**Министерство образования и науки Российской Федерации**

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт Кибернетики

Направление подготовки (специальность) Информационные системы и технологии

Кафедра Вычислительной техники

**Танчики**

**Отчет по дисциплине**

**Учебная практика**

Выполнил студент гр. 8И42 \_\_\_\_\_\_\_\_\_\_\_\_\_ Хурсевич Д. К.

(Номер группы) (Подпись) (Ф.И.О.)

Дата сдачи отчета преподавателю \_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ 2016г.

Руководитель Старший преподаватель кафедры ВТ Лепустин А. В.

(Ученая степень, ученое звание, должность) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Оценка руководителя) (Подпись)

\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ 2016г.

(Дата проверки)

Учебную практику выполнил и защитил с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члены комиссии: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ 2016г.

(Дата защиты )

Томск 2016 г.

Оглавление

[1. Цель работы](#_Toc462039444) 3

[2. Задачи](#_Toc462039445) 3

[3. Техническое задание](#_Toc462039446) 4

[Введение](#_Toc462039447) 5

[Наименование программы:](#_Toc462039448) 5

[Область применения:](#_Toc462039449) 5

[3.1. Основание для разработки](#_Toc462039450) 5

[3.1.1. Документ на основании, которого видеться разработка:](#_Toc462039451) 5

[3.1.2. Организация, утвердившая документ:](#_Toc462039452) 5

[3.2. Назначение разработки](#_Toc462039453) 5

[3.3. Требования к разработке](#_Toc462039454) 5

[3.3.1. Требования к функциональным характеристикам](#_Toc462039455) 5

[3.3.2. Требования к игровому процессу](#_Toc462039456) 6

[3.3.3. Требования к надежности](#_Toc462039457) 10

[3.3.4. Требования к составу и параметрам технических:](#_Toc462039458) 10

[3.3.5. Требования к информационной программной совместимости:](#_Toc462039459) 10

[3.4. Требования к программной документации](#_Toc462039460) 10

[3.5. Стадии и этапы разработки](#_Toc462039462) 10

[3.6. Порядок контроля и приемки](#_Toc462039463) 11

[4. Описание структуры программы](#_Toc462039464) 12

[5. Описание алгоритмов](#_Toc462039465) 22

[5.1. Поведение танков компьютера](#_Toc462039466) 22

[5.2. Метод Tracker](#_Toc462039467) 22

[6. Руководство пользователя](#_Toc462039468) 24

[7. Заключение](#_Toc462039469) 30

[Приложение](#_Toc462039470) 31

# **Цель работы**

Целью работы является получение теоретических знаний и практических навыков по разработке приложений на языке высокого уровня C#, соответствующих парадигме ООП. Получение навыков проектирования архитектуры приложений и составления технического задание к ним.

# **Задачи**

* Написание технического задания будущего приложения;
* Проектирование архитектуры приложения;
* Программная реализация алгоритмов приложения;
* Реализация интерфейса приложения;
* Составление отчета по результатам учебной практики.

# **Техническое задание**

**Министерство образования и науки Российской Федерации**

Федеральное государственное автономное образовательное учреждение

высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт Кибернетики

Направление Информационные системы и технологии

Кафедра Вычислительной техники

|  |  |
| --- | --- |
|  |  |
| Компьютерная игра «Танчики»  наименование вида ИС  «Танчики»  Сокращенное наименование ИС  Техническое задание | |
| **СОГЛАСОВАНО**  Должность  Старший преподаватель кафедры ВТ Лепустин А.В.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2016 г. | **РАЗРАБОТЧИК**  Должность  Студент гр.8И42  Хурсевич Д.К.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2016 г. |
| Томск 2016 | |

# **Введение**

# **Наименование программы:**

Компьютерная игра «Танчики».

# **Область применения:**

Программа предназначена для людей любого возраста, для игры на персональном компьютере.

# **Основание для разработки**

# **Документ на основании, которого видеться разработка:**

Задание на учебную практику.

# **Организация, утвердившая документ:**

Национальный исследовательский томский политехнический университет.

# **Назначение разработки**

Назначение разработки предполагает создание аркадной игры «Танчики», для одного или двух игроков.

Игрок, управляя танком, должен уничтожить на каждом уровне все танки противника. Танки компьютера в свою очередь пытаются уничтожить игрока и его штаб.

Всего в игре есть 35 уровней, на каждом из которых можно встретить различные типы преград и ландшафта. Так же в игре реализовано 4 типа вражеских танков и 6 бонусов, которые появляются в случайном месте на карте после уничтожения бонусного танка.

# **Требования к разработке**

# **Требования к функциональным характеристикам**

При запуске игры программа должна начинать свою работу с отображения главного меню, которое должно предоставлять игроку возможность начать одиночную игру, игру с напарником, перейти к таблице рекордов, выйти из игры.

Таблица рекордов должна состоять из двух колонок. В первой колонке должны быть написаны имена игроков, а во второй количество очков набранные игроками. Результаты должны сортироваться по правилу: чем больше очков набрал игрок, тем выше в списке он находится. Всего в таблице должны находиться 10 лучших результатов. Так же у пользователя должна быть возможность очистить список рекордов или вернуться назад в главное меню.

При попытке выхода из игры должно выходить окно, в котором у пользователя уточняется его желание выйти. При положительном ответе, программа должна закрывать игру, иначе пользователь должен возвращаться к главному меню.

# **Требования к игровому процессу**

Игровое пространство должно представлять собой поле 13х13 секторов (каждый сектор представляет определенный тип ландшафта); счетчик вражеских танков (счетчик показывает танки, которые остались у компьютера не считая тех, которые находятся на поле, всего 20 на уровень); счетчик жизней игрока (3 для каждого игрока на игру); номер уровня (всего в игре 35 уровней).

Игрок, управляя танком, должен уничтожить на каждом уровне все танки противника, которые появляются в верхней части экрана. Танки компьютера в свою очередь пытаются уничтожить игрока и его штаб (обозначается на карте в виде орла).

При выборе в главном меню начала новой игры с напарником, каждому из игроков предоставляется собственный танк.

При прохождении последнего уровня, уничтожении штаба или при отсутствии жизней у обоих игроков игра должна заканчиваться и подсчитывать очки игрока. Далее, показывать таблицу рекордов, и если игрок набрал достаточное количество очков, предоставлять ему возможность вписать свое имя.

Если все танки компьютера на уровне уничтожены, то должен происходить подсчет очков игрока и происходить переход на следующий уровень.

При переходе на следующий уровень должен сохраняться уровень танка игрока; если у одного из игроков в командной игре закончились жизни, то на следующий уровень он не переходит.

Танки должны иметь возможность передвигаться влево, вверх, вниз и вправо. Одновременно лишь в одном направлении

Танк первого игрока должен управляться клавишами A, W, S, D (движение влево, вверх, вниз и вправо соответственно) и стрелять при нажатии клавиши G.

Танк второго игрока должен управляться клавишами ←, ↑, ↓, → (движение влево, вверх, вниз и вправо соответственно) и стрелять при нажатии клавиши Enter.

При попадании снаряда одного игрока по другому, второй должен терять управление движением на 3 секунды. Снаряд, выпущенный одним танком компьютера, должен пролетать сквозь другие танки компьютера. При столкновении снарядов они должны уничтожаться.

У игрока во время игры должна быть возможность поставить игру на паузу и выйти в главное меню.

В игре должны быть реализованы четыре типа вражеских танков, которые различаются скоростью, прочностью и количеством очков, получаемых за их уничтожение:

* Обычный танк (100 очков);
* Бронетранспортёр с удвоенной скоростью хода (200 очков);
* Скорострельный танк с удвоенной скоростью полета снаряда (300 очков);
* Тяжёлый танк, уничтожить который можно четырьмя попаданиями, способен уничтожать бетонные стены (танк меняет цвет в зависимости от оставшейся прочности) (400 очков).

Каждый танк должен стрелять не чаще, чем раз в 2 секунды. Танк компьютера должны производить выстрел в этом промежутке с 50% вероятностью, если на его пути нет игрока или штаба, иначе танк должен стрелять с максимальной частотой (выстрел каждые 2 секунды).

Все танки (кроме Бронетранспортера) должны проезжать за 1 секунду 3 сектора игрового поля.

У четвёртого, одиннадцатого и восемнадцатого танков на уровне должна быть переливающаяся окраска. Если игрок попадает в такой танк, то на поле в случайном месте появляется бонус. Взятие бонуса должно приносить 500 очков. Одновременно на поле может находиться только 1 бонус (при появлении нового бонуса предыдущий бонус, если его не взяли до этого момента, исчезает с поля).

У танка игрока должно быть 4 уровня, который увеличивается при взятии определенного бонуса:

* 1 уровень – Обычный танк с минимальной скоростью полета снаряда;
* 2 уровень – Танк с удвоенной скоростью полета снаряда;
* 3 уровень – Танк с возможностью стрелять очередями: 3 снаряда в очереди (Танк может стрелять 3 раза в 2 секунды);
* 4 уровень – Танк способный пробивать бетон.

При уничтожении танка игрока, новый танк (если у игрока не закончились жизни) появляется первым уровнем.

На поле должны находиться различные типы преград и ландшафта:

* Кирпичная стена. Танку 1, 2 и 3го уровня должно быть нужно сделать 2 выстрела для уничтожение стены (после первого выстрела стена, по который был произведен выстрел, должна менять цвет); для танка 4го уровня 1 выстрел;
* Бетонная стена. Возможность разрушить у игрока должна быть только при наличии танка 4го уровня;
* Кусты. Должны снижать видимость бронетехники и снарядов (прорисовываться поверх танков);
* Лед. Должен снижает управляемость передвижения танка: при движении по льду после отпускания кнопки движения танк должен продолжать двигаться в текущем направлении по льду;
* Вода. Должна блокировать передвижение, но пропускать снаряды.

В игре должны быть реализованы следующие бонусы:

* *«Танк»*. Должен прибавлять игроку одну жизнь;
* *«Звезда»*. Должен улучшать танк игрока на 1 уровень;
* *«Бомба».* Должен взрывать танки противника на карте, за их уничтожение очки не должны начисляться — только 500 очков за взятие бонуса как такового;
* *«Часы»*. Должен останавливать врагов и их стрельбу на 5 секунд;
* *«Лопата»*. Должен на 20 секунд делать кирпичную стену штаба бетонной. После прекращения действия вокруг штаба восстанавливается кирпичная стена;
* *«Каска»*. Должен делать танк игрока неуязвимым на 10 секунд.

При игре в одиночном режиме на карте должно находиться не более четырёх танков противника одновременно; при игре вдвоём не более шести. Танки должны появляться один за другим через задержку, зависящую от уровня и количества игроков, которая вычисляется по формуле (3.1).

2 \* (190 - level \* 4 - (player\_count - 1) \* 20) / 60, (3.1)

где

Level – номер уровня;

Player\_count – количество игроков.

Должно существовать три периода поведения танков: сначала они должны двигаться в случайных направлениях, затем в направлении игроков, и после в направлении штаба. Длительность первых двух периодов должна быть одинакова и равна 3 задержкам появления. Последний период должен длиться до уничтожении танка.

# **Требования к надежности**

Программа должна проверять корректность ввода имени игрока в таблице рекордов: Имя должно быть не более 10 символов и содержать буквы только английского алфавита и цифры.

# **Требования к составу и параметрам технических:**

* Процессор с частотой 2ГГц и более;
* Оперативная память 1Гб и более;
* Объем свободного места на жестком диске: 200мб;
* Клавиатура;
* Монитор.

# **Требования к информационной программной совместимости:**

* Операционная система Microsoft Windows7 или новее;
* Microsoft .NET framework 3.5.

# **Требования к программной документации**

Состав программной документации:

* Техническое задание;
* Описание архитектуры приложения;
* Руководство пользователя.

# **Стадии и этапы разработки**

* Написание технического задания (27.06.16 – 29.06.16);
* Проектирование архитектуры приложения (29.06.16 – 08.07.16);
* Реализация приложения (08.07.16 – 15.07.16);
* Тестирование и отладка (15.07.16 – 20.07.16);
* Написание руководства пользователя (20.07.16 – 27.07.16).

# **Порядок контроля и приемки**

Приемка должна осуществляться в соответствии с документами "Техническое задание".

# **Описание структуры программы**

Игровые меню в игре представлены статическим классом Menu. Этот класс реализует методы, добавления и удаление кнопок в текущее меню, а также событие нажатие кнопки (ButtonPress). Сами кнопки представлены отдельными объектами класса Button с обработчиком события нажатия.

Таблица рекордов реализована через статический класс TableRecords, с функциями добавления рекорда, сохранения и очистки таблицы. Сам рекорд представлен структурой Record с полями name и score. Структура реализует интерфейс IComparable для возможностями сортировки в массивах и листах встроенной функцией Sort().

Для описания логики переходов между уровнями или завершения игры, запуска новой игры, выхода из игры, хранения игровой информацией (текущий уровень, счет игроков, количество танков, списки(List) с игровыми объектами и тд), логики уничтожения и создания игровых объектов, а также взаимодействия между ними был реализован статический класс GameManager.

Абстрактный класс GameObject является базовым для любого игрового объекта. Класс содержит абстрактную функцию отрисовки, поле с координатами, и функцию-обработчик столкновения игровых объектов.

Действие игры происходит на двухмерном поле 13 на 13 секторов. Сектор может быть или пустым, или представлять собой один из игровых объектов. элемент ландшафта (объекты классов Water, Ice и Shrub) или стену (объекты класса Wall).

Игрок-(и) и компьютер управляют танками, которые представлены объектами со своими характеристиками (скорость передвижения, скорость стрельбы, уровень, количество здоровья и тд.). Для реализации таких объектов был создан базовый абстрактный класс Tank, который хранить общие для танков компьютера и игрока поля и методы, а также событие передвижения по полю (TankMove). Поскольку логика стрельбы и передвижения, а также отрисовка у танков компьютера и игрока разная, то было создано два класса PlayerTank и EnemyTank, которые являются производными от класса Tank, в которых переопределены методы его методы.

Для реализации стрельбы был создан класс Bullet(пуля), объекты которого при создании перемещаются по игровому полю от выстрелившего танка до одно из игровых объектов (стена, танк или другой объект класса Bullet). Во время движения пули при каждом ее шаге происходит событие BulletMove, на которое подписаны все игровые объекты, с которыми пуля может столкнуться.

Так же в игре реализован бонусный объект (класс Bonus), появляющийся на поле после уничтожения определенного танка и уничтожающийся при перемещении танка игрока в его координаты. После уничтожения бонуса срабатывает одна из функций (определяемая типом бонуса), которая реализует эффект бонуса.

Проверка нахождения танка в координатах бонусного объекта производиться через событие TankMove класса Tank. Аналогично проверяется нахождение танка на объектах класса Ice (находясь на таких объектах танк должен начинать скользить).

**GameManager**

Класс GameManager реализует в игре логику перехода между окнами меню, игровыми уровнями, отрисовки игрового процесса, создания и уничтожения игровых объектов; обрабатывает события столкновения игровых объектов.

Методы класса:

* void ON(Form1 \_form) – Вызывается при загрузке формы приложения. Задает ссылку на форму соответствующему полю класса, задает значения для таймеров и полей статических классов, вызывает функцию отрисовки главного меню (MainMenu());
* void StartGame(Boolean two = false) – Вызывается при начале новой игры для одного (two=false) или двух игроков (two=true). Создает на поле штаб и границы игрового поля;
* void Nextlvl() – Считывает из файла и создает игровое поле соответствующее текущему уровню.
* static void Clearlvl(bool fully = false) - Очищает уровень от игровых объектов, кроме штаба и границ игрового поля (при fully=true уничтожаются и штаб с границами игрового поля);
* static void Changelvl() – Вызоывает при прохождении игрового уровня;
* static void GameOver() – Вызывается при проигрыше – у всех игроков закончились жизни или штаб был уничтожен;
* static void Interrupt() – Вызывается при прерывании игры (клавиша “Esc”);
* void Go\_To\_TR() – Вызывается после поражения или прохождения последнего уровня. Определяет достаточно ли очков набрали игроки для занесения их в таблицу рекордов и переносит игрока к таблице рекордов;
* void MainMenu() – создает главное меню игры;
* void Menu\_1Player() – Обработчик нажатия на кнопку одиночной игры;
* void Menu\_2Players() – Обработчик нажатия на кнопку игры для 2 игроков;
* void Menu\_Records() – Обработчик нажатия на кнопку таблицы рекордов;
* void Menu\_Exit() – Обработчик нажатия на кнопку выхода из игры;
* void NewTank() – Создает новый танк компьютера;
* void NewBullet(Tank tank) – Создает новую пулю;
* void NewBonus() – Создает новый бонус;
* void Collide(Bullet bullet, object \_object) – Обработчик столкновения игровых объектов: пуля с пулей, пуля с стеной, пуля с танком;
* void Hit\_EnemyTank(EnemyTank tank, Bullet bullet) – Обработчик попадания по танку компьютера;
* void Hit\_Player(PlayerTank tank, Bullet bullet) – Обработчик попадания по игроку;
* void Hit\_Wall(Wall block, Bullet bullet) – Обработчик попадания по стене;
* void RemoveBullet(Bullet bullet) – уничтожает пулю;
* void RemoveWall(Wall block) – Уничтожает стену;
* void Remove\_EnemyTank(EnemyTank tank, Bullet bullet) – Уничтожает танк компьютера;
* void RemoveIce(Ice \_ice) – Уничтожает блок со льдом;
* void RemoveLandScape(Block landscape) – уничтожает блок с ландшафтом (трава или вода);
* void RemoveBonus() – Уничтожает бонус;
* void RemovePlayer(PlayerTank tank) – Уничтожает танк игрока;
* void UseBonus(Bonus bonus, Tank tank) – Обработчик бонуса для события движения танка игрока;
* void Bonus\_Beton(object HP) – Заменяет стены возле штаба на бетоные на 20 секунд;
* void Bonus\_Set\_Shield(PlayerTank tank), void Bonus\_Destroy\_Shield( object empty) – Делают танк неуязвимым на 10 секунд;
* void Bonus\_Bomb() – Уничтожает все танки компьютера, которые находятся на поле;
* void Bonus\_lvl(PlayerTank tank) – Повышает уровень игрока;
* void Bonus\_Live(Tank tank) – Добавляет игроку 1 жизнь;
* void Bonus\_TimeStop() – Останавливает движение, стрельбу и появления танков компьютера;
* Cordinates Place() – Находит координаты свободной ячейки на игровом поле;
* static void OnArea(Object \_object, Tank tank) – Обработчик движение танка по объекту ландшафта;
* void OnPause() – Запуск/остановка(пауза) игры;
* void OnIce(object \_object, Tank tank) – Обработчик движения танка по льду;
* void Under\_Bush() – Вызывается при создании танков и пуль. Ставит обработчик отрисовки кустов на карте последнем в очереди всех обработчиков события Paint.
* void Global\_Timer\_Tick(object sender, EventArgs e) – Обработчик игрового таймера игры. Проверяет завершен ли уровень, следует ли стрелять танкам компьютера и нужно ли создать на поле новый танк;
* void StopTime\_Tick(object sender, EventArgs e) – Обработчик таймера для остановки танков компьютера (метод Bonus\_TimeStop());
* void pause\_tick(object sender, EventArgs e) – Обработчик таймера паузы в игре;
* void Player1Move\_keydown(object sender, KeyEventArgs e), void Player1Move\_keyup(object sender, KeyEventArgs e) – Обработчики движения 1 игрока;
* void Player2Move\_keydown(object sender, KeyEventArgs e), void Player2Move\_keyup(object sender, KeyEventArgs e) – Обработчики движения 2 игрока;
* void Press\_Interrupt(object sender, KeyEventArgs e) – Обработчик нажатия кнопок для прерывания игрового процесса или установки игры на паузу;
* void Press\_Nextlvl(object sender, KeyEventArgs e) – Обработчик нажатия кнопок при подсчете очков при переходе на следующий уровень;
* void Press\_WriteName(object sender, KeyPressEventArgs e) – Обработчик нажатия кнопок при вводе имени игрока для записи его в таблицу рекордов;
* void Draw\_Mainmenu(object sender, PaintEventArgs e) – Обработчик прорисовки главного меню;
* void Draw\_Game(object sender, PaintEventArgs e) – Обработчик прорисовки игрового процесса (текущий уровень, количество танков компьютера и жизней игроков);
* void Draw\_Between\_lvl(object sender, PaintEventArgs e) – Обработчик прорисовки окна подсчета очков игроков при переходе на новый уровень;
* void Draw\_Records(object sender, PaintEventArgs e) – Обработчик прорисовки таблицы рекордов;
* void Draw\_Writename(object sender, PaintEventArgs e) – Обработчик прорисовки набираемого имени в таблице рекордов;
* void Draw\_Exit(object sender, PaintEventArgs e) – Обработчик отрисовки окна уточнения желания покинуть игру;
* void Draw\_Pause(object sender, PaintEventArgs e) – Обработчик отрисовки паузы в игре;

**Menu**

Menu реализует игровые меню, состоящие из кнопок (класс Button).

Методы класса:

* static void Add(Button button) – Добавляет кнопку в текущее меню;
* static void Remove(string title) – Удаляет указанную кнопку с текущего меню;
* static void Remove() – Удаляет все кнопки с текущего меню;
* static void Sort() – Сортирует список с кнопками текущего меню по координате Y;
* int Get\_focus() – Определяет индекс кнопки, которая находиться в фокусе;
* static void OnPress(object sender, KeyEventArgs e) – Обработчик нажатия клавиш клавиатуры для взаимодействия с меню;

**Button**

Button реализует кнопки в игровых меню.

Методы класса:

* void press\_handler() – Обработчик события нажатия одной из кнопок меню;
* void Draw (object sender, PaintEventArgs e) – Обработчик прорисовки кнопки;

**Coordinates**

Coordinates – структура для координат X и Y. Содержит два поля для значений соответствующих координат.

**GameObject**

GameObject – абстрактный класс базовый для любого игрового объекта.

Методы класса:

* abstract void Draw(object sender, PaintEventArgs e) – Обработчик отрисовки;
* void SearchObject(GameObject obj) – Обработчик события движения игровых объектов;

**Wall, Ice, Water, Shrub и Bonus**

Классы производные от пустого класса Block, реализующие неуправляемые игровые объекты. Имеют лишь перегруженный метод Draw;

**Bullet**

Bullet реализует снаряды выпускаемые танками.

Методы:

* void Draw(object sender, PaintEventArgs e) – Обработчик отрисовки;
* void Timer\_tick(object sender, EventArgs e) – Обработчик таймера движения пули;

**Tank**

Tank, абстрактный базовый класс для EnemyTank и PlayerTank.

Методы:

* virtual void Timer\_move\_tick(object sender, EventArgs e) – Обработчик таймера движения танка;
* public virtual void Fire() – Производит выстрел танка;

**EnemyTank**

EnemyTank, класс танков компьютера.

Методы:

* void EnemyMove() – Меняет направление движения танка в соответствии с найденным путем (поле way);
* void ChangeColor() – Меняет цвет танка в соответствии с текущем значением здоровья;
* void Draw(object sender, PaintEventArgs e) – Обработчик отрисовки;
* override void Fire() – Перегруженный метод стрельбы танка;
* Boolean Order\_To\_Fire() – Определяет следует ли танку стрелять;
* List<Cordinates> Tracker(Cordinates target\_cordinates) – Возвращает список координат (класс Coordinates) кратчайшего пути от танка до указанной точки на игровом поле;
* void Find\_Players() – Определяет поведение танка в соответствии с его текущей фазой передвижения;
* void Change\_color(object sender, EventArgs e) – Обработчик таймера смены цвета для бонусного танка;
* override void Timer\_move\_tick(object sender, EventArgs e) – Обработчик таймера движения танка;
* void Timer\_stage\_tick(object sender, EventArgs e) – Обработчик таймера смены фазы передвижения танка;
* void Timer\_handler\_tick(object sender, EventArgs e) – Обработчик таймера проверки на необходимость стрельбы и нахождения нового пути;

**PlayerTank**

PlayerTank, класс танка игрока.

Методы:

* void Reload(object sender, EventArgs e) – Задает количество снарядов соответствующее текущему уровню;
* void Lvl\_Up() – Повышает уровень танка игрока;
* void Draw(object sender, PaintEventArgs e) – Обработчик отрисовки;
* Boolean Can\_It\_Go\_to() – Определяет может ли танк двигаться в указанном направлении на поле;
* override void Timer\_move\_tick(object sender, EventArgs e) – Обработчик таймера движения танка;
* void Timer\_slipping\_tick(object sender, EventArgs e) – Обработчик таймера скольжения танка;
* override void Fire() – Перегруженный метод стрельбы танка;

**TableRecords**

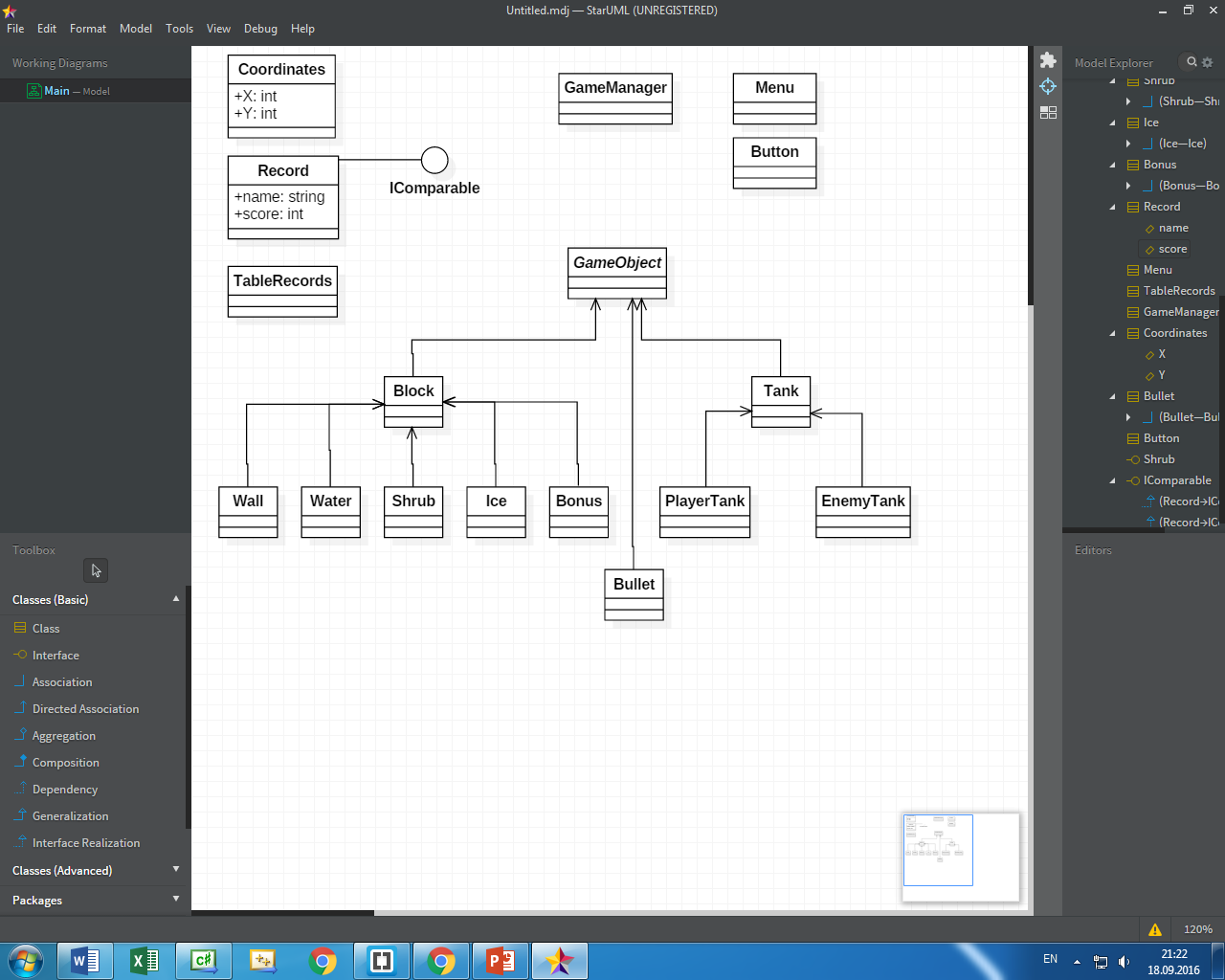
TableRecords, класс для таблицы рекордов в игре. Содержит лист с рекордами игроков(структура Record) получаемых из текстового файла.

Методы:

* static void Sort() – Сортирует полученые рекорды в порядке убывания;
* static Boolean Try\_Add(int score) – Определяет можно ли занести указаное количество очков таблицу рекордов;
* static void Add(string name,int score) – Добавляет указанный рекорд с указанным именем и счетом игрока в таблицу;
* static void Submit() – Переписывает текстовый файл с рекордами текущеми значениями рекордов в таблице;
* static void Clear() – Очищает таблицу;

**Record**

Record, структура для рекордов игроков. Содержит два поля: имя игрока и количество очков. Реализует интерфейс IComparable для возможности сортировки втренными методами.

  
Диаграмма 1 – Архитектура приложения

# **Описание алгоритмов**

# **Поведение танков компьютера**

У танков компьютера есть 3 фазы поведения: 1) Движение в случайную точку на поле; 2) Движение к ближайшему игроку; 3) Движение к штабу; В зависимости от текущей фазы в методе Find\_Players() определяется какой путь движения задать танку. Сам путь определяется методом Tracker(). Таймер timer\_for\_stage меняет через каждый тик текущую фазу на следующую. Тик таймера равен тройному интервалу появления танков на игровом поле. Интервал высчитывается по формуле (2 \* ((190 – текущий уровень \* 4 - (количество игроков - 1) \* 20) / 60)).

Timer\_for\_fire, который срабатывает каждые 2 секунды, реализует случайную стрельбу с вероятностью 50%.

Timer\_for\_handler постоянно проверяет через метод Find\_Players() нужно ли поменять текущий путь и через метод Order\_To\_Fire() находиться ли на пути движения танк игрока, по которому необходимо совершить выстрел.

# **Метод Tracker**

Метод Tracker реализует волновой алгоритм поиска кратчайшего пути. Изначально выбирается начальная точка – ячейка массива (field[,]), в координатах которых находиться танк компьютера, для которого ищется путь движения. Эта ячейка помечается индексом 1. Остальные ячейки массива равны 0, если по этим ячейкам может проехать танк, и -1, если по этим ячейкам танк проехать не может (стена или вода). Переменной step задается значение равное 1.

Далее в цикле while проверяются все ячейки массива.

1) Для каждой ячейки, индекс которой равен step, соседние ячейки, индекс которых равен 0, нумеруются индексом step+1;

2) Если индекс ячейки, в которой находятся координаты точки, в которую ищется путь, не равен 0, то цикл прерывается. Если он равен 0, то step увеличивается на 1 и ячейки массива проверяются снова.

Далее от ячейки с искомыми координатами восстанавливается путь. Текущими координатами задаются координаты этой ячейки.

1) Проверяются соседние к текущей ячейке ячейки. Выбирается ячейка с наименьшим индексом.

2) Координаты текущей ячейки записываются в List. После текущей ячейкой становиться выбранная в прошлом пункте ячейка.

3) Если координаты текущей ячейки равны координатам танка, для которого искался путь, то путь найден, иначе возвращаемся к пункту 1.

Для наглядного примера смотрите Рисунок 1.



Рисунок 1 – Восстановление пути в волновом алгоритме нахождения кратчайшего пути

# **Руководство пользователя**

При запуске программы пользователь видит главное меню (Рисунок 2), которое состоит из 4 кнопок: “1 Игрок”, “2 Игрока”, “Рекорды” и “Выход”.

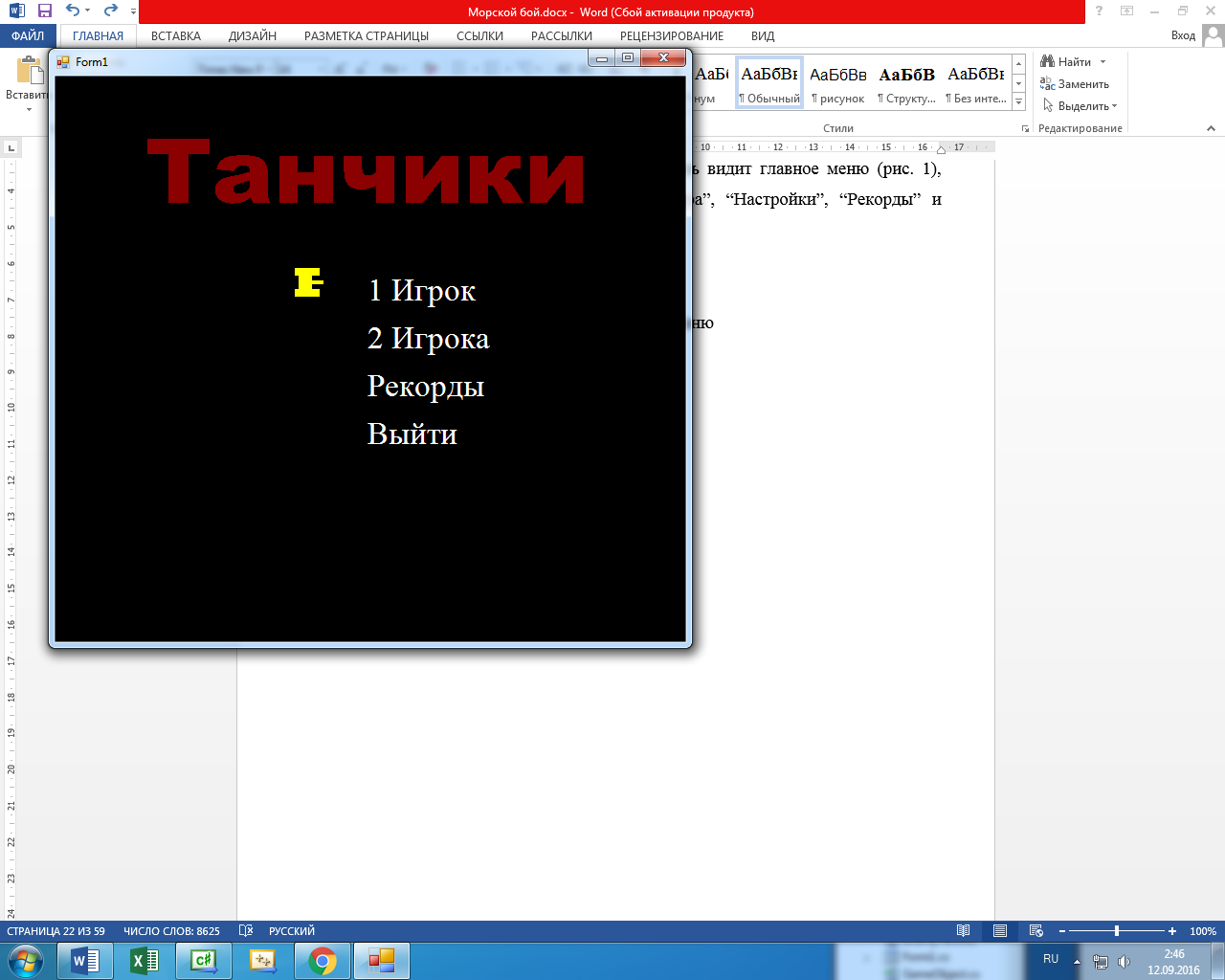


Рисунок 2 – Главное меню

При нажатии на кнопку “Рекорды” пользователь видит таблицу рекордов игроков (Рисунок 3). Так же пользователю предоставляется возможность очистки таблицы путем нажатия на кнопку “Очистить”.

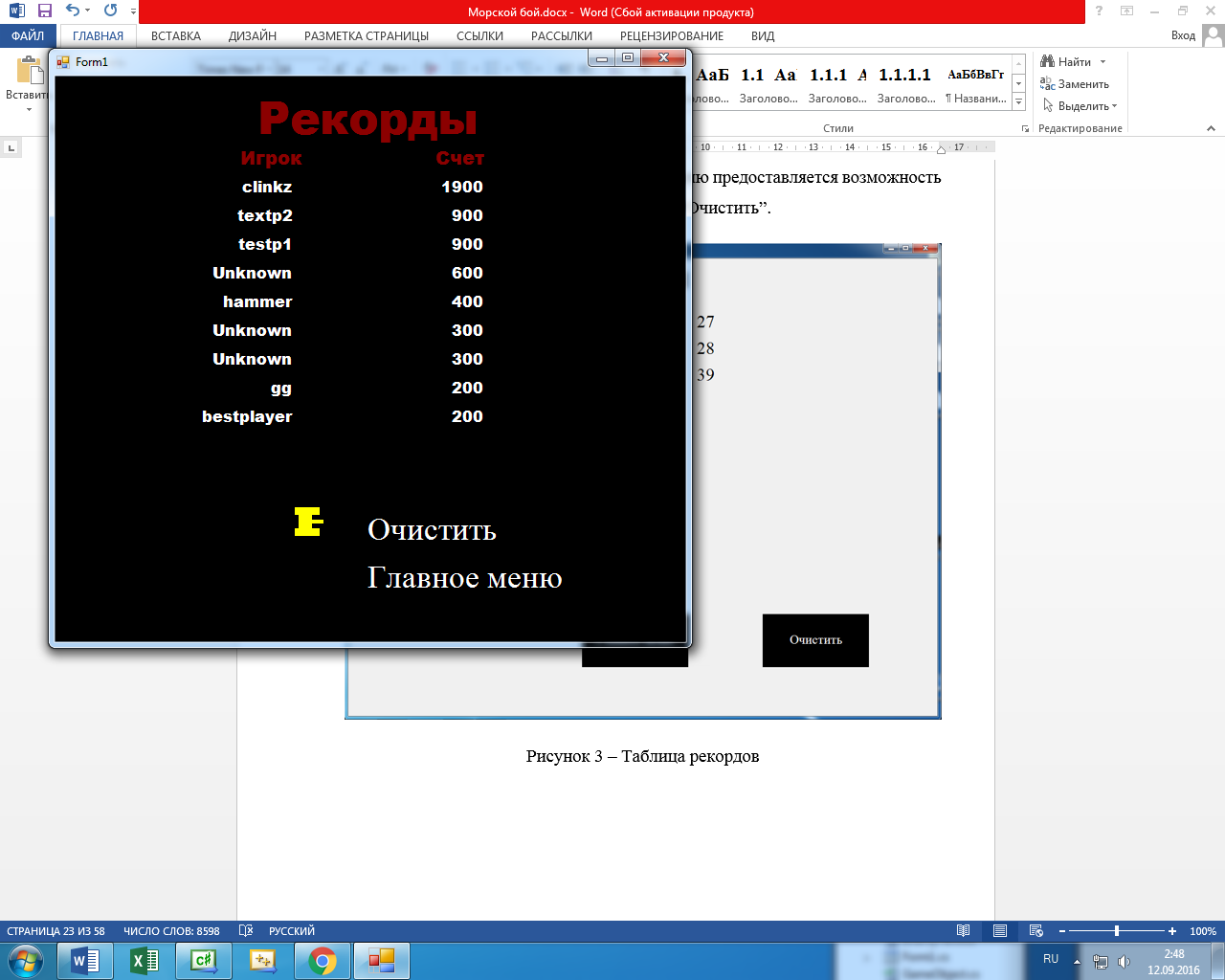


Рисунок 3 – Таблица рекордов

При нажатии на кнопку “Выход” у пользователя уточняя желание покинуть игру (Рисунок 4). При подтверждении приложение закрывается, иначе пользователь возвращается в главное меню.

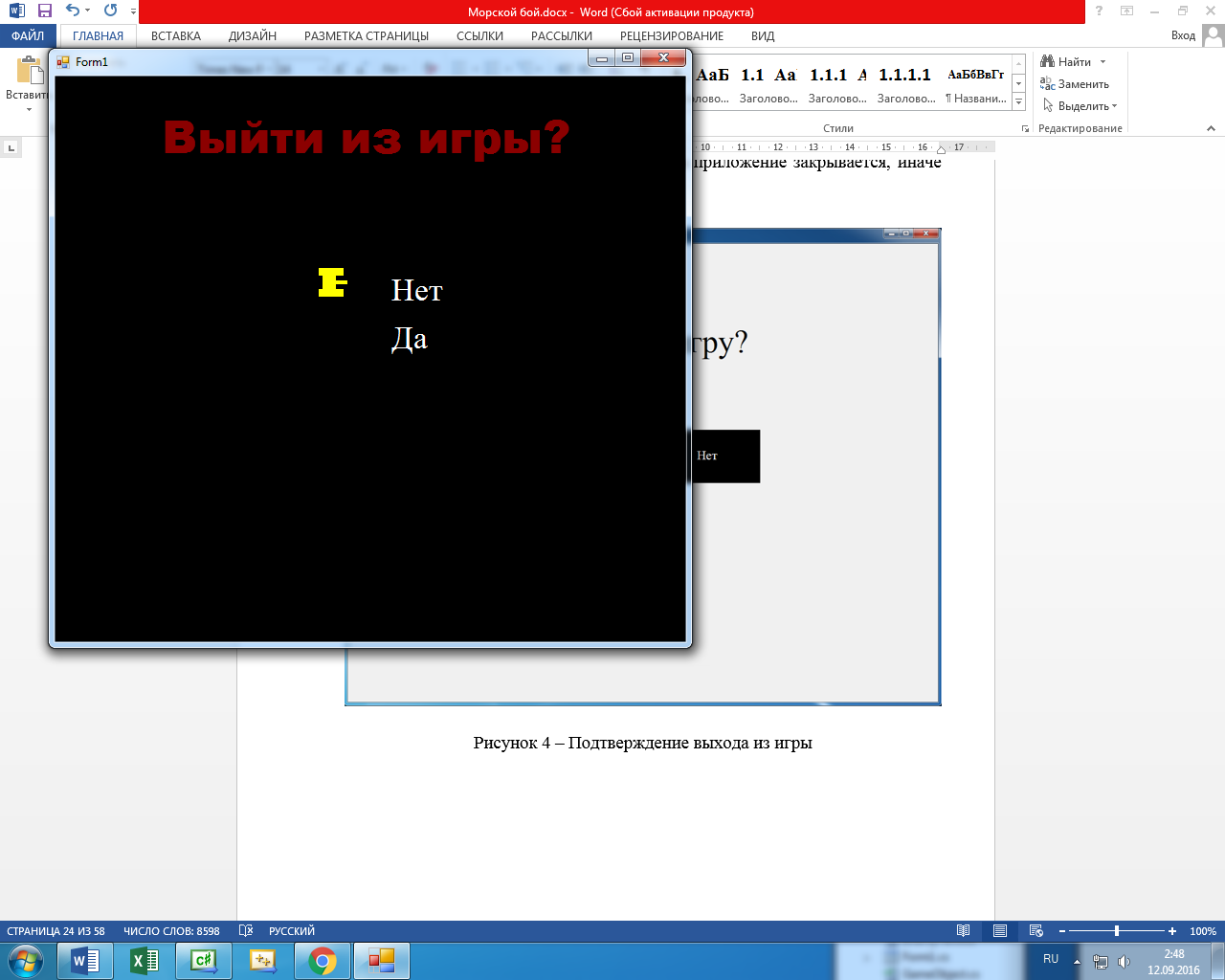


Рисунок 4 – Подтверждение выхода из игры

При нажатии на одну из кнопок “1 Игрок” или “2 Игрока” начинается игровой процесс для одного или двух игроков соответственно (Рисунок 5).

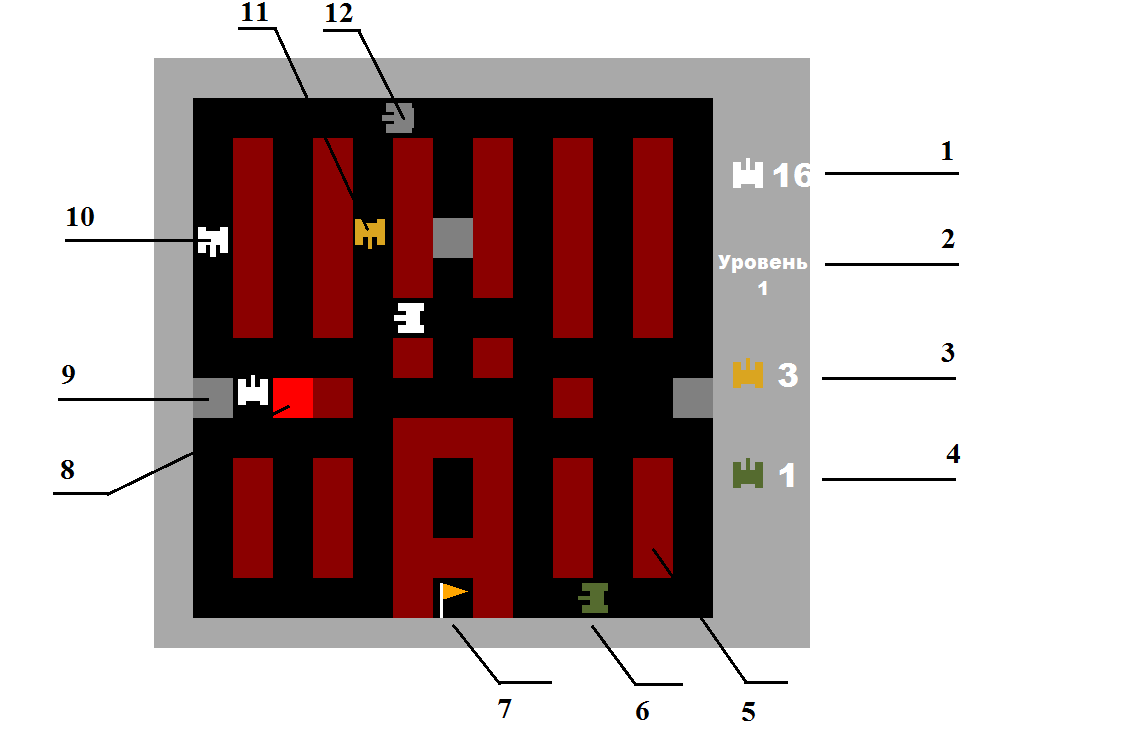


Рисунок 5 – Игровой процесс

1 – Количество оставшихся в запасе у компьютера танков;

2 – Номер текущего уровня;

3 – Количество жизней у 1-го игрока;

4 – Количество жизней у 2-го игрока;

5 – Кирпичная стена;

6 – Танк 2-го игрока;

7 – Штаб;

8 – Поврежденная кирпичная стена;

9 – Бетонная стена;

10 – Танк компьютера 1-го уровня;

11 – Танк 1-го игрока;

12 – Танк компьютера 4-го уровня.

Существует 4 типа танка компьютера:

* Обычный;
* Бронетранспортер, с удвоенной скоростью хода;
* Скоростельный, с удвоенной скоростью полета снаряда;
* Тяжелый, с 3мя жизнями и возможностью пробивать бетон.

У танка игрока есть 4 уровня:

1. Обычный;
2. С удвоенной скоростью полета снаряда;
3. С возможностью стрелять очередью (3 снаряда в 2 секунды);
4. С возможностью пробивать бетон.

Характиристики танков показаны на рисунке 6.



Рисунок 6 – Виды танков

Танк 1 игрока управляется при помощи клавиш “A” “W” “S” “D” и стреляет при нажатии “G”. Танк 2 игрока клавишами “↓” “←” “→” “↑” и “Enter”. Также игроки могут прервать игру, нажав клавишу “Esc”, или поставить игру на паузу, нажав клавишу “P”.

В игре существуют разные типы ландшафта:

* Стена(кирпичная). Танку (кроме танка игрока 4-го уровня и тяжелого танка компьютера) требуется 2 выстрела для уничтожения;
* Бетонная стена. Уничтожить может только танк игрока 4-го уровня и тяжелый танк компьютера;
* Трава. Визуально скрывает танк;
* Вода. По воде танки ехать не могут;
* Лед. Если танк не управляется игроком, то он начинает скользить.

Также в игре реализованы бонусные объекты, которые появляются после уничтожения бонусного танка (танк с переливающейся окраской).

Существует 6 видов бонусов:

* *«Танк»*. Прибавляет игроку одну жизнь;
* *«Звезда»*. Улучшает танк игрока на 1 уровень;
* *«Бомба».* Взрывает танки противника на карте, за их уничтожение очки не начисляются — только 500 очков за взятие бонуса как такового;
* *«Часы»*. Останавливает врагов и их стрельбу на 5 секунд;
* *«Лопата»*. На 20 секунд делает кирпичную стену штаба бетонной. После прекращения действия вокруг штаба восстанавливается кирпичная стена;
* *«Каска»*. Делает танк игрока неуязвимым на 10 секунд.

После прохождения каждого уровня происходит подсчет заработанных очков (Рисунок 7).

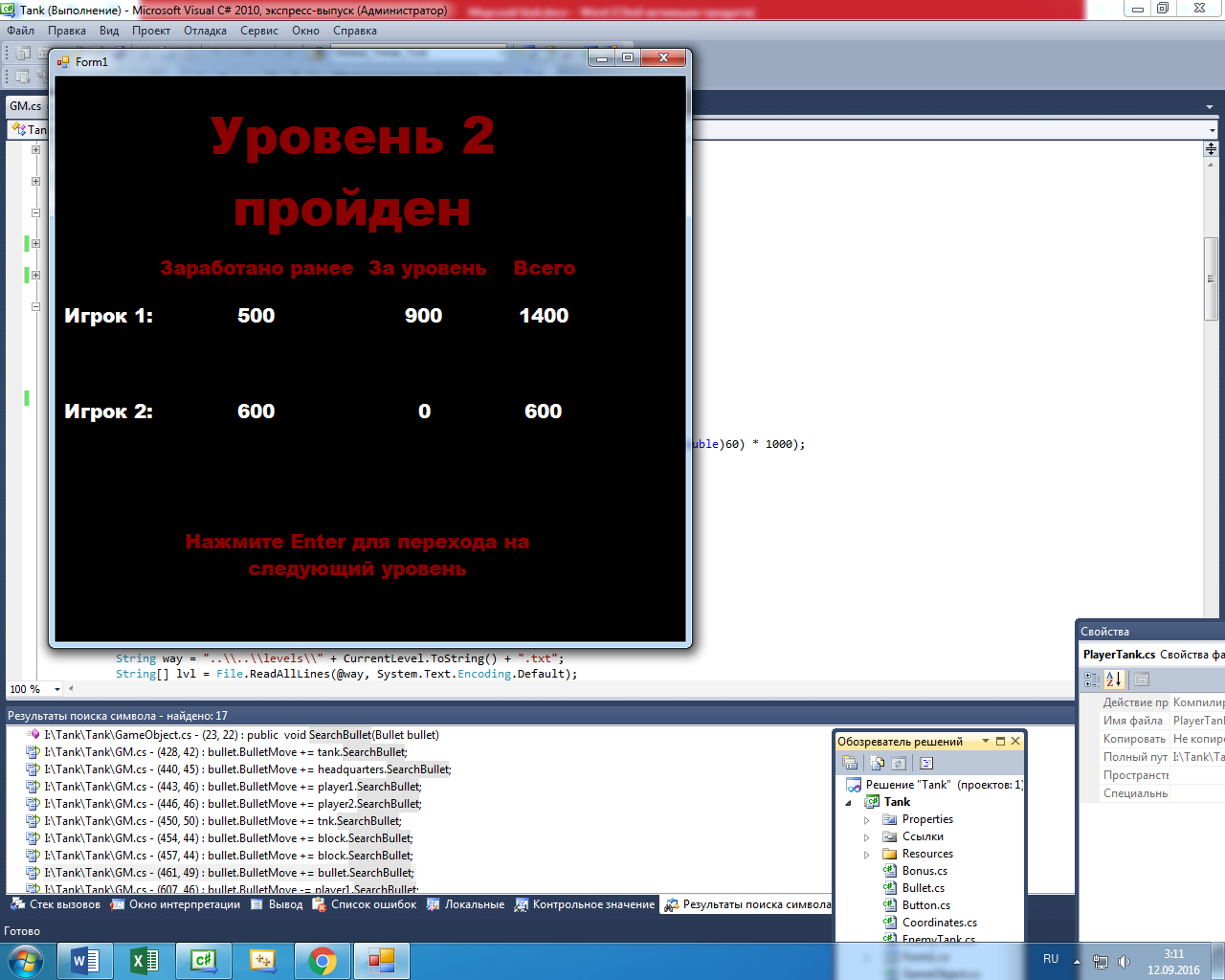


Рисунок 7 – Подсчет очков в конце уровня

После окончания игры, игрок перенаправляется в таблицу рекордов. Если игрок набрал достаточно очков для записи в таблицу рекордов, то ему предлагается ввести свое имя для записи в таблицу (Рисунок 8).

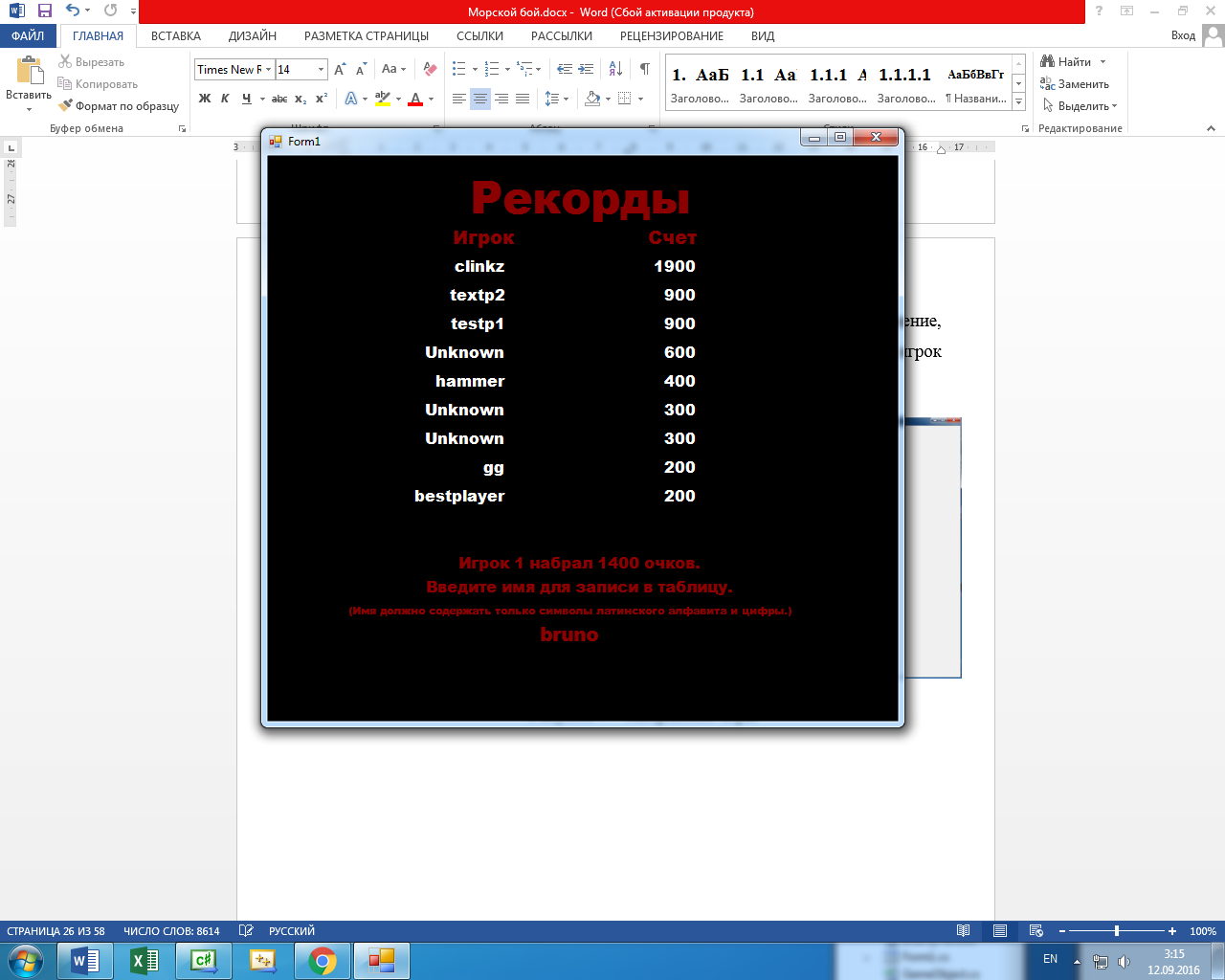


Рисунок 8 – Запись в таблицу рекордов

# **Заключение**

В результате выполнения учебной практики было реализовано приложение, соответствующие парадигмам ООП и представляющее собой аркадную игру «Танчики». В процессе разработки были изучены такие понятия в программирование, как “События”, “Делегаты”, “Интерфейс”, “Абстрактные классы и методы”, “Виртуальные функции” и “Перегрузка функций”.

# **Приложение**

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

this.DoubleBuffered = true;

this.BackColor = System.Drawing.Color.Black;

GameManager.ON(this);

}

}

public delegate void DelTankMove (Tank tank);

public delegate void DelNewBullet(Bullet bullet);

public delegate void DelBulletMove(Bullet b);

public delegate void DelCollide(GameObject one, GameObject two);

public delegate void DelPlayerInArea (object \_object, PlayerTank tank);

public delegate Coordinates DelForPLace ();

public delegate Boolean DelOrderToMove(Tank tank,Direction direction);

public static class GameManager

{

public static Form1 form;

#region Lists

static List<EnemyTank> tanks = new List<EnemyTank>();

static List<Wall> walls = new List<Wall>();

static List<Wall> fence = new List<Wall>();

static List<GameObject> landscapes = new List<GameObject>();

static List<Ice> ice = new List<Ice>();

static List<Bullet> bullets = new List<Bullet>();

static Wall headquarters;

#endregion

#region Timers

static System.Threading.Timer timer\_for\_beton;

static System.Threading.Timer timer\_for\_shield;

static System.Threading.Timer timer\_for\_ffire;

static Timer global\_timer = new Timer();

static Timer timer\_for\_pause = new Timer();

static Timer timer\_for\_stoptime = new Timer();

#endregion

#region Fields

static int interval;

static int tank\_for\_lvl;

static int[,] field = new int[15, 15];

static int CurrentLevel;

static Bonus currentbonus;

static PlayerTank player1 = null;

static PlayerTank player2 = null;

static int player\_count;

static int player1\_total\_score = 0;

static int player2\_total\_score = 0;

static int player1\_lvl\_score = 0;

static int player2\_lvl\_score = 0;

static string playername="";

static DateTime delay\_for\_tank = DateTime.Now;

static Random random = new Random();

#endregion

#region Game

public static void ON(Form1 \_form)

{

form = \_form;

Menu.form = form;

MainMenu();

global\_timer.Tick += Global\_Timer\_Tick;

timer\_for\_stoptime.Tick += StopTime\_Tick;

timer\_for\_pause.Tick += pause\_tick;

global\_timer.Interval = 80;

timer\_for\_stoptime.Interval = 5000;

timer\_for\_pause.Interval = 50;

GameObject.collide = Collide;

Tank.NewBullet = NewBullet;

EnemyTank.Place = Place;

}

static void StartGame(Boolean two = false)

{

form.KeyDown += Press\_Interrupt;

form.Paint -= Draw\_Mainmenu;

form.Paint += Draw\_Game;

Menu.Remove();

headquarters = new Wall(new Coordinates(7 \* 40, 13 \* 40), 1);

field[7, 13] = -1;

global\_timer.Start();

player1\_total\_score = 0;

player2\_total\_score = 0;

player1\_lvl\_score = 0;

player2\_lvl\_score = 0;

Tank.field = field;

player\_count = 1;

player1 = new PlayerTank(new Coordinates(202, 525),1);

form.KeyDown += Player1Move\_keydown;

form.KeyUp += Player1Move\_keyup;

EnemyTank.player1 = player1;

if (two)

{

player\_count = 2;

player2 = new PlayerTank(new Coordinates(369, 525), 1, true);

form.KeyDown += Player2Move\_keydown;

form.KeyUp += Player2Move\_keyup;

EnemyTank.player2 = player2;

}

Wall block;

for (int i = 0; i < 15; i++)

{

field[0, i] = -1;

block = new Wall(new Coordinates( 0, i \* 40 ), -1);

fence.Add(block);

field[14, i] = -1;

block = new Wall(new Coordinates(14\*40, i \* 40), -1);

fence.Add(block);

field[i, 0] = -1;

block = new Wall(new Coordinates(i \* 40 , 0), -1);

fence.Add(block);

field[i, 14] = -1;

block = new Wall(new Coordinates(i \* 40 , 14 \* 40 ), -1);

fence.Add(block);

}

CurrentLevel = 16;

Nextlvl();

}

static void Nextlvl()

{

CurrentLevel++;

player1\_lvl\_score = 0;

player2\_lvl\_score = 0;

tank\_for\_lvl = 20;

interval = (int)(2 \* ((double)(190 - CurrentLevel \* 4 - (player\_count - 1) \* 20) / (double)60) \* 1000);

if (player1 != null)

{

player1.coordinates = player1.startposition;

player1.direction = Direction.top;

form.Paint += player1.Draw;

}

if (player2 != null)

{

player2.coordinates = player2.startposition;

player2.direction = Direction.top;

form.Paint += player2.Draw;

}

String way = "..\\..\\levels\\" + CurrentLevel.ToString() + ".txt";

String[] lvl = File.ReadAllLines(@way, System.Text.Encoding.Default);

Wall block;

GameObject landscape;

Ice \_ice;

for (int i = 0; i < 26; i += 2)

for (int j = 0; j < 26; j += 2)

{

switch (lvl[i][j])

{

case '#'://стена);

block = new Wall(new Coordinates(j \* 20 + 40, i \* 20 + 40), 2);

walls.Add(block);

form.Paint += block.Draw;

field[j / 2 + 1, i / 2 + 1] = -1;

break;

case '@'://бетон);

block = new Wall(new Coordinates(j \* 20 + 40, i \* 20 + 40), 3);

walls.Add(block);

form.Paint += block.Draw;

field[j / 2 + 1, i / 2 + 1] = -1;

break;

case '%'://кусты);

landscape = new Shrub(new Coordinates(j \* 20 + 40, i \* 20 + 40));

landscapes.Add(landscape);

form.Paint += landscape.Draw;

break;

case '~'://вода);

landscape = new Water(new Coordinates(j \* 20 + 40, i \* 20 + 40));

landscapes.Add(landscape);

form.Paint += landscape.Draw;

field[j / 2 + 1, i / 2 + 1] = -1;

break;

case '-': // "лед);

\_ice = new Ice(new Coordinates(j \* 20 + 40, i \* 20 + 40));

ice.Add(\_ice);

if (player1 != null)

player1.TankMove += \_ice.SearchGameObject;

if (player2 != null)

player2.TankMove += \_ice.SearchGameObject;

form.Paint += \_ice.Draw;

break;

}

}

if (ice.Count != 0)

{

if (player1 != null)

{

form.Paint -= player1.Draw;

form.Paint += player1.Draw;

}

if (player2 != null)

{

form.Paint -= player2.Draw;

form.Paint += player2.Draw;

}

}

block = new Wall(new Coordinates(6 \* 40, 40 \* 13), 2);

walls.Add(block);

form.Paint += block.Draw;

field[6, 13] = -1;

block = new Wall(new Coordinates(6 \* 40, 40 \* 12), 2);;

walls.Add(block);

form.Paint += block.Draw;

field[6, 12] = -1;

block = new Wall(new Coordinates(7 \* 40, 40 \* 12), 2);

walls.Add(block);

form.Paint += block.Draw;

field[7, 12] = -1;

block = new Wall(new Coordinates(8 \* 40, 40 \* 12), 2);

walls.Add(block);

form.Paint += block.Draw;

field[8, 12] = -1;

global\_timer.Start();

}

static void Clearlvl(bool fully = false)

{

while (walls.Count != 0)

RemoveWall(walls[0]);

while (bullets.Count != 0)

RemoveBullet(bullets[0]);

while (landscapes.Count != 0)

RemoveLandScape(landscapes[0]);

while (ice.Count != 0)

RemoveIce(ice[0]);

if (currentbonus != null)

RemoveBonus();

form.Paint -= Draw\_Game;

form.KeyDown -= Press\_Interrupt;

form.KeyDown -= Player1Move\_keydown;

form.KeyUp -= Player1Move\_keyup;

form.KeyDown -= Player2Move\_keydown;

form.KeyUp -= Player2Move\_keyup;

if (player1 != null)

{

form.Paint -= player1.Draw;

player1.Timer\_Move.Stop();

}

if (player2 != null)

{

form.Paint -= player2.Draw;

player2.Timer\_Move.Stop();

}

if (fully)

{

while (tanks.Count != 0)

Remove\_EnemyTank(tanks[0], null);

while (fence.Count != 0)

{

form.Paint -= fence[0].Draw;

fence.Remove(fence[0]);

}

if (player1 != null) RemovePlayer(player1);

if (player2 != null) RemovePlayer(player2);

}

form.Invalidate();

}

static void Changelvl()

{

global\_timer.Stop();

form.Paint -= Draw\_Game;

if (CurrentLevel != 35)

{

Clearlvl();

form.Paint += Draw\_Between\_lvl;

form.KeyUp += Press\_Nextlvl;

}

else

{

Clearlvl(true);

if (player1 != null) RemovePlayer(player1);

if (player2 != null) RemovePlayer(player2);

player1\_total\_score += player1\_lvl\_score;

player2\_total\_score += player2\_lvl\_score;

Go\_To\_TR();

}

}

static void GameOver()

{

global\_timer.Stop();

player1\_total\_score += player1\_lvl\_score;

player2\_total\_score += player2\_lvl\_score;

Clearlvl(true);

form.Paint -= Draw\_Game;

Go\_To\_TR();

}

static void Interrupt()

{

Clearlvl(true);

global\_timer.Stop();

form.Paint -= Draw\_Game;

form.Paint -= Draw\_Pause;

MainMenu();

}

static void Go\_To\_TR()

{

playername = "";

if (TableRecords.Order\_To\_Add(player1\_total\_score) || TableRecords.Order\_To\_Add(player2\_total\_score))

{

form.KeyPress += Press\_WriteName;

form.Paint += Draw\_Writename;

form.Paint += Draw\_Records;

}

else

Menu\_Records();

}

#endregion

#region Menu

static void MainMenu()

{

form.Paint -= Draw\_Records;

form.Paint -= Draw\_Exit;

form.Paint += Draw\_Mainmenu;

Menu.Remove();

Menu.Add(new Button(new Coordinates(250, 200), "1 Игрок", Menu\_1Player));

Menu.Add(new Button(new Coordinates(250, 250), "2 Игрока", Menu\_2Players));

Menu.Add(new Button(new Coordinates(250, 300), "Рекорды", Menu\_Records));

Menu.Add(new Button(new Coordinates(250, 350), "Выйти", Menu\_Exit));

}

static void Menu\_1Player()

{

StartGame(); ;

}

static void Menu\_2Players()

{

StartGame(true);

}

static void Menu\_Records()

{

form.Paint -= Draw\_Mainmenu;

form.Paint += Draw\_Records;

Menu.Remove();

Menu.Add(new Button(new Coordinates(250, 450), "Очистить", TableRecords.Clear));

Menu.Add(new Button(new Coordinates(250, 500), "Главное меню", MainMenu));

}

static void Menu\_Exit()

{

form.Paint -= Draw\_Mainmenu;

form.Paint += Draw\_Exit;

Menu.Remove();

Menu.Add(new Button(new Coordinates(275, 200), "Нет", MainMenu));

Menu.Add(new Button(new Coordinates(275, 250), "Да", () => { System.Environment.Exit(0); }));

}

#endregion

#region Newobjects

static void NewTank()

{

Boolean bonustank = false;

if (tank\_for\_lvl == 2 || tank\_for\_lvl == 9 || tank\_for\_lvl == 16)

bonustank = true;

int choice = random.Next(8);

int tanklevel = 0;

if (choice >= 0 && choice <= 2)

tanklevel = 1;

if (choice >= 3 && choice <= 4)

tanklevel = 2;

if (choice >= 5 && choice <= 6)

tanklevel = 3;

if (choice == 7)

tanklevel = 4;

EnemyTank tank = new EnemyTank(new Coordinates(45 + 240 \* random.Next(3), 45), tanklevel,3\*interval ,bonustank);

form.Paint += tank.Draw;

foreach (var bullet in bullets)

bullet.BulletMove += tank.SearchGameObject;

tanks.Add(tank);

tank\_for\_lvl--;

Under\_Bush();

}

static void NewBullet(Bullet bullet)

{

form.Paint += bullet.Draw;

bullet.BulletMove += headquarters.SearchGameObject;

if (player1 != null && bullet.tank != player1)

bullet.BulletMove += player1.SearchGameObject;

if (player2 != null && player2 != bullet.tank)

bullet.BulletMove += player2.SearchGameObject;

if (bullet.tank is PlayerTank)

foreach (var tnk in tanks)

if (tnk != bullet.tank)

bullet.BulletMove += tnk.SearchGameObject;

foreach(var block in walls)

bullet.BulletMove += block.SearchGameObject;

foreach (var block in fence)

bullet.BulletMove += block.SearchGameObject;

foreach (var \_bullet in bullets)

if (\_bullet != bullet)

bullet.BulletMove += \_bullet.SearchGameObject;

bullets.Add(bullet);

Under\_Bush();

}

static void NewBonus()

{

if (currentbonus != null)

RemoveBonus();

BonusType bonustype;

switch (random.Next(6))

{

case 0:

bonustype = BonusType.beton;

break;

case 1:

bonustype = BonusType.bomb;

break;

case 2:

bonustype = BonusType.live;

break;

case 3:

bonustype = BonusType.lvl;

break;

case 4:

bonustype = BonusType.shield;

break;

case 5:

bonustype = BonusType.timestop;

break;

default:

bonustype = BonusType.beton;

break;

}

Bonus bonus = new Bonus(Place(), bonustype);

form.Paint += bonus.Draw;

if(player1!=null)

player1.TankMove+=bonus.SearchGameObject;

if(player2!=null)

player2.TankMove += bonus.SearchGameObject;

currentbonus = bonus;

}

#endregion

#region Remove

static void Collide(GameObject one, GameObject two)

{

if (one is Bullet)

{

RemoveBullet(one as Bullet);

if (two is Bullet)

RemoveBullet(two as Bullet);

if (two is EnemyTank)

Hit\_EnemyTank(two as EnemyTank, one as Bullet);

if (two is PlayerTank)

Hit\_Player(two as PlayerTank, one as Bullet);

if (two is Wall)

Hit\_Wall(two as Wall, one as Bullet);

if (two == headquarters)

{

RemoveWall(two as Wall);

GameOver();

}

}

else

{

OnArea( two, one as Tank);

}

}

static void Hit\_EnemyTank(EnemyTank tank, Bullet bullet)

{

if (tank.bonustank)

{

tank.timer\_for\_color.Stop();

tank.brush.Color = tank.color;

NewBonus();

tank.bonustank = false;

}

tank.HP--;

if (tank.HP == 0)

Remove\_EnemyTank(tank, bullet);

else

tank.ChangeColor();

}

static void Hit\_Player(PlayerTank tank, Bullet bullet)

{

if (!tank.shield)

{

if (bullet.tank is PlayerTank)

{

if (tank == player1)

{

form.KeyDown -= Player1Move\_keydown;

form.KeyUp -= Player1Move\_keyup;

player2.Timer\_Move.Stop();

timer\_for\_ffire = new System.Threading.Timer(Friendly\_Fire, player1 as object, 3000, -1);

}

else

{

form.KeyDown -= Player2Move\_keydown;

form.KeyUp -= Player2Move\_keyup;

player2.Timer\_Move.Stop();

timer\_for\_ffire = new System.Threading.Timer(Friendly\_Fire, player2 as object, 3000, -1);

}

}

else

{

tank.HP--;

tank.lvl = 1;

tank.coordinates = tank.startposition;

if (tank.HP == 0)

RemovePlayer(tank);

}

}

}

static void Hit\_Wall(Wall block, Bullet bullet)

{

if (block.coordinates.x >= 40 && block.coordinates.x < 560 && block.coordinates.y >= 40 && block.coordinates.y < 560)

{

if (block.HP != 3)

block.HP--;

if (block.HP == 0 || bullet.lvl == 4)

RemoveWall(block);

}

}

static void RemoveBullet(Bullet bullet)

{

bullets.Remove(bullet);

bullet.timer.Enabled = false;

form.Paint -= bullet.Draw;

if (player1 != null)

bullet.BulletMove -= player1.SearchGameObject;

if (player2 != null)

bullet.BulletMove -= player2.SearchGameObject;

foreach (var tank in tanks)

bullet.BulletMove -= tank.SearchGameObject;

foreach (var block in walls)

bullet.BulletMove -= block.SearchGameObject;

foreach (var \_bullet in bullets)

bullet.BulletMove -= \_bullet.SearchGameObject;

}

static void RemoveWall(Wall block)

{

walls.Remove(block);

form.Paint -= block.Draw;

field[block.coordinates.x / 40, block.coordinates.y / 40] = 0;

foreach (var bullet in bullets)

{

bullet.BulletMove -= block.SearchGameObject;

}

}

static void Remove\_EnemyTank(EnemyTank tank, Bullet bullet)

{

int score = 0;

if (bullet != null)

{

switch (tank.lvl)

{

case 1:

score = 100;

break;

case 2:

score = 200;

break;

case 3:

score = 300;

break;

case 4:

score = 400;

break;

}

if (bullet.tank == player1)

player1\_lvl\_score += score;

else

player2\_lvl\_score += score;

}

form.Paint -= tank.Draw;

tank.timer\_for\_stage.Stop();

tank.timer\_for\_fire.Stop();

tank.timer\_for\_handler.Stop();

foreach (var \_bullet in bullets)

{

\_bullet.BulletMove -= tank.SearchGameObject;

}

tanks.Remove(tank);

}

static void RemoveIce(Ice \_ice)

{

form.Paint -= \_ice.Draw;

if (player1 != null)

player1.TankMove -= \_ice.SearchGameObject;

if (player2 != null)

player2.TankMove -= \_ice.SearchGameObject;

ice.Remove(\_ice);

}

static void RemoveLandScape(GameObject landscape)

{

landscapes.Remove(landscape);

form.Paint -= landscape.Draw;

field[landscape.coordinates.x / 40, landscape.coordinates.y / 40] = 0;

}

static void RemoveBonus()

{

form.Paint -= currentbonus.Draw;

if (player1 != null)

player1.TankMove -= currentbonus.SearchGameObject;

if (player2 != null)

player2.TankMove -= currentbonus.SearchGameObject;

currentbonus = null;

}

static void RemovePlayer(PlayerTank tank)

{

if (tank == player1)

{

form.KeyDown -= Player1Move\_keydown;

form.KeyUp -= Player1Move\_keyup;

form.Paint -= player1.Draw;

player1.Timer\_Move.Stop();

player1 = null;

}

else

{

form.KeyDown -= Player2Move\_keydown;

form.KeyUp -= Player2Move\_keyup;

form.Paint -= player2.Draw;

player2.Timer\_Move.Stop();

player2 = null;

}

foreach (var bullet in bullets)

{

bullet.BulletMove -= tank.SearchGameObject;

}

}

#endregion

#region Bonus

static void UseBonus(Bonus bonus, Tank tank)

{

if (tank == player1)

player1\_lvl\_score += 500;

else

player2\_lvl\_score += 500;

switch (bonus.type)

{

case BonusType.beton:

Bonus\_Beton(3);

break;

case BonusType.bomb:

Bonus\_Bomb();

break;

case BonusType.live:

Bonus\_Live(tank);

break;

case BonusType.lvl:

Bonus\_lvl(tank as PlayerTank);

break;

case BonusType.shield:

Bonus\_Set\_Shield(tank as PlayerTank);

break;

case BonusType.timestop:

Bonus\_TimeStop();

break;

}

RemoveBonus();

}

static void Bonus\_Beton(object HP)

{

List<Wall> blocks\_buffer = new List<Wall>();

foreach (var block in walls)

if (block.coordinates.x / 40 >= 6 && block.coordinates.x / 40 <= 8

&& block.coordinates.y / 40 >= 12 && block.coordinates.y / 40 <= 13)

blocks\_buffer.Add(block);

foreach (var block in blocks\_buffer)

{

walls.Remove(block);

form.Paint -=block.Draw;

}

Wall \_block;

for (int i = 12; i <= 13; i++)

{

field[6, i] = -1;

field[8, i] = -1;

\_block = new Wall(new Coordinates(6 \* 40 , i \* 40), (int)HP);

walls.Add(\_block);

form.Paint += \_block.Draw;

\_block = new Wall(new Coordinates(8 \* 40 , i \* 40), (int)HP);

walls.Add(\_block);

form.Paint += \_block.Draw;

}

field[7, 12] = -1;

\_block = new Wall(new Coordinates(7 \* 40 , 12 \* 40), (int)HP);

walls.Add(\_block);

form.Paint += \_block.Draw;

timer\_for\_beton = new System.Threading.Timer(Bonus\_Beton, 2, 20000, -1);

}

static void Bonus\_Set\_Shield( PlayerTank tank)

{

tank.shield = true;

timer\_for\_shield = new System.Threading.Timer(Bonus\_Destroy\_Shield, null, 10000, -1);

}

static void Bonus\_Destroy\_Shield( object empty)

{

if (player1 != null)

player1.shield = false;

if (player2 != null)

player2.shield = false;

}

static void Bonus\_Bomb()

{

while( tanks.Count > 0)

Remove\_EnemyTank(tanks[0],null);

}

static void Bonus\_lvl(PlayerTank tank)

{

tank.Lvl\_Up();

}

static void Bonus\_Live(Tank tank)

{

tank.HP++;

}

static void Bonus\_TimeStop()

{

timer\_for\_stoptime.Start();

foreach (var tank in tanks)

{

tank.Timer\_Move.Stop();

tank.timer\_for\_stage.Stop();

tank.timer\_for\_fire.Stop();

tank.timer\_for\_handler.Stop();

}

}

#endregion

#region Logic

static Coordinates Place()

{

int x;

int y;

Random random = new Random();

while (true)

{

x = random.Next(14) + 1;

y = random.Next(14) + 1;

if (field[x, y] != -1)

break;

}

return new Coordinates(40 \* x + 1, 40 \* y + 1);

}

static void OnArea(Object \_object, Tank tank)

{

if (\_object is Ice)

OnIce(\_object, tank as PlayerTank);

if (\_object is Bonus)

UseBonus(\_object as Bonus, tank);

}

static void Friendly\_Fire(object tank)

{

if (tank == player1)

{

form.KeyDown += Player1Move\_keydown;

form.KeyUp += Player1Move\_keyup;

}

else

{

form.KeyDown += Player2Move\_keydown;

form.KeyUp += Player2Move\_keyup;

}

}

static void OnPause()

{

if (global\_timer.Enabled == true)

{

global\_timer.Stop();

form.KeyDown -= Player1Move\_keydown;

form.KeyDown -= Player2Move\_keydown;

foreach (var tank in tanks)

{

tank.Timer\_Move.Stop();

tank.timer\_for\_stage.Stop();

tank.timer\_for\_fire.Stop();

tank.timer\_for\_handler.Stop();

}

timer\_for\_pause.Start();

form.Paint += Draw\_Pause;

form.Invalidate();

}

else

{

global\_timer.Start();

form.KeyDown += Player1Move\_keydown;

form.KeyDown += Player2Move\_keydown;

foreach (var tank in tanks)

{

tank.Timer\_Move.Start();

tank.timer\_for\_stage.Start();

tank.timer\_for\_fire.Start();

tank.timer\_for\_handler.Start();

}

timer\_for\_pause.Stop();

form.Paint -= Draw\_Pause;

}

}

static void OnIce(object \_object, PlayerTank tank)

{

if (global\_timer.Enabled == true)

tank.Timer\_Slipping.Start();

}

static void Under\_Bush()

{

foreach (var bush in landscapes)

{

if (bush is Shrub)

{

form.Paint -= bush.Draw;

form.Paint += bush.Draw;

}

}

}

#endregion

#region Timers\_Ticks

static void Global\_Timer\_Tick(object sender, EventArgs e)

{

form.Invalidate();

if (tank\_for\_lvl <= 0)

{

delay\_for\_tank = DateTime.Now;

if (tanks.Count == 0)

{

Changelvl();

return;

}

}

if (tanks.Count >= 4 || tanks.Count >= 6 && player2 != null)

delay\_for\_tank = DateTime.Now;

if (DateTime.Now.Subtract(delay\_for\_tank).TotalMilliseconds > interval )

{

delay\_for\_tank = DateTime.Now;

NewTank();

}

if (player1 == null && player2 == null)

GameOver();

}

static void StopTime\_Tick(object sender, EventArgs e)

{

foreach (var tank in tanks)

{

tank.Timer\_Move.Start();

tank.timer\_for\_handler.Start();

tank.timer\_for\_stage.Start();

tank.timer\_for\_fire.Start();

}

timer\_for\_stoptime.Stop();

}

static void pause\_tick(object sender, EventArgs e)

{

foreach (var tank in tanks)

{

tank.last\_shot = tank.last\_shot.AddMilliseconds(timer\_for\_pause.Interval);

}

if (player1 != null)

player1.last\_shot.AddMilliseconds(timer\_for\_pause.Interval);

if (player2 != null)

player2.last\_shot.AddMilliseconds(timer\_for\_pause.Interval);

delay\_for\_tank = delay\_for\_tank.AddMilliseconds(timer\_for\_pause.Interval);

}

#endregion

#region Handlers

static void Player1Move\_keydown(object sender, KeyEventArgs e)

{

switch (e.KeyCode)

{

case Keys.D:

player1.direction = Direction.right;

player1.Timer\_Move.Enabled = true;

break;

case Keys.S:

player1.direction = Direction.bottom;

player1.Timer\_Move.Enabled = true;

break;

case Keys.A:

player1.direction = Direction.left;

player1.Timer\_Move.Enabled = true;

break;

case Keys.W:

player1.direction = Direction.top;

player1.Timer\_Move.Enabled = true;

break;

case Keys.G:

player1.Fire();

break;

}

}

static void Player1Move\_keyup(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.D || e.KeyCode == Keys.S || e.KeyCode == Keys.A || e.KeyCode == Keys.W)

{

player1.Timer\_Move.Enabled = false;

}

}

static void Player2Move\_keydown(object sender, KeyEventArgs e)

{

switch (e.KeyCode)

{

case Keys.Right:

player2.direction = Direction.right;

player2.Timer\_Move.Start();

break;

case Keys.Down:

player2.direction = Direction.bottom;

player2.Timer\_Move.Start();

break;

case Keys.Left:

player2.direction = Direction.left;

player2.Timer\_Move.Start();

break;

case Keys.Up:

player2.direction = Direction.top;

player2.Timer\_Move.Start();

break;

case Keys.Enter:

player2.Fire();

break;

}

}

static void Player2Move\_keyup(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Right || e.KeyCode == Keys.Down || e.KeyCode == Keys.Left || e.KeyCode == Keys.Up)

player2.Timer\_Move.Stop();

}

static void Press\_Interrupt(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

Interrupt();

if (e.KeyCode == Keys.P)

OnPause();

}

static void Press\_Nextlvl(object sender, KeyEventArgs e)

{

if(e.KeyCode == Keys.Enter)

{

player1\_total\_score += player1\_lvl\_score;

player2\_total\_score += player2\_lvl\_score;

player1\_lvl\_score = 0;

player2\_lvl\_score = 0;

form.Paint -= Draw\_Between\_lvl;

form.Paint += Draw\_Game;

form.KeyUp -= Press\_Nextlvl;

form.KeyDown += Press\_Interrupt;

if (player1 != null)

{

form.KeyDown += Player1Move\_keydown;

form.KeyUp += Player1Move\_keyup;

}

if (player2 != null)

{

form.KeyDown += Player2Move\_keydown;

form.KeyUp += Player2Move\_keyup;

}

Nextlvl();

}

}

static void Press\_WriteName(object sender, KeyPressEventArgs e)

{

if ((e.KeyChar >= 97 && e.KeyChar <= 122 && playername.Length<11) ||

(e.KeyChar >= 48 && e.KeyChar <= 57 && playername.Length < 11))

{

playername += e.KeyChar.ToString();

}

if (playername.Length > 0 && e.KeyChar == (Char)Keys.Back)

{

playername = playername.Substring(0, playername.Length - 1);

}

if (e.KeyChar == (Char)Keys.Enter)

{

if (TableRecords.Order\_To\_Add(player1\_total\_score))

{

TableRecords.Add(playername, player1\_total\_score);

player1\_total\_score = -1;

playername = "";

if (TableRecords.Order\_To\_Add(player2\_total\_score))

goto Exit;

}

if (TableRecords.Order\_To\_Add(player2\_total\_score))

TableRecords.Add(playername, player2\_total\_score);

form.Paint -= Draw\_Writename;

form.KeyPress -= Press\_WriteName;

form.Paint -= Draw\_Records;

Menu\_Records();

}

Exit:

form.Invalidate();

}

#endregion

#region Drawing

static void Draw\_Mainmenu(object sender, PaintEventArgs e)

{

Rectangle rec = new Rectangle(75, 50, 500, 100);

StringFormat sf = new StringFormat();

sf.Alignment = StringAlignment.Center;

sf.LineAlignment = StringAlignment.Center;

e.Graphics.DrawString("Танчики", new Font("Arial Black", 70), Brushes.DarkRed, rec, sf);

}

static void Draw\_Game(object sender, PaintEventArgs e)

{

Point[] flag = { new Point(7 \* 40 + 10, 13 \* 40 + 5), new Point(7 \* 40 + 35, 13 \* 40 + 13), new Point(7 \* 40 + 10, 13 \* 40 + 21) };

e.Graphics.FillPolygon(Brushes.Orange,flag);

e.Graphics.FillRectangle(Brushes.White, new Rectangle(7\*40+7,13\*40+5,3,35));

SolidBrush brush = new SolidBrush(Color.DarkGray);

e.Graphics.FillRectangle(brush, new Rectangle(0, 40, 40, 520));

e.Graphics.FillRectangle(brush, new Rectangle(0, 0, 560, 40));

e.Graphics.FillRectangle(brush, new Rectangle(0, 560, 560, 40));

e.Graphics.FillRectangle(brush, new Rectangle(560, 0, 100, 600));

StringFormat sf = new StringFormat();

sf.Alignment = StringAlignment.Center;

sf.LineAlignment = StringAlignment.Center;

e.Graphics.FillRectangle(Brushes.White, new Rectangle(580, 100 + 4, 8, 26));

e.Graphics.FillRectangle(Brushes.White, new Rectangle(580 + 22, 100 + 4, 8, 26));

e.Graphics.FillRectangle(Brushes.White, new Rectangle(580, 100 + 12, 30, 14));

e.Graphics.FillRectangle(Brushes.White, new Rectangle(580 + 13, 100, 4, 15));

e.Graphics.DrawString(tank\_for\_lvl.ToString(), new Font("Arial Black", 25), Brushes.White, new Rectangle(605, 93, 70, 50), sf);

e.Graphics.DrawString( "Уровень " + CurrentLevel.ToString() , new Font("Arial Black", 14), Brushes.White, new Rectangle(560, 193, 100, 50), sf);

e.Graphics.FillRectangle(Brushes.Goldenrod, new Rectangle(580, 300 + 4, 8, 26));

e.Graphics.FillRectangle(Brushes.Goldenrod, new Rectangle(580 + 22, 300 + 4, 8, 26));

e.Graphics.FillRectangle(Brushes.Goldenrod, new Rectangle(580, 300 + 12, 30, 14));

e.Graphics.FillRectangle(Brushes.Goldenrod, new Rectangle(580 + 13, 300, 4, 15));

e.Graphics.DrawString(player1.HP.ToString(), new Font("Arial Black", 25), Brushes.White, new Rectangle(600, 293, 70, 50), sf);

if (player2 != null)

{

e.Graphics.FillRectangle(Brushes.DarkOliveGreen, new Rectangle(580, 400 + 4, 8, 26));

e.Graphics.FillRectangle(Brushes.DarkOliveGreen, new Rectangle(580 + 22, 400 + 4, 8, 26));

e.Graphics.FillRectangle(Brushes.DarkOliveGreen, new Rectangle(580, 400 + 12, 30, 14));

e.Graphics.FillRectangle(Brushes.DarkOliveGreen, new Rectangle(580 + 13, 400, 4, 15));

e.Graphics.DrawString(player2.HP.ToString(), new Font("Arial Black", 25), Brushes.White, new Rectangle(600, 393, 70, 50), sf);

}

}

static void Draw\_Between\_lvl(object sender, PaintEventArgs e)

{

int sum1 = player1\_lvl\_score + player1\_total\_score;

int sum2 = player2\_lvl\_score + player2\_total\_score;

StringFormat sf = new StringFormat();

sf.Alignment = StringAlignment.Center;

sf.LineAlignment = StringAlignment.Center;

e.Graphics.DrawString("Уровень "+CurrentLevel.ToString()+" пройден", new Font("Arial Black", 40), Brushes.DarkRed, new Rectangle(60, 25, 500, 150), sf);

e.Graphics.DrawString("Заработано ранее", new Font("Arial Black", 15), Brushes.DarkRed, new Rectangle(85, 150, 250, 100), sf);

e.Graphics.DrawString(" За уровень", new Font("Arial Black", 15), Brushes.DarkRed, new Rectangle(310, 150, 150, 100), sf);

e.Graphics.DrawString("Всего", new Font("Arial Black", 15), Brushes.DarkRed, new Rectangle(460, 150, 100, 100), sf);

e.Graphics.DrawString("Игрок 1:", new Font("Arial Black", 15), Brushes.White, new Rectangle(0, 200, 110, 100), sf);

e.Graphics.DrawString(player1\_total\_score.ToString(), new Font("Arial Black", 15), Brushes.White, new Rectangle(85, 200, 250, 100), sf);

e.Graphics.DrawString(player1\_lvl\_score.ToString(), new Font("Arial Black", 15), Brushes.White, new Rectangle(310, 200, 150, 100), sf);

e.Graphics.DrawString(sum1.ToString(), new Font("Arial Black", 15), Brushes.White, new Rectangle(460, 200, 100, 100), sf);

if (player2\_total\_score != 0 || player2!=null)

{

e.Graphics.DrawString("Игрок 2:", new Font("Arial Black", 15), Brushes.White, new Rectangle(0, 300, 110, 100), sf);

e.Graphics.DrawString(player2\_total\_score.ToString(), new Font("Arial Black", 15), Brushes.White, new Rectangle(85, 300, 250, 100), sf);

e.Graphics.DrawString(player2\_lvl\_score.ToString(), new Font("Arial Black", 15), Brushes.White, new Rectangle(310, 300, 150, 100), sf);

e.Graphics.DrawString(sum2.ToString(), new Font("Arial Black", 15), Brushes.White, new Rectangle(460, 300, 100, 100), sf);

}

e.Graphics.DrawString("Нажмите Enter для перехода на следующий уровень", new Font("Arial Black", 15), Brushes.DarkRed, new Rectangle(115, 400, 400, 200), sf);

}

static void Draw\_Records(object sender, PaintEventArgs e)

{

StringFormat sf = new StringFormat();

sf.Alignment = StringAlignment.Far;

sf.LineAlignment = StringAlignment.Center;

e.Graphics.DrawString("Рекорды", new Font("Arial Black", 35), Brushes.DarkRed, new Rectangle(150, 20, 300, 50), sf);

e.Graphics.DrawString("Игрок", new Font("Arial Black", 14), Brushes.DarkRed, new Rectangle(160, 60, 100, 50), sf);

e.Graphics.DrawString("Счет", new Font("Arial Black", 14), Brushes.DarkRed, new Rectangle(350, 60, 100, 50), sf);

for (int i = 0; i < TableRecords.records.Count; i++)

{

if (TableRecords.records[i].score != 1)

{

e.Graphics.DrawString(TableRecords.records[i].name.ToString(), new Font("Arial Black", 12), Brushes.White, new Rectangle(100, 100 + 30 \* i, 150, 30), sf);

e.Graphics.DrawString(TableRecords.records[i].score.ToString(), new Font("Arial Black", 12), Brushes.White, new Rectangle(350, 100 + 30 \* i, 100, 30), sf);

}

}

}

static void Draw\_Writename(object sender, PaintEventArgs e)

{

StringFormat sf = new StringFormat();

sf.Alignment = StringAlignment.Center;

sf.LineAlignment = StringAlignment.Center;

if (TableRecords.Order\_To\_Add(player1\_total\_score))

e.Graphics.DrawString("Игрок 1 набрал "+ player1\_total\_score.ToString()+ " очков.",new Font("Arial Black",12), Brushes.DarkRed,new Rectangle( 125,400,400,50),sf);

else

e.Graphics.DrawString("Игрок 2 набрал " + player2\_total\_score.ToString() + " очков.", new Font("Arial Black", 12), Brushes.DarkRed, new Rectangle(125, 400, 400, 50), sf);

e.Graphics.DrawString("Введите имя для записи в таблицу.", new Font("Arial Black", 12), Brushes.DarkRed, new Rectangle(125, 425, 400, 50), sf);

e.Graphics.DrawString("(Имя должно содержать только символы латинского алфавита и цифры.)", new Font("Arial Black", 8), Brushes.DarkRed, new Rectangle(80, 450, 470, 50), sf);

e.Graphics.DrawString(playername, new Font("Arial Black", 15), Brushes.DarkRed, new Rectangle(80, 475, 470, 50), sf);

}

static void Draw\_Exit(object sender, PaintEventArgs e)

{

StringFormat sf = new StringFormat();

sf.Alignment = StringAlignment.Center;

sf.LineAlignment = StringAlignment.Center;

e.Graphics.DrawString("Выйти из игры?", new Font("Arial Black", 35), Brushes.DarkRed, new Rectangle(75, 40, 500, 50), sf);

}

static void Draw\_Pause(object sender, PaintEventArgs e)

{

StringFormat sf = new StringFormat();

sf.Alignment = StringAlignment.Center;

sf.LineAlignment = StringAlignment.Center;

e.Graphics.DrawString("Пауза", new Font("Arial Black", 30), Brushes.White,new Rectangle(150,300,300,50), sf);

}

#endregion

}

public delegate void DelButton\_Press();

public static class Menu

{

public static List<Button> buttons = new List<Button>();

public static Form1 form;

public static event DelButton\_Press button\_press;

public static void Add(Button button)

{

if (buttons.Count == 0)

form.KeyDown += OnPress;

form.Paint += button.Draw;

button\_press += button.press\_handler;

buttons.Add(button);

if (buttons.Count > 1)

buttons.Sort();

else

buttons[0].focus = true;

}

public static void Remove(string title)

{

foreach (var button in buttons)

{

if (button.title == title)

{

form.Paint -= button.Draw;

buttons.Remove(button);

break;

}

}

if (buttons.Count == 0)

form.KeyDown -= OnPress;

}

public static void Remove()

{

while (buttons.Count > 0)

{

form.Paint -= buttons[0].Draw;

button\_press -= buttons[0].press\_handler;

buttons.RemoveAt(0);

}

form.KeyDown -= OnPress;

}

static int Get\_focus()

{

foreach (var button in buttons)

if (button.focus == true)

return buttons.IndexOf(button);

return -1;

}

public static void OnPress(object sender, KeyEventArgs e)

{

int index = Get\_focus();

if(index!=-1)

switch (e.KeyCode)

{

case Keys.Up:

if (buttons[0].focus != true)

{

buttons[index].focus = false;

buttons[index - 1].focus = true;

form.Invalidate(new Rectangle(buttons[index].cordinates.x, buttons[index].cordinates.y,370,50));

form.Invalidate(new Rectangle(buttons[index - 1].cordinates.x, buttons[index - 1].cordinates.y, 370, 50));

}

break;

case Keys.Down:

if (buttons[buttons.Count - 1].focus != true)

{

buttons[index].focus = false;

buttons[index + 1].focus = true;

form.Invalidate(new Rectangle(buttons[index].cordinates.x, buttons[index].cordinates.y, 370, 50));

form.Invalidate(new Rectangle(buttons[index + 1].cordinates.x, buttons[index + 1].cordinates.y, 370, 50));

}

break;

case Keys.Enter:

{

button\_press();

form.Invalidate();

}

break;

}

}

}

public class Button: IComparable<Button>

{

public Coordinates cordinates;

public string title;

public Boolean focus = false;

public DelButton\_Press method;

public Button(Coordinates cordinates, string title, DelButton\_Press method)

{

this.cordinates = cordinates;

this.title = title;

this.method += method;

}

public int CompareTo(Button button)

{

if (this.cordinates.y < button.cordinates.y)

return -1;

if (this.cordinates.y > button.cordinates.y)

return 1;

else

return 0;

}

public void press\_handler()

{

if (focus)

method();

}

public void Draw(object sender, PaintEventArgs e)

{

Rectangle rec = new Rectangle(cordinates.x + 70, cordinates.y, 300, 50);

StringFormat sf = new StringFormat();

sf.Alignment = StringAlignment.Near;

sf.LineAlignment = StringAlignment.Center;

e.Graphics.DrawString(title, new Font("Times", 25), Brushes.White, rec, sf);

if (focus)

{

e.Graphics.FillRectangle(Brushes.Yellow, new Rectangle(cordinates.x , cordinates.y, 26, 8));

e.Graphics.FillRectangle(Brushes.Yellow, new Rectangle(cordinates.x , cordinates.y + 22, 26, 8));

e.Graphics.FillRectangle(Brushes.Yellow, new Rectangle(cordinates.x + 4 , cordinates.y, 14, 30));

e.Graphics.FillRectangle(Brushes.Yellow, new Rectangle(cordinates.x + 4 , cordinates.y + 13, 26, 4));

}

}

}

public struct Record: IComparable<Record>

{

public string name;

public int score;

public Record(string name, int score)

{

this.name = name;

this.score = score;

}

public int CompareTo(Record record)

{

if (this.score > record.score)

return -1;

if (this.score < record.score)

return 1;

else

return 0;

}

}

public static class TableRecords

{

public static List<Record> records = new List<Record>();

static TableRecords()

{

string line;

StreamReader sr = new StreamReader("Records.txt");

while ((line = sr.ReadLine()) != null)

{

records.Add(new Record(line,Int32.Parse(sr.ReadLine())));

}

sr.Close();

sr.Dispose();

records.Sort();

}

public static Boolean Order\_To\_Add(int score)

{

if (records[9].score < score)

return true;

else

return false;

}

public static void Add(string name,int score)

{

if (name == "")

name = "Unknown";

records.Add(new Record(name, score));

records.Sort();

if(records.Count>10)

records.RemoveAt(10);

Submit();

}

public static void Submit()

{

StreamWriter sw = new StreamWriter("Records.txt", false, Encoding.Default, 10);

foreach( var player in records)

{

sw.WriteLine(player.name);

sw.WriteLine(player.score);

}

sw.Close();

sw.Dispose();

}

public static void Clear()

{

records = new List<Record>();

for (int i = 0; i < 10; i++)

{

records.Add(new Record("", 1));

}

Submit();

}

}

public struct Coordinates

{

public int x;

public int y;

public Coordinates(int x, int y)

{

this.x = x;

this.y = y;

}

}

public enum Direction { top, right, bottom, left }

public abstract class GameObject

{

public Coordinates coordinates;

static public DelCollide collide;

static public DelPlayerInArea OnArea;

public GameObject(Coordinates cordinates)

{

this.coordinates = cordinates;

}

public abstract void Draw(object sender, PaintEventArgs e);

public void SearchBullet(Bullet bullet)

{

if (bullet.coordinates.x + 5 >= coordinates.x && bullet.coordinates.x + 5 <= coordinates.x + 30

&& bullet.coordinates.y + 5 >= coordinates.y && bullet.coordinates.y + 5 <= coordinates.y + 30)

collide(bullet, this);

}

public void SearchTank(PlayerTank tank)

{

if (tank.coordinates.x + 15 >= coordinates.x && tank.coordinates.x + 15 <= coordinates.x + 40

&& tank.coordinates.y + 15 >= coordinates.y && tank.coordinates.y + 15 <= coordinates.y + 40)

OnArea(this, tank);

}

}

public class Block: GameObject

{

public Block(Coordinates coordinates)

: base(coordinates)

{ }

public override void Draw(object sender, PaintEventArgs e)

{ }

}

public class Wall:GameObject

{

public int HP;

public Wall(Coordinates coordinates, int HP)

:base(coordinates)

{

this.HP = HP;

}

public override void Draw(object sender, PaintEventArgs e)

{

SolidBrush brush = null;

switch (HP)

{

case 2:

brush = new SolidBrush(Color.DarkRed);

break;

case 1:

brush = new SolidBrush(Color.Red);

break;

case 3:

brush = new SolidBrush(Color.Gray);

break;

}

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y, 40, 40));

}

}

class Water: GameObject

{

public Water(Coordinates coordinates)

:base(coordinates)

{ }

public override void Draw(object sender, PaintEventArgs e)

{

SolidBrush brush= new SolidBrush(Color.DarkBlue);

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y, 40, 40));

}

}

class Shrub: GameObject

{

public Shrub(Coordinates coordinates)

:base(coordinates)

{ }

public override void Draw(object sender, PaintEventArgs e)

{

SolidBrush brush = new SolidBrush(Color.DarkGreen);

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 4, 20, 12));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y, 12, 20));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 20, coordinates.y + 4, 20, 12));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 20 + 4, coordinates.y, 12, 20));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 4 + 20, 20, 12));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y + 20, 12, 20));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 20, coordinates.y + 4 + 20, 20, 12));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 20 + 4, coordinates.y + 20, 12, 20));

}

}

public class Ice:GameObject

{

public Ice(Coordinates coordinates)

:base(coordinates)

{ }

public override void Draw(object sender, PaintEventArgs e)

{

e.Graphics.FillRectangle(Brushes.Gainsboro, new Rectangle(coordinates.x, coordinates.y, 40, 40));

}

}

public enum BonusType { bomb, shield, live, lvl, timestop, beton }

public class Bonus:GameObject

{

public BonusType type;

public Bonus(Coordinates coordinates, BonusType type)

:base(coordinates)

{

this.type = type;

}

public override void Draw(object sender, PaintEventArgs e)

{

String name=null;

switch (type)

{

case BonusType.beton:

name = "Л";

break;

case BonusType.bomb:

name = "Б";

break;

case BonusType.live:

name = "З";

break;

case BonusType.lvl:

name = "У";

break;

case BonusType.shield:

name = "Щ";

break;

case BonusType.timestop:

name = "Ч";

break;

}

Rectangle rec = new Rectangle(coordinates.x,coordinates.y, 40,40);

e.Graphics.FillEllipse(Brushes.DarkBlue, rec);

StringFormat sf = new StringFormat();

sf.Alignment = StringAlignment.Center;

sf.LineAlignment = StringAlignment.Center;

e.Graphics.DrawString(name, new Font("Times", 25),Brushes.White,rec,sf);

}

}

public abstract class Tank:GameObject

{

public static int[,] field;

public int HP;

public int drive\_speed;

public int bullet\_speed;

public int lvl;

public DateTime last\_shot = DateTime.Now;

public Timer Timer\_Move = new Timer();

public static DelNewBullet NewBullet;

public Direction direction = Direction.top;

public event DelTankMove TankMove;

public Tank(Coordinates coordinates)

:base(coordinates)

{

this.coordinates = coordinates;

Timer\_Move.Tick += Timer\_move\_tick;

}

public virtual void Fire()

{

Bullet bullet = new Bullet(new Coordinates(this.coordinates.x, this.coordinates.y), bullet\_speed, lvl, direction, this);

NewBullet(bullet);

last\_shot = DateTime.Now;

}

public virtual void Timer\_move\_tick(object sender, EventArgs e)

{

if (TankMove != null) TankMove(this);

}

}

public enum Stage { random, to\_Player, to\_Headquarters }

public class EnemyTank:Tank

{

public SolidBrush brush;

public Color color;

public Timer timer\_for\_color = new Timer();

public Timer timer\_for\_stage = new Timer();

public Timer timer\_for\_fire = new Timer();

public Timer timer\_for\_handler = new Timer();

public Boolean bonustank;

public List<Coordinates> way = new List<Coordinates>();

public Stage stage = Stage.random;

public static PlayerTank player1;

public static PlayerTank player2;

public static DelForPLace Place;

int tick;

public EnemyTank(Coordinates cordinates, int lvl, int interval,Boolean bonustank =false)

:base(cordinates)

{

this.lvl = lvl;

this.bonustank = bonustank;

switch (lvl)

{

case 1:

HP = 1;

drive\_speed = 1;

bullet\_speed = 1;

break;

case 2:

HP = 1;

drive\_speed = 2;

bullet\_speed = 1;

break;

case 3:

HP = 1;

drive\_speed = 1;

bullet\_speed = 2;

break;

case 4:

HP = 4;

drive\_speed = 1;

bullet\_speed = 1;

break;

}

tick=10/drive\_speed;

if (lvl == 4)

color = Color.Gray;

else

color = Color.White;

brush = new SolidBrush(color);

if (bonustank)

{

timer\_for\_color.Interval = 500;

timer\_for\_color.Tick += Change\_color;

timer\_for\_color.Enabled = true;

}

timer\_for\_stage.Interval = interval;

timer\_for\_stage.Tick += Timer\_stage\_tick;

timer\_for\_stage.Start();

timer\_for\_fire.Interval = 2000;

timer\_for\_fire.Tick += Timer\_fire\_tick;

timer\_for\_fire.Start();

timer\_for\_handler.Interval = 200;

timer\_for\_handler.Tick += Timer\_handler\_tick;

timer\_for\_handler.Start();

Timer\_Move.Interval = 60;

Timer\_Move.Start();

}

public void EnemyMove()

{

try

{

if (((coordinates.x + 15) / 40) == way[0].x && ((coordinates.y + 15) / 40) == way[0].y)

{

way.RemoveAt(0);

}

if ((coordinates.x + 15) / 40 < way[0].x)

direction = Direction.right;

if ((coordinates.x + 15) / 40 > way[0].x)

direction = Direction.left;

if ((coordinates.y + 15) / 40 < way[0].y)

direction = Direction.bottom;

if ((coordinates.y + 15) / 40 > way[0].y)

direction = Direction.top;

}

catch(ArgumentOutOfRangeException)

{

Timer\_Move.Stop();

tick = 10/drive\_speed;

}

}

public override void Draw(object sender, PaintEventArgs e)

{

switch (lvl)

{

case 1:

Drawlvl1(sender, e);

break;

case 2:

Drawlvl2(sender, e);

break;

case 3:

Drawlvl3(sender, e);

break;

case 4:

Drawlvl4(sender, e);

break;

}

}

public void ChangeColor()

{

switch (HP)

{

case 1:

brush.Color = Color.White;

break;

case 2:

brush.Color = Color.Gainsboro;

break;

case 3:

brush.Color = Color.DarkGray;

break;

case 4:

brush.Color = Color.Gray;

break;

}

}

public override void Fire()

{

if (DateTime.Now.Subtract(last\_shot).TotalMilliseconds >= 2000)

base.Fire();

}

public Boolean Order\_To\_Fire()

{

int x = (coordinates.x + 15) / 40;

int y = (coordinates.y + 15) / 40;

int \_x = 0;

int \_y = 0;

switch (direction)

{

case Direction.top:

\_x = 0;

\_y = -1;

break;

case Direction.right:

\_x = 1;

\_y = 0;

break;

case Direction.bottom:

\_x = 0;

\_y = 1;

break;

case Direction.left:

\_x = -1;

\_y = 0;

break;

}

while (true)

{

if (x == 7 && y == 13)

return true;

if (field[x, y] == -1)

return false;

if (player1 != null && x == (player1.coordinates.x + 15) / 40 && y == (player1.coordinates.y + 15) / 40)

return true;

if (player2 != null && x == (player2.coordinates.x + 15) / 40 && y == (player2.coordinates.y + 15) / 40)

return true;

x += \_x;

y += \_y;

}

}

private void Drawlvl1(object sender, PaintEventArgs e)

{

switch (direction)

{

case Direction.top:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 4, 8, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 22, coordinates.y + 4, 8, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 12, 30, 14));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 13, coordinates.y, 4, 15));

break;

case Direction.right:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y, 26, 8));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 22, 26, 8));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y, 14, 30));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y + 13, 26, 4));

break;

case Direction.bottom:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y, 8, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 22, coordinates.y, 8, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 4, 30, 14));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 13, coordinates.y + 4, 4, 26));

break;

case Direction.left:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y, 26, 8));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y + 22, 26, 8));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 12, coordinates.y, 14, 30));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 13, 26, 4));

break;

}

}

private void Drawlvl2(object sender, PaintEventArgs e)

{

switch (direction)

{

case Direction.top:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 4, 8, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 22, coordinates.y + 4, 8, 26)); //faster

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 8, 30, 14));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 13, coordinates.y, 4, 15));

break;

case Direction.right:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y, 26, 8));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 22, 26, 8));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 8, coordinates.y, 14, 30));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 15, coordinates.y + 13, 15, 4));

break;

case Direction.bottom:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y, 8, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 22, coordinates.y, 8, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 8, 30, 14));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 13, coordinates.y + 16, 4, 14));

break;

case Direction.left:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y, 26, 8));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y + 22, 26, 8));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 8, coordinates.y, 14, 30));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 13, 15, 4));

break;

}

}

private void Drawlvl3(object sender, PaintEventArgs e)

{

switch (direction)

{

case Direction.top:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 4, 8, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 22, coordinates.y + 4, 8, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 12, 30, 14));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 12, coordinates.y, 6, 15));

break;

case Direction.right:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y, 26, 8));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 22, 26, 8));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y, 14, 30));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y + 12, 26, 6));

break;

case Direction.bottom:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y, 8, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 22, coordinates.y, 8, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 4, 30, 14));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 12, coordinates.y + 4, 6, 26));

break;

case Direction.left:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y, 26, 8));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y + 22, 26, 8));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 12, coordinates.y, 14, 30));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 12, 26, 6));

break;

}

}

private void Drawlvl4(object sender, PaintEventArgs e)

{

switch (direction)

{

case Direction.top:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 4, 9, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 21, coordinates.y + 4, 9, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 5, coordinates.y + 12, 20, 20));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 12, coordinates.y, 6, 15));

break;

case Direction.right:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y, 26, 9));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 21, 26, 9));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x - 2, coordinates.y + 5, 20, 20));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y + 12, 26, 6));

break;

case Direction.bottom:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y, 9, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 21, coordinates.y, 9, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 5, coordinates.y - 2, 20, 20));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 12, coordinates.y + 4, 6, 26));

break;

case Direction.left:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y, 26, 9));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y + 21, 26, 9));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 12, coordinates.y + 5, 20, 20));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 12, 26, 6));

break;

}

}

private void Change\_color(object sender, EventArgs e)

{

if (brush.Color == Color.Red)

brush.Color = color;

else

brush.Color = Color.Red;

}

public List<Coordinates> Tracker( Coordinates target\_coordinates)

{

Coordinates tank\_cordinates = new Coordinates((coordinates.x + 15) / 40, (coordinates.y + 15) / 40);

Coordinates target = new Coordinates((target\_coordinates.x + 15) / 40, (target\_coordinates.y + 15) / 40);

if (tank\_cordinates.x == target.x && tank\_cordinates.y == target.y)

return new List<Coordinates>();

var \_field = (int[,])field.Clone();

int step = 1;

\_field[tank\_cordinates.x, tank\_cordinates.y] = 1;

while (true)

{

for (int x = 1; x < 14; x++)

for (int y = 1; y < 14; y++)

if (\_field[x, y] == step)

{

if (\_field[x, y - 1] == 0)

\_field[x, y - 1] = step + 1;

if (\_field[x - 1, y] == 0)

\_field[x - 1, y] = step + 1;

if (\_field[x, y + 1] == 0)

\_field[x, y + 1] = step + 1;

if (\_field[x + 1, y] == 0)

\_field[x + 1, y] = step + 1;

}

step++;

if (\_field[target.x, target.y] > 0)

break;

if (step > 13 \* 13)

return new List<Coordinates>();

}

int \_x = target.x;

int \_y = target.y;

int[] borders = new int[4];

List<Coordinates> way = new List<Coordinates>();

way.Add(new Coordinates(\_x, \_y));

while (true)

{

if (\_field[\_x - 1, \_y] > 0)

borders[0] = \_field[\_x - 1, \_y];

else

borders[0] = 100000;

if (\_field[\_x + 1, \_y] > 0)

borders[1] = \_field[\_x + 1, \_y];

else

borders[1] = 100000;

if (\_field[\_x, \_y - 1] > 0)

borders[2] = \_field[\_x, \_y - 1];

else

borders[2] = 100000;

if (\_field[\_x, \_y + 1] > 0)

borders[3] = \_field[\_x, \_y + 1];

else

borders[3] = 100000;

Array.Sort(borders);

if (borders[0] == \_field[\_x - 1, \_y])

{

way.Add(new Coordinates(\_x - 1, \_y));

\_x--;

}

else if (borders[0] == \_field[\_x + 1, \_y])

{

way.Add(new Coordinates(\_x + 1, \_y));

\_x++;

}

else if (borders[0] == \_field[\_x, \_y - 1])

{

way.Add(new Coordinates(\_x, \_y - 1));

\_y--;

}

else if (borders[0] == \_field[\_x, \_y + 1])

{

way.Add(new Coordinates(\_x, \_y + 1));

\_y++;

}

if (tank\_cordinates.x == \_x && tank\_cordinates.y == \_y)

break;

}

way.Reverse(0, way.Count);

way.RemoveAt(0);

return way;

}

public void Find\_Players()

{

List<Coordinates> way1 = new List<Coordinates>();

List<Coordinates> way2 = new List<Coordinates>();

if (way.Count == 0 && stage == Stage.random)

{

way = Tracker( Place());

Timer\_Move.Start();

return;

}

if (stage == Stage.to\_Player)

{

if (player1 != null)

way1 = Tracker( player1.coordinates);

if (player2 != null)

way2 = Tracker( player2.coordinates);

if (player1 != null && player2 != null)

{

if (way1.Count > way2.Count)

way = way2;

else

way = way1;

}

else

{

if (player1 != null)

way = way1;

if (player2 != null)

way = way2;

}

Timer\_Move.Start();

return;

}

if (stage == Stage.to\_Headquarters)

{

way1 = Tracker( new Coordinates(5 \* 40, 13 \* 40));

way2 = Tracker( new Coordinates(9 \* 40, 13 \* 40));

if (way1.Count > way2.Count)

way = way2;

else

way = way1;

if (way.Count == 0 && coordinates.y > 40 \* 13)

if (coordinates.x < 40 \* 7)

direction = Direction.right;

else

direction = Direction.left;

Fire();

}

}

public override void Timer\_move\_tick(object sender, EventArgs e)

{

if (tick == 10/drive\_speed)

{

tick = 0;

EnemyMove();

}

if (Timer\_Move.Enabled != false)

{

tick++;

switch (direction)

{

case Direction.top:

coordinates.y -= 4 \* drive\_speed;

break;

case Direction.right:

coordinates.x += 4 \* drive\_speed;

break;

case Direction.bottom:

coordinates.y += 4 \* drive\_speed;

break;

case Direction.left:

coordinates.x -= 4 \* drive\_speed;

break;

}

}

}

public void Timer\_stage\_tick(object sender, EventArgs e)

{

if (stage != Stage.to\_Headquarters)

stage++;

}

public void Timer\_fire\_tick(object sender, EventArgs e)

{

Random rnd= new Random();

if (rnd.Next(2) == 0)

Fire();

}

public void Timer\_handler\_tick(object sender, EventArgs e)

{

Find\_Players();

if (Order\_To\_Fire())

Fire();

}

}

public class PlayerTank: Tank

{

public int bullets\_count = 0;

public Coordinates startposition;

public Boolean shield = false;

public Boolean second = false;

public Timer timer\_for\_reload = new Timer();

public Timer Timer\_Slipping = new Timer();

public PlayerTank(Coordinates coordinates, int lvl, Boolean second = false)

:base(coordinates)

{

this.lvl = lvl;

HP = 3;

drive\_speed = 1;

bullet\_speed = 1;

Timer\_Move.Interval = 60;

this.coordinates = coordinates;

startposition = coordinates;

this.second = second;

timer\_for\_reload.Interval = 2000;

timer\_for\_reload.Tick += Reload;

timer\_for\_reload.Enabled = true;

Timer\_Slipping.Interval = 20;

Timer\_Slipping.Tick += Timer\_slipping\_tick;

}

public void Lvl\_Up()

{

lvl++;

if (lvl == 2)

bullet\_speed++;

}

public override void Draw(object sender, PaintEventArgs e)

{

SolidBrush brush = new SolidBrush(Color.Goldenrod);

if (second)

brush = new SolidBrush(Color.DarkOliveGreen);

if (shield)

brush = new SolidBrush(Color.DarkGray);

switch (direction)

{

case Direction.top:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 4, 8, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 22, coordinates.y + 4, 8, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 12, 30, 14));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 13, coordinates.y, 4, 15));

break;

case Direction.right:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y, 26, 8));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 22, 26, 8));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y, 14, 30));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y + 13, 26, 4));

break;

case Direction.bottom:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y, 8, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 22, coordinates.y, 8, 26));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 4, 30, 14));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 13, coordinates.y + 4, 4, 26));

break;

case Direction.left:

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y, 26, 8));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 4, coordinates.y + 22, 26, 8));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x + 12, coordinates.y, 14, 30));

e.Graphics.FillRectangle(brush, new Rectangle(coordinates.x, coordinates.y + 13, 26, 4));

break;

}

}

void Reload(object sender, EventArgs e)

{

if (lvl >= 3)

bullets\_count = 3;

else

bullets\_count = 1;

}

public override void Fire()

{

if (bullets\_count != 0 && ((DateTime.Now.Subtract(last\_shot).TotalMilliseconds >= 700 && lvl >= 3)

|| (DateTime.Now.Subtract(last\_shot).TotalMilliseconds >= 2000 && lvl < 3)))

{

bullets\_count--;

last\_shot = DateTime.Now;

base.Fire();

}

}

public Boolean Can\_It\_Go\_to()

{

int x;

int y;

int block\_size = 40;

int tank\_size = 30;

switch (direction)

{

case Direction.left:

x = (coordinates.x + block\_size - 5) / block\_size;

y = coordinates.y;

if (field[x - 1, y / block\_size] == -1 || field[x - 1, (y + tank\_size) / block\_size] == -1)

return false;

break;

case Direction.right:

x = (coordinates.x + tank\_size - block\_size + 5) / block\_size;

y = coordinates.y;

if (field[x + 1, y / block\_size] == -1 || field[x + 1, (y + tank\_size) / block\_size] == -1)

return false;

break;

case Direction.top:

x = coordinates.x;

y = (coordinates.y + block\_size - 5) / block\_size;

if (field[x / block\_size, y - 1] == -1 || field[(x + tank\_size) / block\_size, y - 1] == -1)

return false;

break;

case Direction.bottom:

x = coordinates.x;

y = (coordinates.y + tank\_size - block\_size + 5) / block\_size;

if (field[x / block\_size, y + 1] == -1 || field[(x + tank\_size) / block\_size, y + 1] == -1)

return false;

break;

}

return true;

}

public override void Timer\_move\_tick(object sender, EventArgs e)

{

if (Can\_It\_Go\_to())

{

switch (direction)

{

case Direction.top:

coordinates.y -= 4 \* drive\_speed;

break;

case Direction.right:

coordinates.x += 4 \* drive\_speed;

break;

case Direction.bottom:

coordinates.y += 4 \* drive\_speed;

break;

case Direction.left:

coordinates.x -= 4 \* drive\_speed;

break;

}

base.Timer\_move\_tick(sender,e);

}

}

public void Timer\_slipping\_tick(object sender, EventArgs e)

{

if (Can\_It\_Go\_to() && Timer\_Move.Enabled==false)

{

switch (direction)

{

case Direction.top:

coordinates.y -= 1;

break;

case Direction.right:

coordinates.x += 1;

break;

case Direction.bottom:

coordinates.y += 1;

break;

case Direction.left:

coordinates.x -= 1;

break;

}

}

Timer\_Slipping.Stop();

base.Timer\_move\_tick(sender,e);

}

}

public class Bullet:GameObject

{

public int speed;

public int lvl;

Direction direction;

public Timer timer = new Timer();

public Tank tank;

public event DelBulletMove BulletMove;

public Bullet(Coordinates coordinates,int speed, int lvl,Direction direction, Tank tank)

:base(coordinates)

{

this.speed = speed\*10;

this.coordinates.x += 6;

this.coordinates.y += 6;

this.lvl = lvl;

this.direction = direction;

this.tank = tank;

timer.Interval = 40;

timer.Tick += Timer\_tick;

timer.Enabled = true;

}

public override void Draw (object sender, PaintEventArgs e)

{

SolidBrush brush = new SolidBrush(Color.White);

e.Graphics.FillEllipse(brush, coordinates.x, coordinates.y, 10, 10);

}

void Timer\_tick(object sender, EventArgs e)

{

switch (direction)

{

case Direction.top:

coordinates.y -= speed;

break;

case Direction.right:

coordinates.x += speed;

break;

case Direction.bottom:

coordinates.y += speed;

break;

case Direction.left:

coordinates.x -= speed;

break;

}

if(BulletMove!=null)

BulletMove(this);

}

}