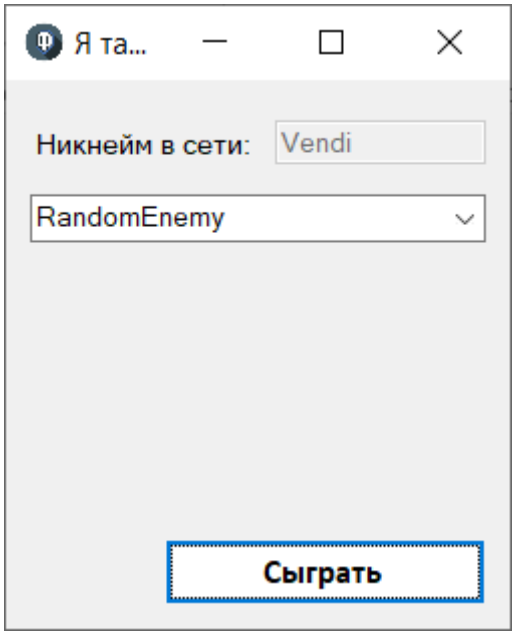

Рисунок 5.2.3 – Окно авторизации с выбранным оппонентом

После установки соединения оба игрока оказываются на игровом поле. Танк игрока изображается зелёным цветом, противник подсвечивается красным цветом. В верхней части экрана отображаются очки здоровья (Health Bar) оппонентов (Рис. 5.2.4).

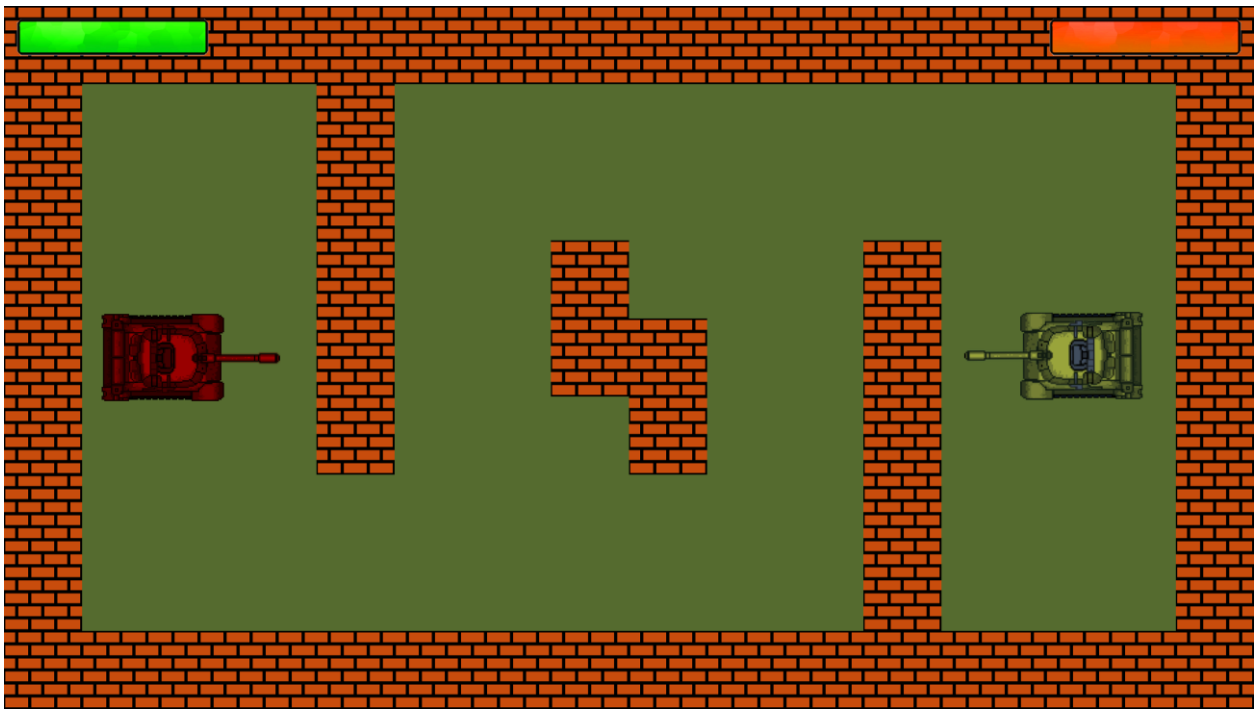

Рисунок 5.2.4 – Окно игры

Управление танком осуществляется с помощью клавиатуры. При нажатии клавиш W и S (русская раскладка Ц и Ы) танк движется вперёд или назад относительного своего текущего местоположения. Нажатие клавиш A и D (Ф и В) приводит к повороту танку против или по часовой стрелке. Поворот во время движения допускается, при этом при движении назад направление поворота инвертируется. Таким образом, движение по игровому полю (Рис. 5.2.5) аналогично большинству подобных игр, в дальнейшем легко реализуется добавление возможности назначения пользователем удобных ему клавиш управления или реализация управления с помощью джойстика или даже сенсорного интерфейса.

Стрельба (Рис. 5.2.6) осуществляется клавишей пробел. После выстрела танку необходимо время на перезарядку (на данный момент 2 секунды), повторный выстрел при зажатой клавише выстрела не допускается. Для попадания во врага необходимо повернуть ствол (башню) в его направлении, для чего используются клавиши Q (Й) и E(У), либо стрелки влево и вправо на клавиатуре (Рис. 5.2.7).

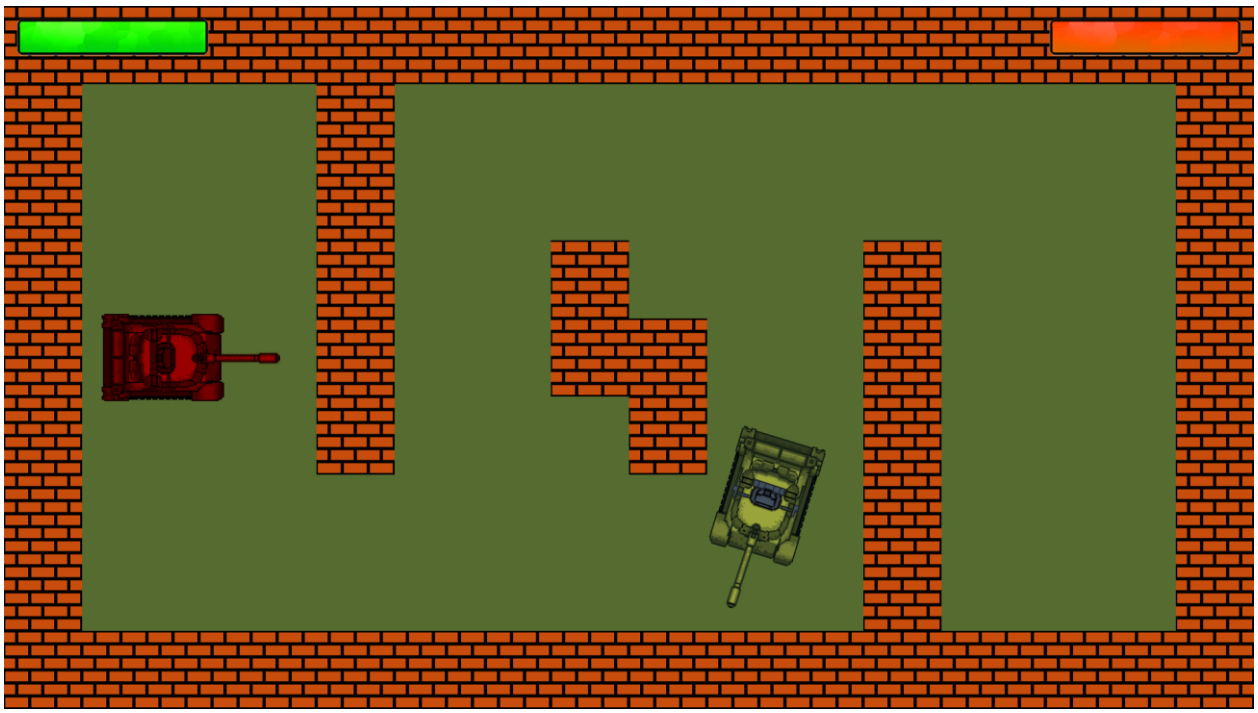

Рисунок 5.2.5 – Окно игры

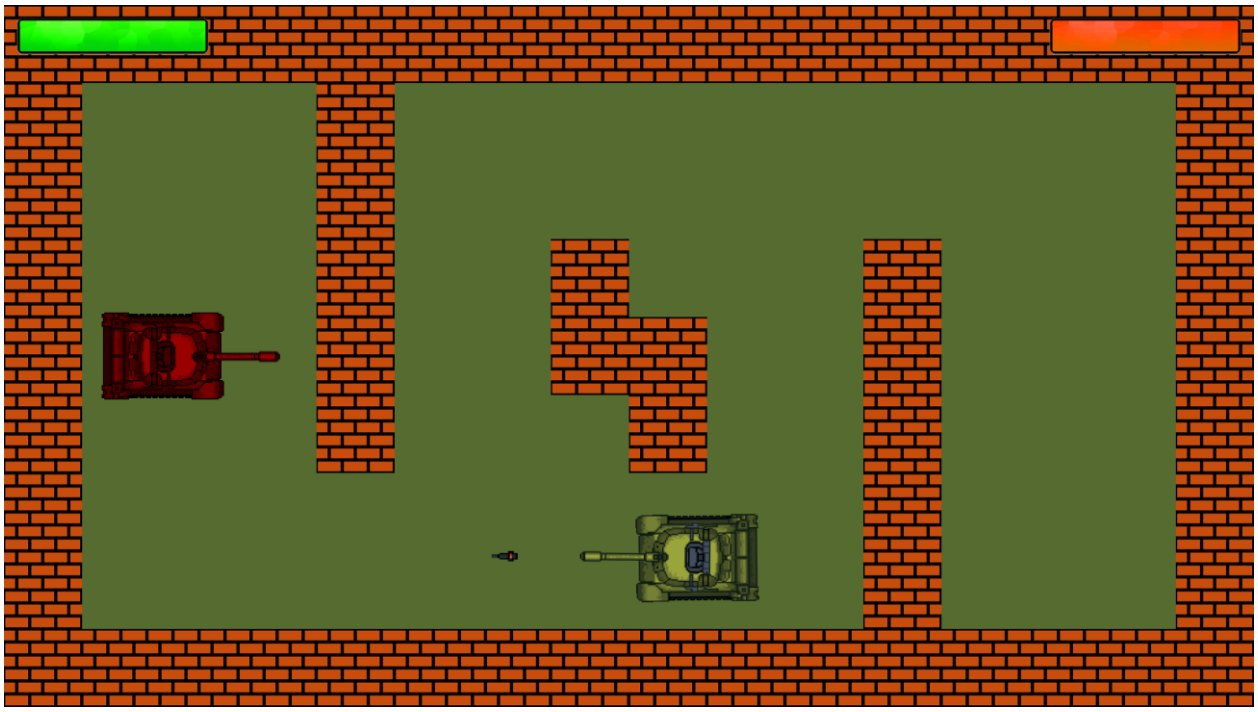

Рисунок 5.2.6 – Окно игры

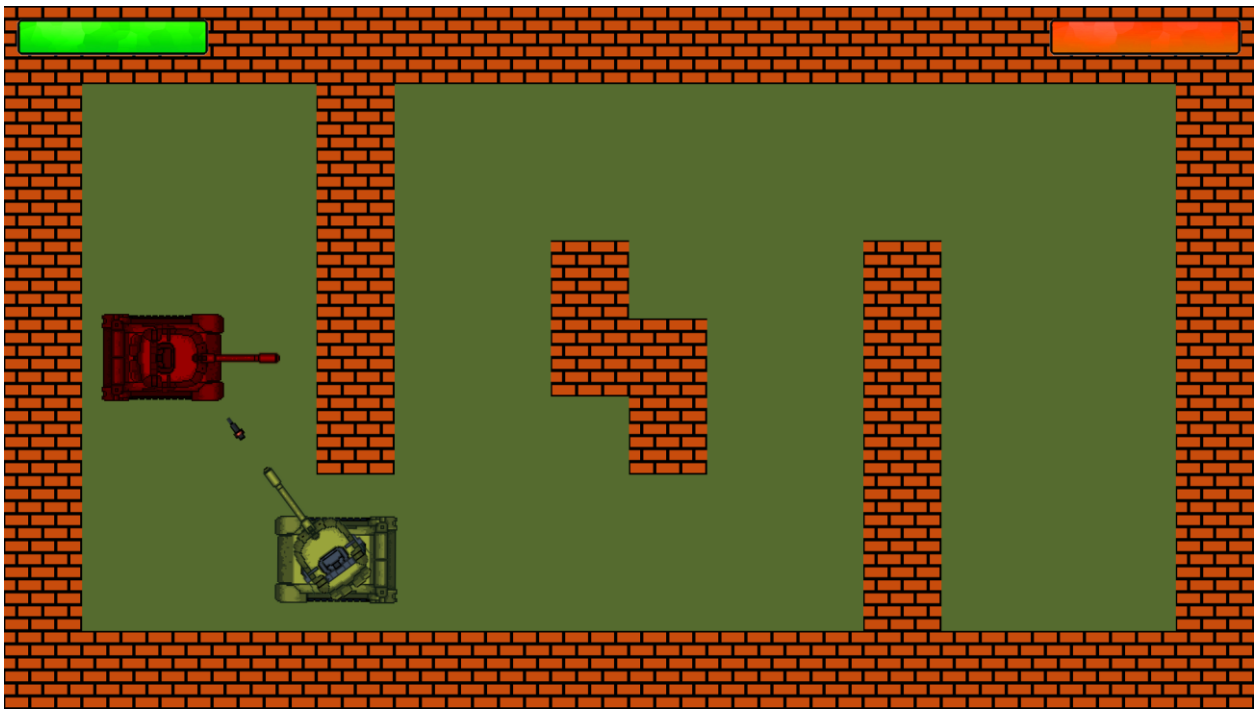

Рисунок 5.2.7 – Окно игры

ЗАКЛЮЧЕНИЕ

Доля пользователей ПК и интернет аудитории растет ежедневно.

В основном пользователи используют ПК в качестве развлекательной платформы. Таким образом, игровая аудитория продолжает расти, тем самым мотивируя разработчиков создавать новые игровые проекты.

В рамках курсовой работы была изучена и проработана технология разработки игр путем создания игрового приложения «Packet Tanks» в жанре «Аркада». Разработка велась на языке программирования C# с использованием фреймворка MonoGame на основе Microsoft XNA и стандартных системных библиотек.

Для достижения данной цели были решены следующие задачи:

– осуществлена постановка игровой задачи;

– произведен анализ аналогов и программных средств разработки компьютерной игры;

– произведен поиск и подбор бесплатных моделей;

– повторены и углублены знания в языке программирования C#;

– изучена работа с платформой MonoGame;

– спроектировано приложение;

– составлены руководство пользователя и документация.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

[1] Профессии будущего. [Электронный ресурс]. – Режим доступа: http://www.proforientator.ru/publications/articles/detail.php?ID=5454

[2] Space Invaders. [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/SpaceInvaders

[3] Radiant. [Электронный ресурс]. – Режим доступа: http://www.hexag.net/radiant/

[4] Игровой движок – Википедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Игровой_движок

[5] Jason, Gregory. Game Engine Architecture : [англ.]. – CRC Press, 2009

[6] Unity [Электронный ресурс]. – Режим доступа: https://unity3d.com/ru

[7] CRYENGINE – The complete solution for next generation game development [Электронный ресурс]. – Режим доступа: https://www.cryengine.com

[8] What is Unreal Engine 4? [Электронный ресурс]. – Режим доступа: https://www.unrealengine.com/

[9] Руководство по игростроению на платформе MonoGame [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/monogame/

[10] Сетевое программирование для разработчиков игр. Часть 1: UDP vs. TCP [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/209144/

ПРИЛОЖЕНИЕ А
(Обязательное)
Исходный код программы

Файл FormLogin.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Tanks2D
{
  public partial class FormLogin : Form
  {
    internal String UserName;
    internal bool Connected = false;
    internal bool Replied = false;
    internal List<Sub> Subs = new List<Sub>();
    internal Task receiveUDP;

    internal class Packet
    {
      public byte Type;
      public UInt16 MsgLength;
      public byte[] Data;

      public Packet(byte T, string D)
      {
        Type = T;
        Data = Encoding.Unicode.GetBytes(D);
        MsgLength = (UInt16)(3 + Data.Length);
      }
      public Packet(byte[] D)
      {
        Type = D[0];
        MsgLength = BitConverter.ToUInt16(D, 1);
        Data = new byte[MsgLength - 3];
        Buffer.BlockCopy(D, 3, Data, 0, MsgLength - 3);
      }
      public Packet(byte T)
      {
        Type = T;
        MsgLength = 3;
```

```
  }
  public byte[] getBytes()
  {
    byte[] data = new byte[MsgLength];
    Buffer.BlockCopy(BitConverter.GetBytes(Type),      0,
                     data, 0, 1);
    Buffer.BlockCopy(BitConverter.GetBytes(MsgLength),
                     0, data, 1, 2);
    if (MsgLength > 3)
      Buffer.BlockCopy(Data, 0, data, 3, MsgLength - 3);
    return data;
  }
}

internal class Sub
{
  public string name;
  public IPEndPoint endPoint;
  public Sub(string str, IPEndPoint iep)
  {
    name = str;
    endPoint = iep;
  }
}

public FormLogin()
{
  InitializeComponent();
}

private    void    buttonConnect_Click(object    sender,
                                       EventArgs e)
{
  if (buttonConnect.Text == "Подключиться")
  {
    UserName = textBoxName.Text.Trim();
    if (UserName.Length > 3 && UserName.Length < 13)
    {
      Connect();
    }
    else
      MessageBox.Show("Введите имя для вашего аккаунта
                       (4-12 символов)!");
  }
  else
  {
    UdpClient request = new UdpClient();
    foreach (Sub sub in Subs)
    {

      MessageBox.Show(sub.name+"     "     +     comboBox.
                      SelectedItem.ToString());
```

```
        if (sub.name == comboBox.SelectedItem.ToString())
        {
          MessageBox.Show("Equal");
          request.Connect(sub.endPoint.Address, 8006);
          Packet msg = new Packet(2, UserName);
          request.Send(msg.getBytes(), msg.MsgLength);
          request.Close();
          break;
        }
      }
    }
  }

  private void Connect()
  {
    textBoxName.Text = UserName;
    buttonConnect.Text = "Сыграть";
    buttonConnect.Enabled = false;
    UdpClient client = new UdpClient();
    IPEndPoint   endPoint   =   new   IPEndPoint(IPAddress.
                          Parse("230.230.230.230"), 8005);
    try
    {
      Packet msg = new Packet(0, UserName);
      client.Send(msg.getBytes(),            msg.MsgLength,
                  endPoint);
      client.Close();
      textBoxName.Enabled = false;
      buttonConnect.Enabled = false;
      Connected = true;
      receiveUDP = new Task(() => ReceiveConnection());
      receiveUDP.Start();
    }
    catch (Exception ex)
    {
      client.Close();
      MessageBox.Show(ex.Message);
    }
  }
  private void ReceiveConnection()
  {
    try
    {
      while (Connected)
      {
        UdpClient client = new UdpClient(8005);
        client.JoinMulticastGroup(IPAddress.Parse
                              ("230.230.230.230"), 100);
        IPEndPoint remoteIp = null;
        byte[] data = client.Receive(ref remoteIp);
        Packet msg = new Packet(data);
        client.Close();
```

```
if (msg.Type == 0)
{
  Sub    sub    =    new    Sub(Encoding.Unicode.
                GetString(msg.Data), new IPEndPoint
                          (remoteIp.Address, 8006));
  if (!Subs.Contains(sub))
  {
    Subs.Add(sub);
    this.Invoke(new MethodInvoker(() =>
    {
      comboBox.Items.Add(sub.name);
    }));
  }
  msg = new Packet(1, UserName);
  UdpClient sender = new UdpClient();
  sender.Connect(sub.endPoint.Address, 8006);
  sender.Send(msg.getBytes(), msg.MsgLength);
  sender.Close();
}
else if (msg.Type == 1)
{
  Sub    sub    =    new    Sub(Encoding.Unicode.
                GetString(msg.Data), new IPEndPoint
                          (remoteIp.Address, 8006));
  if (!Subs.Contains(sub))
  {
    Subs.Add(sub);
    this.Invoke(new MethodInvoker(() =>
    {
      comboBox.Items.Add(sub.name);
    }));
  }
}
else if (msg.Type == 2)
{
  UdpClient reply = new UdpClient();
  reply.Connect(new    IPEndPoint(remoteIp.Address,
                                  8006));
  msg = new Packet(3, UserName);
  reply.Send(msg.getBytes(), msg.MsgLength);
  reply.Close();
  using (var game = new Game1(1, remoteIp))
    game.Run();
  Close();
}
else if (msg.Type == 3)
{
  using (var game = new Game1(2, remoteIp))
    game.Run();
  Close();
}
else
```

```
            MessageBox.Show("An error occured!");
          }
        }
        catch (Exception ex)
        {
          MessageBox.Show(ex.Message);
        }
      }

      public static string GetLocalIPAddress()
      {
        var host = Dns.GetHostEntry(Dns.GetHostName());
        foreach (var ip in host.AddressList)
        {
          if (ip.AddressFamily == AddressFamily.InterNetwork)
          {
            return ip.ToString();
          }
        }
        throw new Exception("No network adapters with an IPv4
                             address in the system!");
      }

      private     void     comboBox_SelectedIndexChanged(object
                                          sender, EventArgs e)
      {
        buttonConnect.Enabled = true;
      }
    }
}
```

Файл Game1.cs

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using System;
using System.Collections.Generic;
using System.Net;
using System.Net.Sockets;

namespace Tanks2D
{
  /// <summary>
  /// This is the main type for your game.
  /// </summary>
  public class Game1 : Game
  {
    GraphicsDeviceManager graphics;
    SpriteBatch spriteBatch;
    Texture2D PlayerCorp, PlayerTowr;
    Texture2D BrickT, ShotT;
```

```
        Texture2D Grn, GrnB;
        Texture2D Red, RedB;
        Vector2 TextureAttitude;
        Tank Player, Enemy;
        int PlayerTeam, EnemyTeam;
        bool ShotFired = false;
        IPEndPoint enemyIP;
        UdpClient listener;
        List<Brick> Bricks = new List<Brick>(144) { //Top
                                new Brick(0, 0, 500), new Brick(120,
0, 500), new Brick(240, 0, 500), new Brick(360, 0, 500),
                                new   Brick(480,   0,   500),   new
Brick(600, 0, 500), new Brick(720, 0, 500), new Brick(840, 0, 500),
                                new   Brick(960,   0,   500),   new
Brick(1080, 0, 500), new Brick(1200, 0, 500), new Brick(1320, 0,
500),
                                new   Brick(1440,   0,   500),   new
Brick(1560, 0, 500), new Brick(1680, 0, 500), new Brick(1800, 0,
500),
                                // Left
                                new Brick(0, 120, 500), new Brick(0,
240, 500), new Brick(0, 360, 500), new Brick(0, 480, 500),
                                new Brick(0, 600, 500), new Brick(0,
720, 500), new Brick(0, 840, 500),
                                //Player Left
                                new  Brick(480,  120,  300),  new
Brick(480, 240, 300), new Brick(480, 360, 300), new Brick(480,
480, 300), new Brick(480, 600, 300),
                                // Center
                                new  Brick(840,  360,  200),  new
Brick(840, 480, 200), new Brick(960, 480, 200), new Brick(960,
600, 200),
                                //Player Right
                                new  Brick(1320,  360,  300),  new
Brick(1320, 480, 300), new Brick(1320, 600, 300), new Brick(1320,
720, 300), new Brick(1320, 840, 300),
                                // Right
                                new  Brick(1800,  120,  500),  new
Brick(1800, 240, 500), new Brick(1800, 360, 500), new Brick(1800,
480, 500),
                                new  Brick(1800,  600,  500),  new
Brick(1800, 720, 500), new Brick(1800, 840, 500),
                                //Bottom
                                new   Brick(0,   960,   500),   new
Brick(120, 960, 500), new Brick(240, 960, 500), new Brick(360,
960, 500),
                                new  Brick(480,  960,  500),  new
Brick(600, 960, 500), new Brick(720, 960, 500), new Brick(840,
960, 500),
                                new  Brick(960,  960,  500),  new
Brick(1080, 960, 500), new Brick(1200, 960, 500), new Brick(1320,
960, 500),
```

```
                                new Brick(1440, 960, 500), new
Brick(1560, 960, 500), new Brick(1680, 960, 500), new Brick(1800,
960, 500)};

        public Game1(int input, IPEndPoint iep)
        {
          enemyIP = iep;
          PlayerTeam = input;
          EnemyTeam = 3 - input;
          graphics = new GraphicsDeviceManager(this);
          graphics.PreferredBackBufferWidth = GraphicsAdapter.De-
faultAdapter.CurrentDisplayMode.Width;
          graphics.PreferredBackBufferHeight     =     GraphicsA-
dapter.DefaultAdapter.CurrentDisplayMode.Height;
          graphics.IsFullScreen = true;
          var hAttitude = graphics.PreferredBackBufferHeight /
1080.0f;
          TextureAttitude = new Vector2(hAttitude, hAttitude);
          Content.RootDirectory = "Content";
        }

        /// <summary>
        /// Allows the game to perform any initialization it needs
to before starting to run.
        /// This is where it can query for any required services
and load any non-graphic
        /// related content.  Calling base.Initialize will enumer-
ate through any components
        /// and initialize them as well.
        /// </summary>
        protected override void Initialize()
        {
          // TODO: Add your initialization logic here

          base.Initialize();
        }

        /// <summary>
        /// LoadContent will be called once per game and is the
place to load
        /// all of your content.
        /// </summary>
        protected override void LoadContent()
        {
          // Create a new SpriteBatch, which can be used to draw
textures.
          spriteBatch = new SpriteBatch(GraphicsDevice);
          PlayerCorp = Content.Load<Texture2D>("MBTCorpSprite");
          PlayerTowr = Content.Load<Texture2D>("MBTTowrSprite");
          BrickT = Content.Load<Texture2D>("Brick");
          ShotT = Content.Load<Texture2D>("ShotSprite");
          Grn = Content.Load<Texture2D>("Grn");
```

```
            GrnB = Content.Load<Texture2D>("GrnB");
            Red = Content.Load<Texture2D>("Red");
            RedB = Content.Load<Texture2D>("RedB");
            Player = new Tank(500, 2, PlayerTeam, 110, new
Point(graphics.PreferredBackBufferWidth, graphics.PreferredBack-
BufferHeight));
            Enemy = new Tank(500, 2, EnemyTeam, 110, new
Point(graphics.PreferredBackBufferWidth, graphics.PreferredBack-
BufferHeight));
            listener = new UdpClient();
            listener.Connect(enemyIP);
            Packet msg;
            if (PlayerTeam == 2)
            {
              msg = new Packet(4);
              listener.Send(msg.getBytes(), msg.MsgLength);
            }
            else
            {
              do
              {
                IPEndPoint ip = null;
                msg = new Packet(listener.Receive(ref ip));
              }
              while (msg.Type != 4);
            }
            // TODO: use this.Content to load your game content here
        }

        /// <summary>
        /// UnloadContent will be called once per game and is the
place to unload
        /// game-specific content.
        /// </summary>
        protected override void UnloadContent()
        {
          // TODO: Unload any non ContentManager content here
        }

        /// <summary>
        /// Allows the game to run logic such as updating the
world,
        /// checking for collisions, gathering input, and playing
audio.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing
values.</param>
        protected override void Update(GameTime gameTime)
        {
          if    (Keyboard.GetState().IsKeyDown(Keys.Escape)    ||
Player.curHP <= 0 || Enemy.curHP <= 0)
            Exit();
```

```
          while (listener.Available > 0)
          {
            IPEndPoint ip = null;
            Packet receive = new Packet(listener.Receive(ref ip));
            if (receive.Type == 5)
            {
              Vector2 coord = new Vector2();
              float dir;
              float time;
              receive.GetShot(out coord, out dir, out time);
              Shot shot = new Shot();
              shot.direction = dir;
              shot.StartPosition = coord;
              shot.StartTime = time;
              Enemy.Shots.Add(shot);
            }
            else if (receive.Type == 6)
            {
              receive.GetCoord(out   Enemy.Position,   out   Enemy.CorpRotation, out Enemy.TowrRotation);
            }
          }
          if (Keyboard.GetState().IsKeyDown(Keys.W))
          {
            Vector2 LastPosition = Player.Position;
            Player.Position.Y  +=  (float)(Math.Sin(Player.CorpRotation + Math.PI / 2));
            Player.Position.X  +=  (float)(Math.Cos(Player.CorpRotation + Math.PI / 2));
            if   (Collisions(Enemy.Position,  Enemy.CorpRotation, 100, 88, PlayerCorp.Width / 2,
              Player.Position, Player.CorpRotation, 100, 88, PlayerCorp.Width / 2))
            {
              Player.Position = LastPosition;
            }
            foreach (Brick brick in Bricks)
            {
              if   (Collisions(Player.Position,  Player.CorpRotation, 100, 88, PlayerCorp.Width / 2, new Vector2(brick.Position.X + 60, brick.Position.Y + 60), 0, 60, 60, 60))
              {
                Player.Position = LastPosition;
              }
            }
          }
          if (Keyboard.GetState().IsKeyDown(Keys.S))
          {
            Vector2 LastPosition = Player.Position;
            Player.Position.Y  -=  (float)(Math.Sin(Player.CorpRotation + Math.PI / 2));
```

```
            Player.Position.X -= (float)(Math.Cos(Player.CorpRo-
tation + Math.PI / 2));
            if (Collisions(Enemy.Position, Enemy.CorpRotation,
100, 88, PlayerCorp.Width / 2,
              Player.Position, Player.CorpRotation, 100, 88, Play-
erCorp.Width / 2))
            {
              Player.Position = LastPosition;
            }
            foreach (Brick brick in Bricks)
            {
              if (Collisions(Player.Position, Player.CorpRota-
tion, 100, 88, PlayerCorp.Width / 2, new Vector2(brick.Position.X
+ 60, brick.Position.Y + 60), 0, 60, 60, 60))
              {
                Player.Position = LastPosition;
              }
            }
          }
          if (Keyboard.GetState().IsKeyDown(Keys.D))
          {
            float LastRotation = Player.CorpRotation;
            if (Keyboard.GetState().IsKeyDown(Keys.S))
              Player.CorpRotation -= 0.005f;
            else
              Player.CorpRotation += 0.005f;
            if (Collisions(Enemy.Position, Enemy.CorpRotation,
100, 88, PlayerCorp.Width / 2,
              Player.Position, Player.CorpRotation, 100, 88, Play-
erCorp.Width / 2))
            {
              Player.CorpRotation = LastRotation;
            }
            foreach (Brick brick in Bricks)
            {
              if (Collisions(Player.Position, Player.CorpRota-
tion, 100, 88, PlayerCorp.Width / 2, new Vector2(brick.Position.X
+ 60, brick.Position.Y + 60), 0, 60, 60, 60))
              {
                Player.CorpRotation = LastRotation;
              }
            }
          }
          if (Keyboard.GetState().IsKeyDown(Keys.A))
          {
            float LastRotation = Player.CorpRotation;
            if (Keyboard.GetState().IsKeyDown(Keys.S))
              Player.CorpRotation += 0.005f;
            else
              Player.CorpRotation -= 0.005f;
            if (Collisions(Enemy.Position, Enemy.CorpRotation,
100, 88, PlayerCorp.Width / 2,
```

```
            Player.Position, Player.CorpRotation, 100, 88, Play-
erCorp.Width / 2))
            {
              Player.CorpRotation = LastRotation;
            }
          }
          if   (Keyboard.GetState().IsKeyDown(Keys.E)   ||   Key-
board.GetState().IsKeyDown(Keys.Right))
            Player.TowrRotation += 0.01f;
          if   (Keyboard.GetState().IsKeyDown(Keys.Q)   ||   Key-
board.GetState().IsKeyDown(Keys.Left))
            Player.TowrRotation -= 0.01f;
          if (Keyboard.GetState().IsKeyDown(Keys.Space) && !Shot-
Fired)
            if (Player.Fired == 0)
            {
              ShotFired = true;
              Shot shot = new Shot(Player, gameTime);
              Player.Shots.Add(shot);
              Player.Fired = Player.Reload;
              Packet newShot = new Packet(5, shot.StartPosition,
shot.direction, shot.StartTime);
              listener.Send(newShot.getBytes(),
newShot.MsgLength);
            }
          if  (Keyboard.GetState().IsKeyUp(Keys.Space)  &&  Shot-
Fired)
          {
            ShotFired = false;
          }
          if (Player.Fired > 0)
            Player.Fired--;

          Packet  newCoord  =  new  Packet(6,  Player.Position,
Player.CorpRotation, Player.TowrRotation);
          listener.Send(newCoord.getBytes(), newCoord.MsgLength);
          // TODO: Add your update logic here
          base.Update(gameTime);
        }

        /// <summary>
        /// This is called when the game should draw itself.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing
values.</param>
        protected override void Draw(GameTime gameTime)
        {
          GraphicsDevice.Clear(Color.DarkOliveGreen);

          // TODO: Add your drawing code here
          spriteBatch.Begin();
```

```
          spriteBatch.Draw(PlayerCorp, new Vector2(Enemy.Posi-
tion.X, Enemy.Position.Y), null, null, new Vector2(Play-
erCorp.Width / 2, 100), Enemy.CorpRotation, TextureAttitude,
Color.Red);
          spriteBatch.Draw(PlayerCorp, new Vector2(Player.Posi-
tion.X, Player.Position.Y), null, null, new Vector2(Play-
erCorp.Width / 2, 100), Player.CorpRotation, TextureAttitude,
null);
          spriteBatch.Draw(PlayerTowr, new Vector2(Enemy.Posi-
tion.X, Enemy.Position.Y), null, null, new Vector2(Play-
erTowr.Width / 2, 80), Enemy.CorpRotation + Enemy.TowrRotation,
TextureAttitude, Color.Red);
          spriteBatch.Draw(PlayerTowr, new Vector2(Player.Posi-
tion.X, Player.Position.Y), null, null, new Vector2(Play-
erTowr.Width / 2, 80), Player.CorpRotation + Player.TowrRotation,
TextureAttitude, null);

          List<Shot> toRemove = new List<Shot>();
          foreach (Shot shot in Player.Shots)
          {
            if (Collisions(Enemy.Position, Enemy.CorpRotation,
100, 88, PlayerCorp.Width / 2,
              new Vector2((float)(shot.StartPosition.X -
(((float)gameTime.TotalGameTime.TotalMilliseconds - shot.Start-
Time) * 60 / 1000 + 25) * shot.speed * Math.Sin(shot.direction)),
                    (float)(shot.StartPosition.Y + (((float)ga-
meTime.TotalGameTime.TotalMilliseconds - shot.StartTime) * 60 /
1000 + 25) * shot.speed * Math.Cos(shot.direction))),
                     shot.direction, ShotT.Height / 2,
ShotT.Height / 2, ShotT.Width / 2))
            {
              Enemy.curHP -= shot.Dmg;
              toRemove.Add(shot);
            }
            foreach (Brick brick in Bricks)
            {
              if (Collisions(new Vector2((float)(shot.StartPosi-
tion.X - (((float)gameTime.TotalGameTime.TotalMilliseconds -
shot.StartTime) * 60 / 1000 + 25) * shot.speed * Math.Sin(shot.di-
rection)),
                              (float)(shot.StartPosition.Y +
(((float)gameTime.TotalGameTime.TotalMilliseconds - shot.Start-
Time) * 60 / 1000 + 25) * shot.speed * Math.Cos(shot.direction))),
                       shot.direction, ShotT.Height / 2,
ShotT.Height / 2, ShotT.Width / 2, new Vector2(brick.Position.X +
60, brick.Position.Y + 60), 0, 60, 60, 60))
              {
                brick.curHP -= shot.Dmg;
                toRemove.Add(shot);
              }
            }
          }
```

```
            foreach (Shot shot in toRemove)
            {
              Player.Shots.Remove(shot);
            }
            toRemove.Clear();

            foreach (Shot shot in Enemy.Shots)
            {
              if (Collisions(Player.Position, Player.CorpRotation,
100, 88, PlayerCorp.Width / 2,
                new     Vector2((float)(shot.StartPosition.X     -
(((float)gameTime.TotalGameTime.TotalMilliseconds  -  shot.Start-
Time) * 60 / 1000 + 25) * shot.speed * Math.Sin(shot.direction)),
                     (float)(shot.StartPosition.Y  +  (((float)ga-
meTime.TotalGameTime.TotalMilliseconds - shot.StartTime) * 60 /
1000 + 25) * shot.speed * Math.Cos(shot.direction))),
                      shot.direction,    ShotT.Height    /    2,
ShotT.Height / 2, ShotT.Width / 2))
              {
                Player.curHP -= shot.Dmg;
                toRemove.Add(shot);
              }
              foreach (Brick brick in Bricks)
              {
                if (Collisions(new Vector2((float)(shot.StartPosi-
tion.X   -   (((float)gameTime.TotalGameTime.TotalMilliseconds   -
shot.StartTime) * 60 / 1000 + 25) * shot.speed * Math.Sin(shot.di-
rection)),
                               (float)(shot.StartPosition.Y      +
(((float)gameTime.TotalGameTime.TotalMilliseconds  -  shot.Start-
Time) * 60 / 1000 + 25) * shot.speed * Math.Cos(shot.direction))),
                        shot.direction,    ShotT.Height    /    2,
ShotT.Height / 2, ShotT.Width / 2, new Vector2(brick.Position.X +
60, brick.Position.Y + 60), 0, 60, 60, 60))
                {
                  brick.curHP -= shot.Dmg;
                  toRemove.Add(shot);
                }
              }
            }
            foreach (Shot shot in toRemove)
            {
              Enemy.Shots.Remove(shot);
            }
            toRemove.Clear();

            foreach (Shot shot in Player.Shots)
              spriteBatch.Draw(ShotT,           new           Vec-
tor2((float)(shot.StartPosition.X   -   (((float)gameTime.TotalGa-
meTime.TotalMilliseconds - shot.StartTime) * 60 / 1000 + 25) *
shot.speed * Math.Sin(shot.direction)),
                               (float)(shot.StartPosition.Y      +
```

```
(((float)gameTime.TotalGameTime.TotalMilliseconds - shot.Start-
Time) * 60 / 1000 + 25) * shot.speed * Math.Cos(shot.direction))),
                    null, null, new Vector2(ShotT.Width / 2,
ShotT.Height / 2), shot.direction, TextureAttitude, null);
          foreach (Shot shot in Enemy.Shots)
            spriteBatch.Draw(ShotT,           new           Vec-
tor2((float)(shot.StartPosition.X - (((float)gameTime.TotalGa-
meTime.TotalMilliseconds - shot.StartTime) * 60 / 1000 + 25) *
shot.speed * Math.Sin(shot.direction)),
                              (float)(shot.StartPosition.Y      +
(((float)gameTime.TotalGameTime.TotalMilliseconds - shot.Start-
Time) * 60 / 1000 + 25) * shot.speed * Math.Cos(shot.direction))),
                    null, null, new Vector2(ShotT.Width / 2,
ShotT.Height / 2), shot.direction, TextureAttitude, null);
          List<Brick> toRem = new List<Brick>();
          foreach (Brick brick in Bricks)
          {
            if (brick.curHP > 0)
              spriteBatch.Draw(BrickT,            brick.Position,
Color.White);
            else
              toRem.Add(brick);
          }
          foreach(Brick brick in toRem)
          {
            Bricks.Remove(brick);
          }
          toRem.Clear();

          spriteBatch.Draw(GrnB,     new     Vector2(20,     20),
Color.White);
          spriteBatch.Draw(Grn, new Rectangle(new Point(20, 20),
new Point(Grn.Width * Player.curHP / 500, 56)), Color.White);
          spriteBatch.Draw(RedB,    new    Vector2(1606,    20),
Color.White);
          spriteBatch.Draw(Red, new Rectangle(new Point(1606, 20),
new Point(Red.Width * Enemy.curHP / 500, 56)), new Rectangle(new
Point(0,  0),  new  Point(Red.Width  *  Enemy.curHP  /  500,  56)),
Color.White);

          spriteBatch.End();

          base.Draw(gameTime);
        }

        public  bool  Collisions(Vector2  firstPosition,  float
firstAngle, int firstForw, int firstBack, int firstSide, Vector2
secondPosition, float secondAngle, int secondForw, int secondBack,
int secondSide)
        {
          Point[] firstPoints = new Point[4];
          Point[] secondPoints = new Point[4];
```

```
          firstPoints[0] = new Point((int)Math.Round(firstPosi-
tion.X + firstSide / Math.Cos(firstAngle) + (firstForw - firstSide
*       Math.Tan(firstAngle))       *       Math.Sin(firstAngle)),
(int)Math.Round(firstPosition.Y  +  (firstForw  -  firstSide  *
Math.Tan(firstAngle)) * Math.Cos(firstAngle)));
          firstPoints[1] = new Point((int)Math.Round(firstPosi-
tion.X + firstSide / Math.Cos(firstAngle) + (firstForw - firstSide
*       Math.Tan(firstAngle))       *       Math.Sin(firstAngle)),
(int)Math.Round(firstPosition.Y - firstBack * Math.Cos(firstAn-
gle) + firstSide * Math.Sin(firstAngle)));
          firstPoints[2] = new Point((int)Math.Round(firstPosi-
tion.X - firstSide / Math.Cos(firstAngle) - (firstBack - firstSide
*       Math.Tan(firstAngle))       *       Math.Sin(firstAngle)),
(int)Math.Round(firstPosition.Y - firstBack * Math.Cos(firstAn-
gle) + firstSide * Math.Sin(firstAngle)));
          firstPoints[3] = new Point((int)Math.Round(firstPosi-
tion.X - firstSide / Math.Cos(firstAngle) - (firstBack - firstSide
*       Math.Tan(firstAngle))       *       Math.Sin(firstAngle)),
(int)Math.Round(firstPosition.Y  +  (firstForw  -  firstSide  *
Math.Tan(firstAngle)) * Math.Cos(firstAngle)));
          secondPoints[0] = new Point((int)Math.Round(secondPosi-
tion.X + secondSide / Math.Cos(secondAngle) + (secondForw - sec-
ondSide  *  Math.Tan(secondAngle))  *  Math.Sin(secondAngle)),
(int)Math.Round(secondPosition.Y  +  (secondForw  -  secondSide  *
Math.Tan(secondAngle)) * Math.Cos(secondAngle)));
          secondPoints[1] = new Point((int)Math.Round(secondPosi-
tion.X + secondSide / Math.Cos(secondAngle) + (secondForw - sec-
ondSide  *  Math.Tan(secondAngle))  *  Math.Sin(secondAngle)),
(int)Math.Round(secondPosition.Y  -  secondBack  *  Math.Cos(sec-
ondAngle) + secondSide * Math.Sin(secondAngle)));
          secondPoints[2] = new Point((int)Math.Round(secondPosi-
tion.X - secondSide / Math.Cos(secondAngle) - (secondBack - sec-
ondSide  *  Math.Tan(secondAngle))  *  Math.Sin(secondAngle)),
(int)Math.Round(secondPosition.Y  -  secondBack  *  Math.Cos(sec-
ondAngle) + secondSide * Math.Sin(secondAngle)));
          secondPoints[3] = new Point((int)Math.Round(secondPosi-
tion.X - secondSide / Math.Cos(secondAngle) - (secondBack - sec-
ondSide  *  Math.Tan(secondAngle))  *  Math.Sin(secondAngle)),
(int)Math.Round(secondPosition.Y  +  (secondForw  -  secondSide  *
Math.Tan(secondAngle)) * Math.Cos(secondAngle)));

          for (int i = 0; i < 4; i++)
            for (int j = 0; j < 4; j++)
            {
              int i2 = (i + 1) % 4;
              int j2 = (j + 1) % 4;
              if   (Collision(firstPoints[i],   firstPoints[i2],
secondPoints[j], secondPoints[j2]))
                return true;
            }
          return false;
        }
```

```
        public bool Collision(Point F1, Point F2, Point S1, Point
S2)
        {
          if (F2.X < F1.X)
          {
            Point tmp = F1;
            F1 = F2;
            F2 = tmp;
          }
          if (S2.X < S1.X)
          {

            Point tmp = S1;
            S1 = S2;
            S2 = tmp;
          }
          if ((Math.Max(F1.X, F2.X) < Math.Min(S1.X, S2.X)) ||
(Math.Min(F1.X, F2.X) > Math.Max(S1.X, S2.X)) || (Math.Max(F1.Y,
F2.Y) < Math.Min(S1.Y, S2.Y)) || (Math.Min(F1.Y, F2.Y) >
Math.Max(S1.Y, S2.Y)))
          {
            return false;
          }
          if ((F2.X - F1.X == 0) && (S2.X - S1.X == 0))
          {
            if (F1.X == S1.X)
              return true;
            else
              return false;
          }
          if (F2.X - F1.X == 0)
          {
            double A = (S2.Y - S1.Y) / (S2.X - S1.X);
            double b = S1.Y - A * S1.X;
            double crossX = F1.X;
            double crossY = A * crossX + b;

            if (S1.X <= crossX && S2.X >= crossX && Math.Min(F1.Y,
F2.Y) <= crossY && Math.Max(F1.Y, F2.Y) >= crossY)
              return true;
            else
              return false;
          }
          if (S2.X - S1.X == 0)
          {
            double A = (F2.Y - F1.Y) / (F2.X - F1.X);
            double b = F1.Y - A * F1.X;
            double crossX = S1.X;
            double crossY = A * crossX + b;

            if (F1.X <= crossX && F2.X >= crossX && Math.Min(S1.Y,
S2.Y) <= crossY && Math.Max(S1.Y, S2.Y) >= crossY)
```

```
            return true;
          else
            return false;
        }

        double FA = (F2.Y - F1.Y) / (F2.X - F1.X);
        double SA = (S2.Y - S1.Y) / (S2.X - S1.X);
        double Fb = F1.Y - FA * F1.X;
        double Sb = S1.Y - SA * S1.X;
        if (FA == SA)
        {
          if (Fb == Sb)
            return true;
          return false;
        }
        else
        {
          double crossX = (Sb - Fb) / (SA - FA);
          if ((crossX < Math.Max(F1.X, S1.X)) || (crossX >
Math.Min(F2.X, S2.X)))
          {
            return false;
          }
          else
          {
            return true;
          }
        }
      }
    }

    public class Tank
    {
      public int curHP;
      public int maxHP;
      public int Speed;
      public Vector2 Position;
      public float CorpRotation;
      public float TowrRotation;
      public List<Shot> Shots;
      public int Fired;
      public int Reload;

      public Tank(int HP, int Sp, int Pl, int Rld, Point Field)
      {
        curHP = maxHP = HP;
        Speed = Sp;
        Reload = Rld;
        Fired = 0;
        Shots = new List<Shot>();
        switch (Pl)
        {
```

```
            case 1:
              CorpRotation = (float)(-Math.PI / 2);
              TowrRotation = (float)(0);
              Position = new Vector2(250, Field.Y / 2);
              break;
            case 2:
              CorpRotation = (float)(Math.PI / 2);
              TowrRotation = (float)(0);
              Position = new Vector2(Field.X - 250, Field.Y / 2);
              break;
          }
        }
      }

      public class Shot
      {
        public int Dmg = 100;
        public float speed = 5;
        public float direction;
        public Vector2 StartPosition;
        public float StartTime;

        public Shot(Tank Source, GameTime curTime)
        {
          StartPosition.X = Source.Position.X;
          StartPosition.Y = Source.Position.Y;
          direction = Source.CorpRotation + Source.TowrRotation;
          StartTime = (float)curTime.TotalGameTime.TotalMillisec-
onds;
        }
        public Shot()
        {
        }
      }

      public class Brick
      {
        public int curHP;
        public int maxHP;
        public Vector2 Position;

        public Brick(float X, float Y, int HP)
        {
          curHP = maxHP = HP;
          Position = new Vector2(X, Y);
        }
      }

      internal class Packet
      {
        public byte Type;
        public UInt16 MsgLength;
```

```
        public byte[] Data;

        public Packet(int T, Vector2 Pos, float corpRot, float
towrRot)
        {
          if (T == 6 || T == 5)
          {
            Type = 6;
            byte[] data = new byte[sizeof(float) * 4];
            MsgLength = 3 + sizeof(float) * 4;
            Buffer.BlockCopy(BitConverter.GetBytes(Pos.X),     0,
data, 0, sizeof(float));
            Buffer.BlockCopy(BitConverter.GetBytes(Pos.Y),     0,
data, sizeof(float), sizeof(float));
            Buffer.BlockCopy(BitConverter.GetBytes(corpRot),   0,
data, sizeof(float) * 2, sizeof(float));
            Buffer.BlockCopy(BitConverter.GetBytes(towrRot),   0,
data, sizeof(float) * 3, sizeof(float));
            Data = new byte[MsgLength];
            Data = data;
          }
        }
        public Packet(byte[] D)
        {
          Type = D[0];
          MsgLength = BitConverter.ToUInt16(D, 1);
          Data = new byte[MsgLength - 3];
          Buffer.BlockCopy(D, 3, Data, 0, MsgLength - 3);
        }
        public Packet(byte T)
        {
          Type = T;
          MsgLength = 3;
        }
        public byte[] getBytes()
        {
          byte[] data = new byte[MsgLength];
          Buffer.BlockCopy(BitConverter.GetBytes(Type),  0,  data,
0, 1);
          Buffer.BlockCopy(BitConverter.GetBytes(MsgLength),   0,
data, 1, 2);
          if (MsgLength > 3)
            Buffer.BlockCopy(Data, 0, data, 3, MsgLength - 3);
          return data;
        }
        public void GetCoord(out Vector2 Pos, out float corpRot,
out float towrRot)
        {
          Pos   =   new   Vector2(BitConverter.ToSingle(Data,   0),
BitConverter.ToSingle(Data, sizeof(float)));
          corpRot  =  BitConverter.ToSingle(Data,  sizeof(float)  *
2);
```

```
          towrRot = BitConverter.ToSingle(Data, sizeof(float) *
3);
        }
        public void GetShot(out Vector2 Pos, out float dir, out
float time)
        {
          Pos = new Vector2(BitConverter.ToSingle(Data, 0),
BitConverter.ToSingle(Data, sizeof(float)));
          dir = BitConverter.ToSingle(Data, sizeof(float) * 2);
          time = BitConverter.ToSingle(Data, sizeof(float) * 3);
        }
      }
    }
```

ВЕДОМОСТЬ ДОКУМЕНТОВ

Обозначение	Наименование	Дополнительные сведения
	Текстовые документы	
БГУИР КП 1–40 01 01 614 ПЗ	Пояснительная записка	45
	Графические документы	
ГУИР.851006-014 СА	Схема алгоритма	Формат А1

					БГУИР КП I- 40 01 01 614 ПЗ		
Из	Лист	№ докум.	Подп.	Дата			
Разраб.		Крот Е.Д.			Программное средство Игра «Packet Tanks 2D» Ведомость курсового проекта	Литера: Т	Лист: 45 / Листов: 45
Провер.		Шимко И.В.				Кафедра ПОИТ гр. 851006	