Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Операционные системы и системное программирование (ОСиСП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

по курсовому проекту
на тему

Аркадная игра “Tanks 2d”

БГУИР КП 1-40 01 01 623 ПЗ

Выполнил
студент: гр. 851006 Одиноченко М.И.

Проверил: Жиденко А.Л.

Минск 2020

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой ПОИТ
_____
(подпись)
Лапицкая Н.В. 2020г.

ЗАДАНИЕ
по курсовому проектированию

Студенту *Одиноченко Максиму Игоревичу*

1. Тема работы Аркадная игра “Tanks 2d”

2. Срок сдачи законченной работы *08.12.2020г.*

3. Исходные данные к работе *Среда программирования Visual Studio 2020.*

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

Введение

1 Анализ предметной области

2 Постановка задачи

3 Разработка программного средства

4 Тестирование и проверка работоспособности программного средства

5 Руководство по использованию программного средства

Заключение

Список использованных источников

Приложения

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

Схема алгоритма в формате А1

6. Консультант по курсовой работе *Жиденко А.Л.*

7. Дата выдачи задания *01.10.2020*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):

Раздел 1. Введение к 20.10.2020г – 10 % готовности работы;

Раздел 2 к 28.10.2020г. – 30% готовности работы

Раздел 3 к 15.11.2020г. – 60% готовности работы

Раздел 4 к 23.11.2020г. – 80% готовности работы

Раздел 5.Заключение. Приложения к 01.12.2020г. – 90% готовности работы;

Оформление пояснительной записки и графического материала к 05.12.2020г. – 100% готовности работы.

Защита курсового проекта с 01.12.2020г. по 10.12.2020г

РУКОВОДИТЕЛЬ _____ *Жиденко А.Л.*

(подпись)

Задание принял к исполнению _____ *Одиноченко М.И.*

08.12.2020г.

(дата и подпись студента)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данный курсовой проект предназначен для реализации игры такого жанра, как аркада. Основная идея заключается в достаточно быстром и при том насыщенном игровом процессе. Зачастую суть игры очень проста и понятна с первого взгляда, а сюжета нет как такового.

На данные момент существуют и разрабатываются множество игр данного жанра. По причине очень простой разработки со стороны программиста а так же большой популярности, а вместе с тем и прибыли, со стороны пользователей, жанр стабильно востребован и считается “классикой” среди игр. По причине крайней простоты геймплея, для любого приложения найдутся случайные и не очень аналоги в огромном количестве, однако основным я считаю одноимённую игру “Tank Shoot 2D”.

Целью работы является создание простой аркадной игры, реализующей основные возможности игр такого жанра в виде полностью подконтрольного игрока, окружения с ненулевыми возможностями для взаимодействия и противника, задающего смысл всей игры, придавая ей напряжённость и интерес своими действиями.

Главная причина актуальности данного приложения, как и всех остальных игр жанра, это низкий порог вхождения, обычно заключающийся в простоте управления и понимания игры, низкий занимаемый объём оперативной и обычной памяти. Вместе с тем даже с помощью самых разных аркад можно получить удовольствие от победы, либо просто скоротать время ожидания. Разработчики же любят этот жанр по сугубо экономическим аспектам. Трудозатраты крайне малы по сравнению с другими играми, а рекламу либо другие способы монетизации продукта вставить очень просто и выгодно. Так же приложение не обязательно должно вообще иметь взаимодействие с Интернетом, что позволяет разработчикам при необходимости не иметь никаких серверов для выполнения вычислений, что для маленьких групп или компаний может играть важную роль.

Геймплей в данной работе основан на простом принципе противостояния 2 игроков на карте, имеющей препятствия и ловушки.

Приложение проектируется максимально долговечным при небольших ресурсозатратах, и общая структура выполнена умеренно просто. В реализации проекта решаются следующие задачи:

- движение спрайтов;
- столкновение с препятствиями;
- ловушки на локации с разными эффектами;
- стрельба;
- противника-бот с такими же возможностями как у игрока.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Анализ существующих аналогов

Как и было сказано во введении, аналогов у данного приложения множество, однако первым мы рассмотрим “ Tank Shoot 2D ”.

1.1.1 Tank Shoot 2D

Tank Shoot 2D - это лучшая кооперативная аркадная игра для 1 или 2 игроков. Вы сражаетесь, чтобы защитить свой флаг города - если вражеские танки стреляют по флагу, игра окончена. Убейте всех врагов, чтобы перейти на следующий этап. Собирайте предметы, чтобы получить особые способности. Новые уровни приносят новых и более умных врагов. А новые уровни приносят новые типы стен и новые испытания.

Эта игра с классической игровой механикой вернет детские воспоминания и подарит вам бесчисленные часы веселья и развлечений. Приложение имеет следующий вид (рис. 1.1)

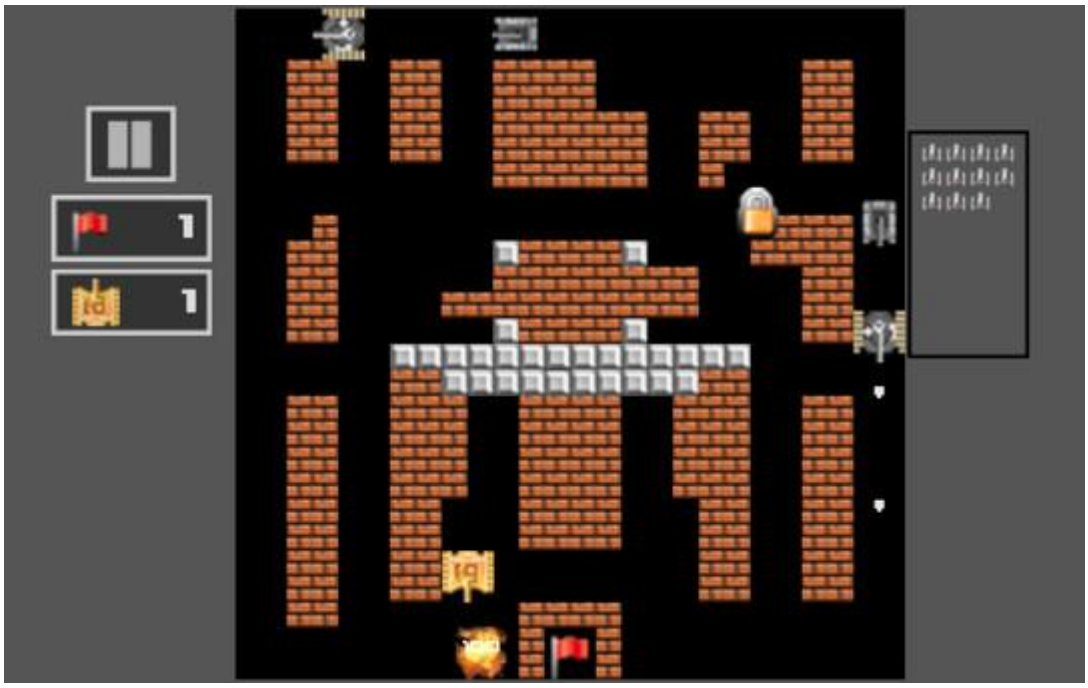

Рисунок 1.1 – Интерфейс игры “Tank Shoot 2D”

В общем, это довольно классическая игра, содержащая все основные функции и возможности аркад: игрока, различных врагов, разрушаемое окружение, подбираемые предметы. По причине довольно большого возраста игры, интерфейс не выделяется большой чёткостью или красочностью, однако игра не стала со временем чем-то абсолютно ненужным.

В огромном множестве аналогов встречаются довольно хорошо

развитые идеи и наработки данной игры, где-то даже собственные особенности, однако аркадные игры решает не их вид, а то, какие эмоции от них получает конечный пользователь, то есть игрок. Если одному графика мешает, что другому даёт чувство ностальгии.

1.1.2 Pocket Tanks

компьютерная игра в жанре пошаговая стратегия. Два танка, принадлежащих двум разным игрокам, по очереди ведут перестрелку. (рис. 1.2)

Рисунок 1.2 – “Pocket Tanks”

Перед началом игры игроки выбирают оружие. Для этого формируется список из 20 видов оружия, которые могут повторяться. Всего в Shareware-версии игры 30 видов, в полной Deluxe — 320 видов на конец 2016 года (28 паков оружий скачиваются на официальном сайте Blitwise и ставятся поверх Deluxe). Оружие различается по воздействию (вид выстрела и количество очков за попадание) и может служить не только для зарабатывания очков (например, измеритель углов — Tracer).

В начале игры случайным образом генерируется карта, в разных концах которой располагаются танки. Каждый игрок (человек или компьютер) управляет одним из этих танков. Танк может четыре раза за всю партию передвигаться по карте на ограниченное расстояние. Однако, танк может проехать не по всякой местности: чересчур крутые склоны он не сможет

преодолеть.

Перед выстрелом игрок выбирает оружие, устанавливает угол наклона ствола и силу выстрела. После выстрела оружие исчезает из списка. Количество ходов в игре ограничено количеством оружия (20 ходов). Снаряд, вылетевший за левый или правый угол экрана, пропадает.

Победу одерживает тот игрок, который в конце раунда набрал большее количество очков. Очки начисляются за попадание во вражеский танк и снимаются за попадание в свой собственный.

В данной игре так же 2 измерения, вот только они используются абсолютно по другому. Обычный обмен “снарядами” превращается в полномасштабное сражение двух артиллеристов. Этот эффект достигается физикой движения снарядов, выпущенных с определённой силой и под определённым углом, а так же самим выбором снаряда для стрельбы. Изменение каждого из параметров может повлиять на полёт снаряда, к тому же, некоторым типам снарядов и вовсе не требуется точное попадание, а иногда и вовсе прицел. В таких условиях попасть по противнику и не попасться самому становится очень увлекательной и нетривиальной задачей, приятно вовлекающей в себя.

В игре так же имеются настройки для определения формы карты, типов вооружения и противника. В ней даже можно устроить борьбу двух реальных игроков, а не только ботов, как с одного, так и с разных компьютеров, что ещё больше раскрывает потенциал игры.

1.1.3 Танки Онлайн

Ещё один аналог игры, на этот раз созданный исключительно для пользования через браузер. Приложение гораздо моложе чем предыдущие аналоги. (рис. 1.3)

Рисунок 1.2 – “Pocket Tanks”

Игра "Танки Онлайн" обедняет реальных игроков по всему миру, которые могут играть в виртуальном мире, управляя боевыми танками разных типов. Игровой процесс основан на системе "ПВП" - игрок против игрока. При уничтожении вражеского танка, игрок получает опыт и "фраги".

Танки Онлайн - это многопользовательская браузерная онлайн-игра, являющаяся танковым симулятором. Крутая графика, реалистичное управление и куча возможностей не дадут заскучать никому! Вас ждут массовые сражения на больших картах, которые дают возможность каждому игроку проявить способности и доказать свое превосходство в реальном бою. Что может быть круче?

Выбирайте свою боевую машину и приступайте к бою! Все игроки вступают в битвы с другими пользователями и зарабатывает игровую валюту игры — кристаллы! В игре существует несколько командных режимов и разное оружие для танков. После выезда в бой, нужно вести себя очень осторожно. Стреляйте по любому из ваших противников, попытайтесь подбить их танки и сохранить прочность своего танка. Игроки могут общаться между собой, как в чате, так и при помощи голосовой связи. В игре существует система воинских званий, которые достигаются при получении очков опыта. Высокие звания дадут возможность улучшений вашего танка, оружия и прочего. Чтобы играть, вам нужно изначально пройти обучение, затем зарегистрироваться и приступить к бою. Игра проходит на красочных и больших картах, где могут воевать не только парни, но и девушки. На полях боя расположены разные постройки, деревья, заборы и прочее.

Игра "Танки Онлайн" основана на реальных законах физики. Ваш танк может упасть с моста, или же перевернуться. Простое управление, несколько игровых режимов и куча других возможностей. Все это ждет вас прямо сейчас!

1.2 Постановка задачи

Для того, чтобы программное средство можно было считать аркадой, способной к конкуренции с существующими аналогами, на основе анализа популярных клиентов, в конечном приложении необходимо наличие следующих возможностей:

- движение спрайтов;
- столкновение с препятствиями;
- ловушки на локации с разными эффектами;
- стрельба;
- противника-бот с такими же возможностями как у игрока;
- система игры, завершающаяся с здоровьем одного из игроков

2 МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

2.1 Описание предметной области

Игры всегда были популярным средством развлечения. Они помогали отвлечься, поднять настроение, узнать что-то новое, пообщаться с другими людьми. Аркады не приносят существенной пользы в виде полезных навыков или знаний, однако дают пользователю приятные ощущения победы в тяжёлой схватке и вместе с тем никак не обязывают игрока тратить много времени.

По этой причине такой жанр как аркады подходит очень многим людям, даже очень занятым. Так же, по некоторым исследованиям, такие игры развивают реакцию и мышление, что может быть полезно для детей.

2.2 Информационная база задачи

Весь проект написан на WinAPI, что подразумевает работу на более низком уровне, чем ныне актуальный. Windows API (англ. application programming interfaces) — общее наименование набора базовых функций интерфейсов программирования приложений операционных систем семейств Microsoft Windows корпорации «Майкрософт». Предоставляет прямой способ взаимодействия приложений пользователя с операционной системой Windows. Для создания программ, использующих Windows API, корпорация «Майкрософт» выпускает комплект разработчика программного обеспечения, который называется Platform SDK и содержит документацию, набор библиотек, утилит и других инструментальных средств для разработки.

Поскольку рисование в данной программе основано не на встроенных функциях для рисования графических примитивов, а на рисовании битмапов, далее информация непосредственно про них.

Структура BITMAP определяет тип, ширину, высоту, формат цвета и битовые значения растрового изображения.

```
typedef struct tagBITMAP {
  LONG   bmType;
  LONG   bmWidth;
  LONG   bmHeight;
  LONG   bmWidthBytes;
  WORD   bmPlanes;
  WORD   bmBitsPixel;
  LPVOID bmBits;
} BITMAP, *PBITMAP, *NPBITMAP, *LPBITMAP;
```

bmType – Тип растрового изображения. Этот член должен быть нулевым.

bmWidth – Ширина растрового изображения в пикселях. Ширина должна быть больше нуля.

bmHeight – Высота растрового изображения в пикселях. Высота должна быть больше нуля.

bmWidthBytes – Количество байтов в каждой строке сканирования. Это значение должно делиться на 2, потому что система предполагает, что битовые значения растрового изображения образуют массив, выровненный по словам.

bmPlanes – Подсчет цветовых плоскостей.

bmBitsPixel – Количество битов, необходимых для обозначения цвета пикселя.

bmBits – Указатель на расположение битовых значений для растрового изображения. BmBits член должен быть указатель на массив символов (1 байт) значений.

Функция TransparentBlt выполняет передачу битовых блоков данных цвета, соответствующих прямоугольнику пикселей, из указанного исходного контекста устройства в контекст целевого устройства. Данная функция используется довольно часто для отрисовки битмапа без его фона, что довольно важно для игры.

```
BOOL TransparentBlt(
  HDC  hdcDest,
  int  xoriginDest,
  int  yoriginDest,
  int  wDest,
  int  hDest,
  HDC  hdcSrc,
  int  xoriginSrc,
  int  yoriginSrc,
  int  wSrc,
  int  hSrc,
  UINT crTransparent
);
```

Так же в приложении необходима реализация поворота самих игроков. Поскольку из встроенных в WinAPI функций для этого подходит только функция SetWorldTransform/ GetWorldTransform, которые значительно усложняют задание координат для отрисовки объектов, необходимо создать собственную функцию поворота основанную на GetBitmapBits/SetBitmapBits.

Функция GetBitmapBits копирует биты битовой карты указанного зависящего от устройства битовой карты в буфер.

Функция SetBitmapBits устанавливает биты данных цвета для растрового изображения в указанные значения.

3 ПРОЕКТИРОАВНИЕ ПРОГРАММНОГО СРЕДСТВА

3.1 Схема алгоритма обработки сообщения WM_KEYDOWN

Поскольку нам обязательно нужен подконтрольный игрок, для него должно быть управление, в нашем случае обработка WM_KEYDOWN, в котором устанавливается задержка для действия и совершается само действие (рис. 3.1). Задержка крайне важна для соблюдения “ритма” игры и некой стандартизации реакций.

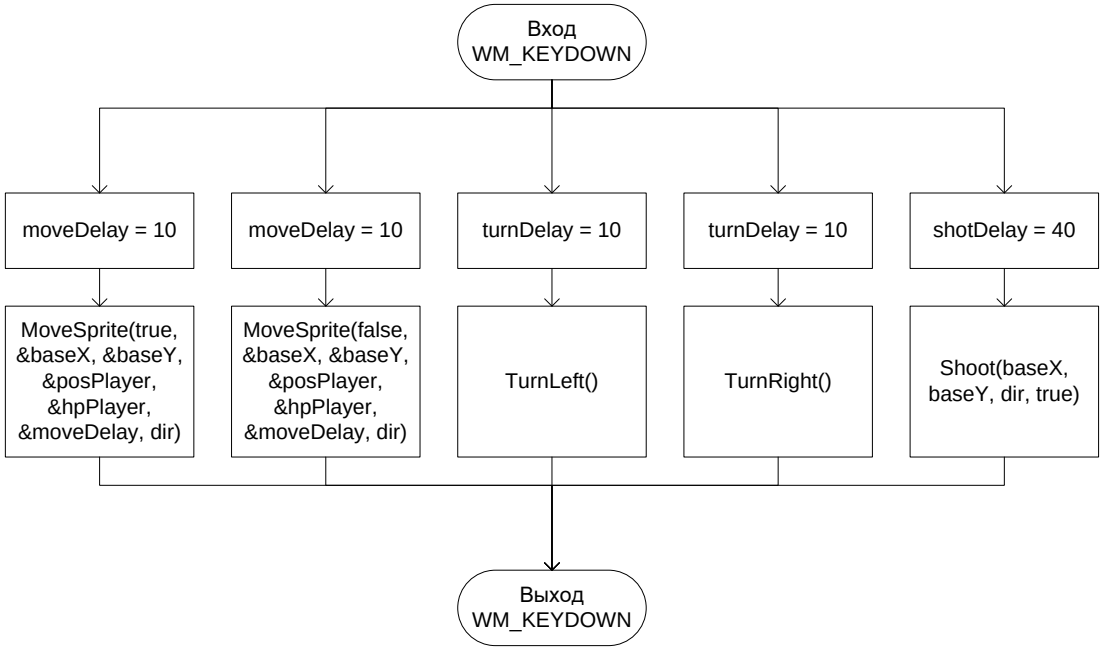

Рисунок 3.1 – Схема алгоритма обработки сообщения WM_KEYDOWN

3.2 Схема алгоритма метода TurnRight

Для функционирования приложения необходим метод поворота картинки (рис. 3.2). Он выполняется прямой заменой битов. Так же функция модифицирует глобальную переменную, характеризующую направление игрока. Данный алгоритм выполняет поворот направо битмапа игрока, однако есть аналогичные функции поворота налево, а так же поворотов для битмапа врага, которые очень схожи с данной, а потому не будут отображены в схемах.

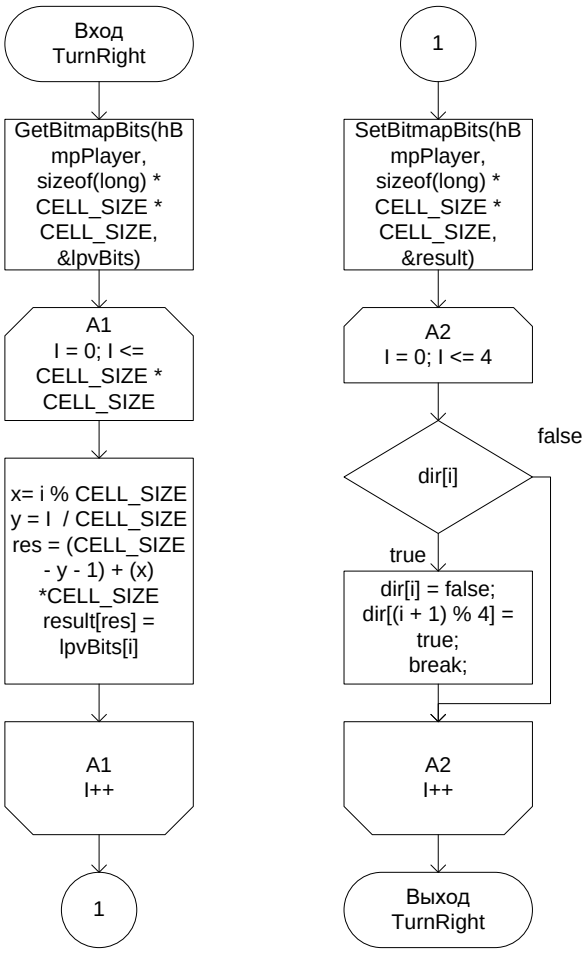

Рисунок 3.2 – Схема алгоритма метода TurnRight

3.3 Схема алгоритма метода Shot

При вызове метода Shot происходит инициализация “снаряда” и его занесение в массив таких снарядов для последующих действий (отрисовка, проверка на столкновения)

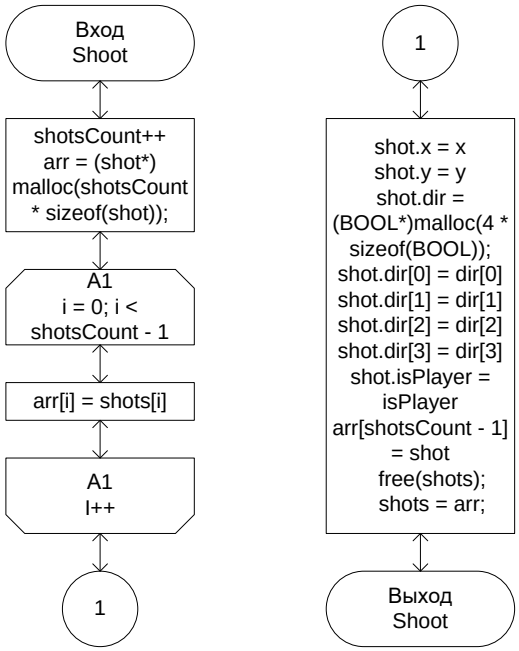

Рисунок 3.3 – Схема алгоритма метода Shot

3.4 Алгоритм действий компьютера

В каждый момент времени противник в первую очередь проверяет, может ли он стрелять (задержка = 0), и если может, то проверяет, не находится ли на одной линии с ним игрок. В случае, если проверка успешна, компьютер проверяет, есть ли препятствия на пути до противника. И уже в случае успеха всех проверок, проверяет, направлен ли он на игрока. Если противник смотрит в другую сторону, то поворачивается, получая задержку на это действие, иначе стреляет.

И в тот же момент времени, но после возможной стрельбы, бот проверяет задержки на поворот и передвижение. Если они отсутствуют, то принимается случайное решение ехать вперёд или повернуть в случайную сторону. К слову, если впереди препятствие, то бот в любом случае повернёт, а не будет пытаться ехать в стену.

3.5 Графический интерфейс

Приложение при запуске сразу начинает игру. Поскольку игровая механика строится на фиксированном количестве ячеек, а окно имеет фиксированный размер, то фон включает в себя сетку, по которой двигается

игрок. На фоне так же изображены препятствия и ловушки, имеющие логическое отображение в игре. В качестве интерфейса имеется полоса в углу экрана, отображающая здоровье игроков.

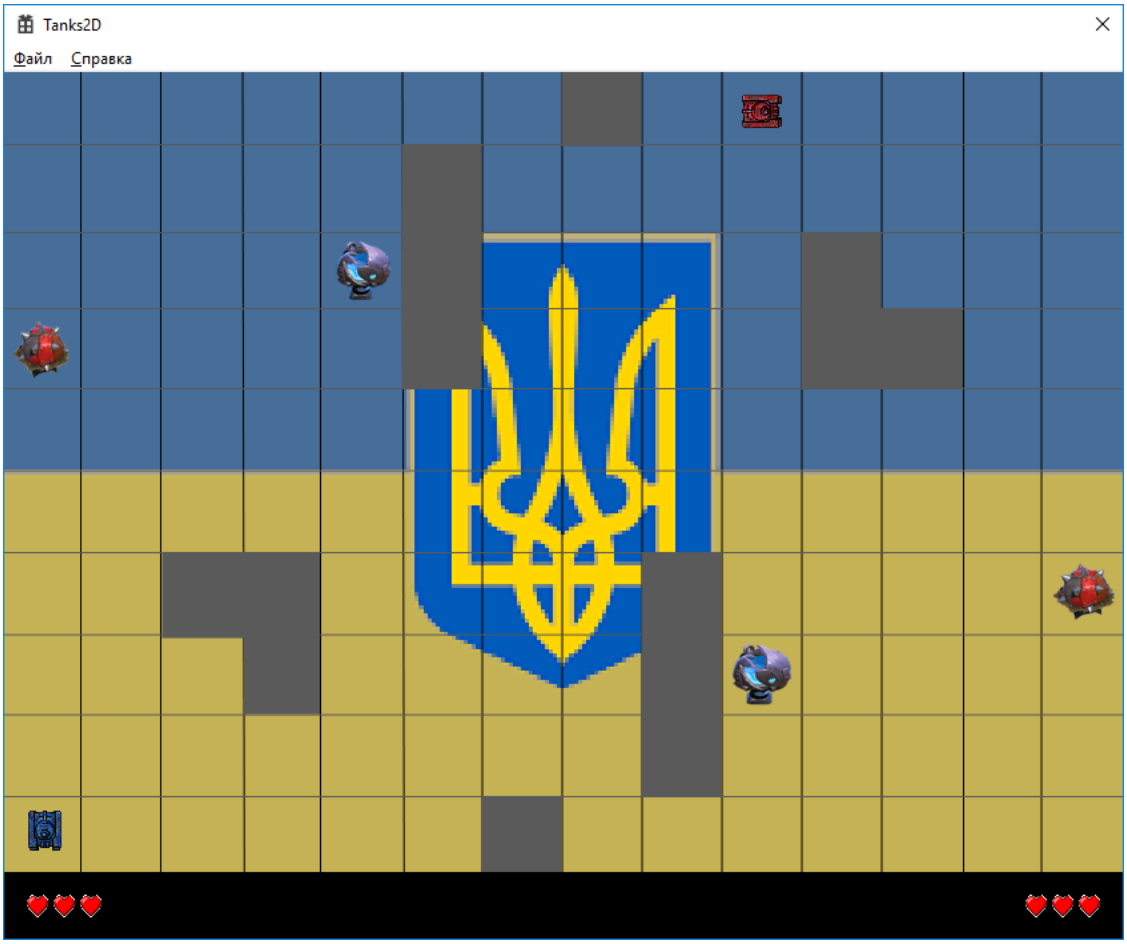

Рисунок 3.4 – Графический интерфейс приложения

4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ ПРОГРАММНОГО СРЕДСТВА

4.1 Системные требования

Для нормальной работы программного средства необходимы следующие минимальные системные требования:

– Операционная система: Windows XP, Windows 7, Windows 10;

– Процессор: Intel Pentium Silver N5000 с тактовой частотой 1.1Ггц или эквивалентный;

– Оперативная память 256 МБ (само приложение использует 3,2 МБ);

– Свободное место на жестком диске: 4,17 МБ.

4.2 Установка

Для запуска программы необходима предварительная установка.

Шаг 1. Открыть ярлык установочного файла, находящийся на диске программы. Этот ярлык изображен на рисунке 4.1.

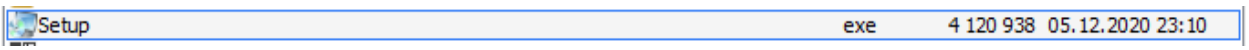

Рисунок 4.1 – Setup.exe

Шаг 2. Нажать кнопку Next на открывшемся окне (см. рисунок 4.2).

Рисунок 4.2 – Приветственное окно

Шаг 3. Изучить информацию окна «Выбор папки для установки программы», изображенную на рисунке 4.3. Следовать дальнейшим инструкциям. После этого нажать на кнопку «Далее».

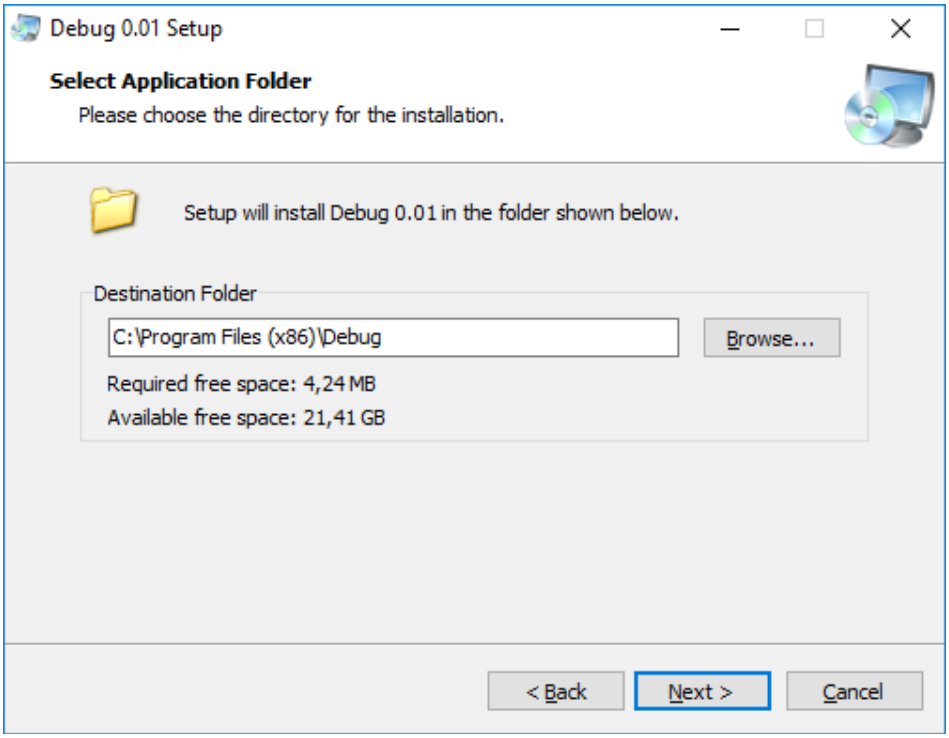

Рисунок 4.3 – Окно «Select Application Folder»

Шаг 4. Изучить информацию окна «Additional Tasks», изображенную на рисунке 4.4. Следовать дальнейшим инструкциям. После этого нажать на кнопку «Next».

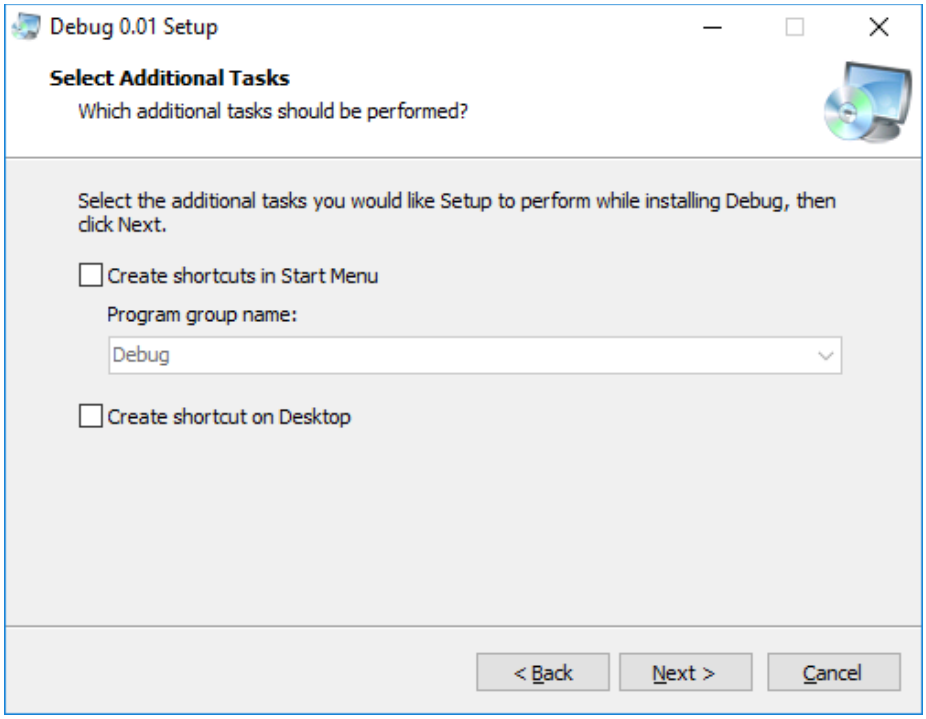

Рисунок 4.4 – Окно «Select Additional Tasks»

Шаг 5. Изучить информацию окна «Ready to Install», изображенную на рисунке 4.5. После этого нажать на кнопку «Install».

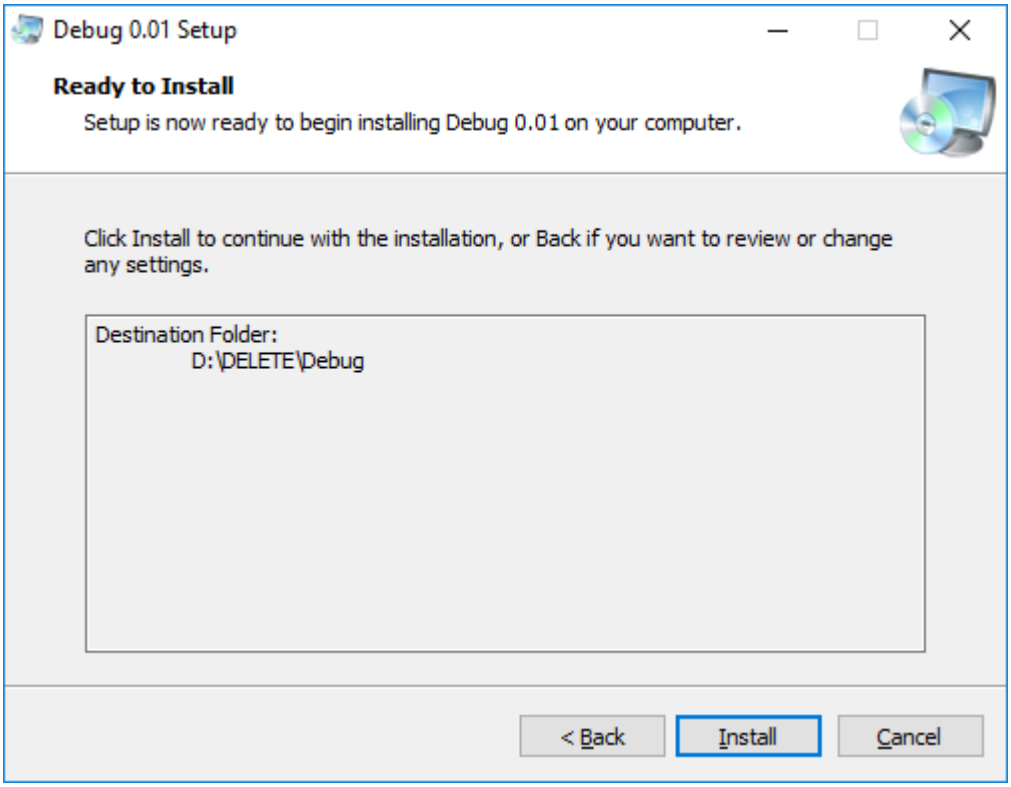

Рисунок 4.5 – Окно «Ready to Install»

Шаг 6. Изучить информацию окна «Installation Completed», изображенную на рисунке 4.6. После этого нажать на кнопку «Finish».

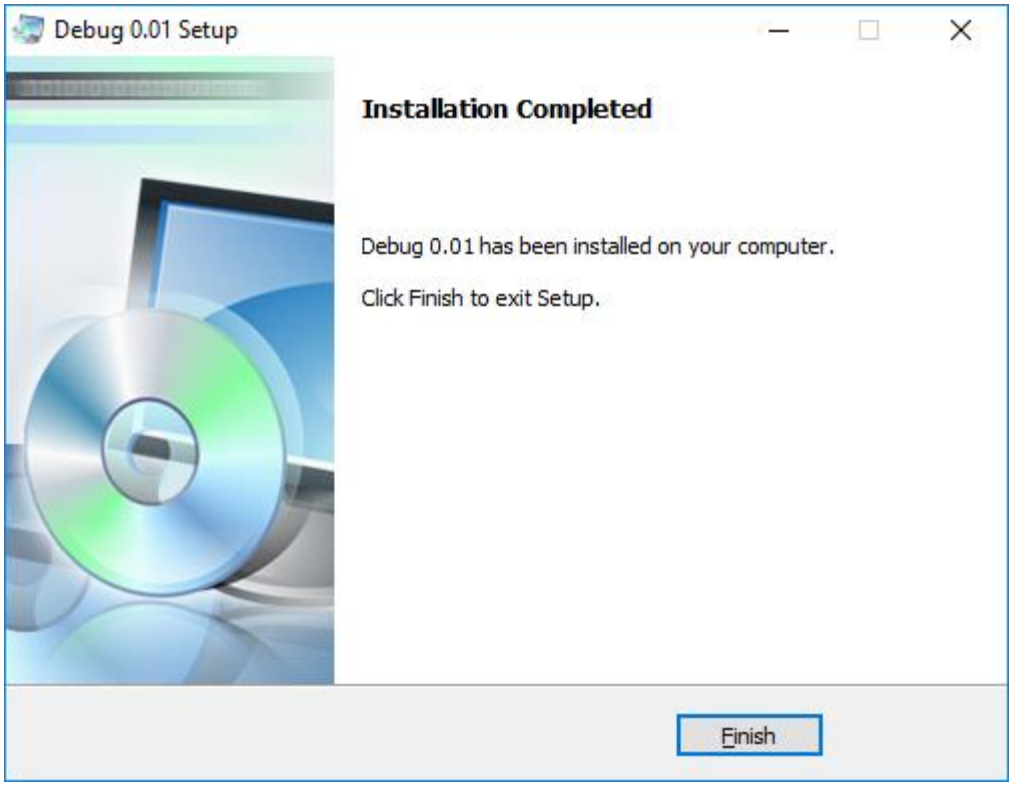

Рисунок 4.6 – Окно «Installation Completed»

4.3 Работа с программным средством

После запуска приложения появится главная игровая форма в начальном состоянии, изображенная на рисунке 4.7.

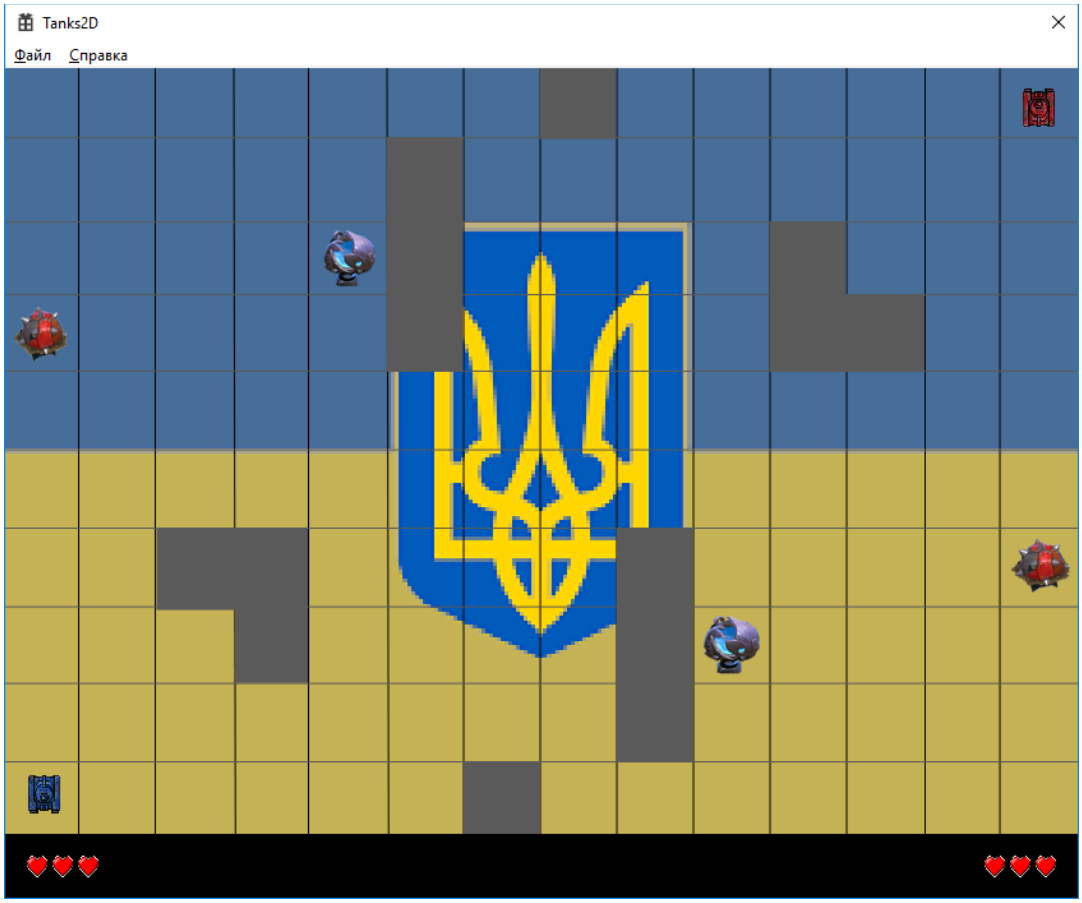

Рисунок 4.7 – Изначальный вид формы игры

В этот момент игроку, а вместе с тем и боту, предоставляется полная свобода действий. В ходе игры можно пробовать новые стратегии для победы над соперником. Основным способом является прямая перестрелка с попытками увернуться от снарядов врага (рис. 4.8)

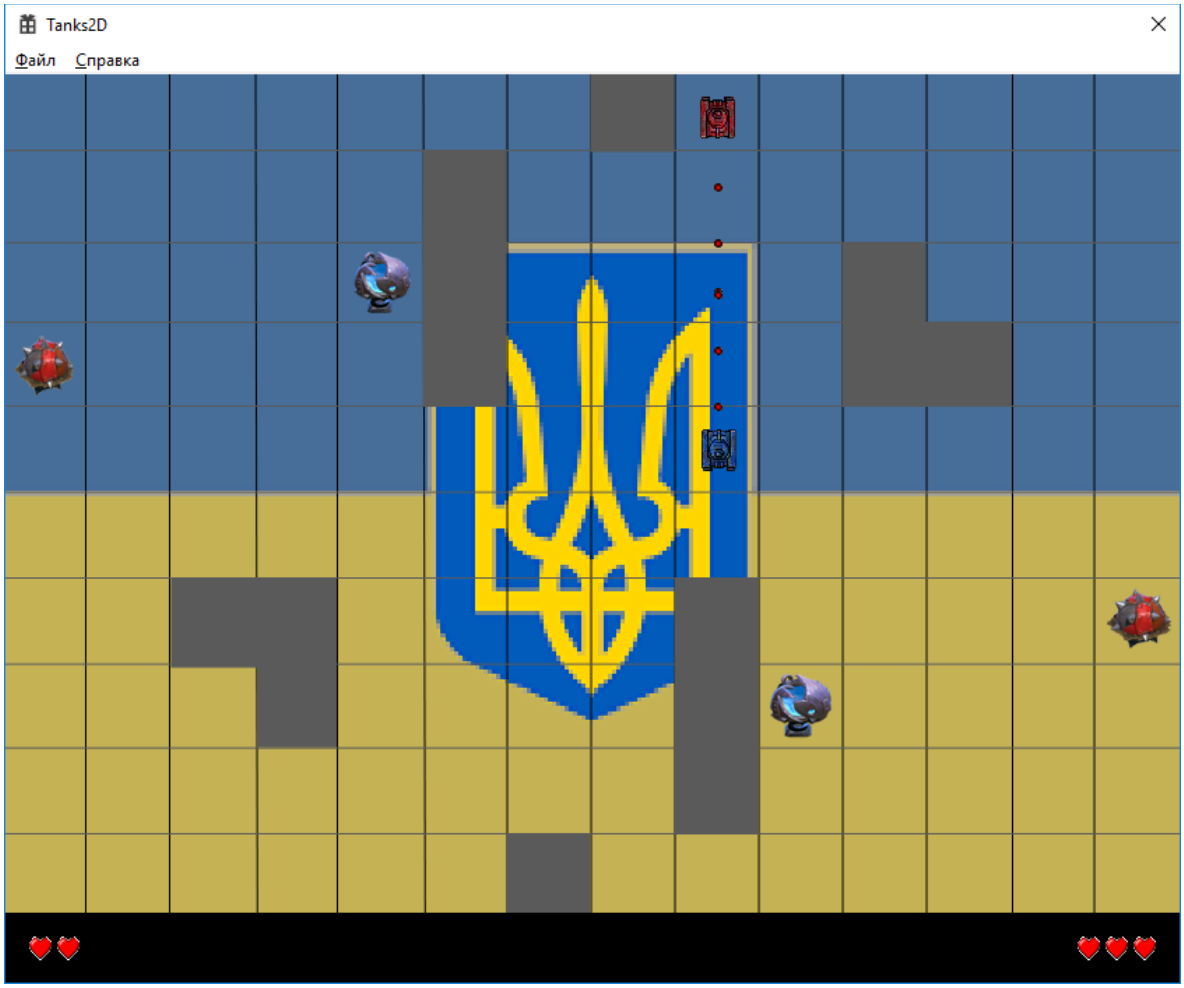

Рисунок 4.8 – Перестрелка игроков

Как можно заметить из рисунка выше, наш персонаж потерял одну единицу жизни к данному моменту от попадания противника. Однако это не единственный способ потерять здоровье. Так же на карте расположены красные мины, отнимающие здоровье у того, кто на них наехал и синие мины, блокирующие передвижение на три секунды (рис. 4.9). Это может быть как обычная случайность, так и единственное место для отступления.

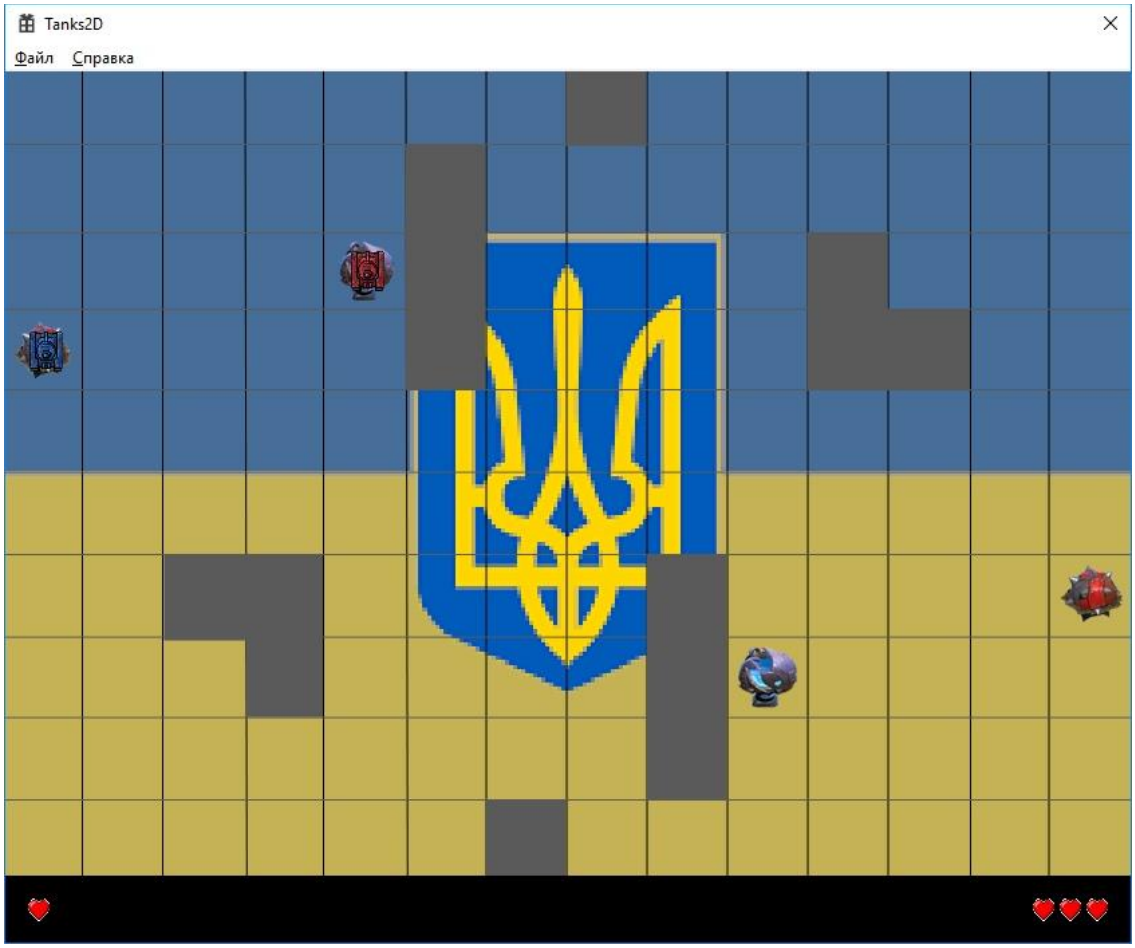

Рисунок 4.9 – Попадание в ловушку

Так же на карте можно увидеть серые препятствия. Они являются неразрушаемыми, а так же не пропускают снаряды и игроков. Комбинируя эти факты, можно придумывать различные стратегии и продолжать играть всё снова и снова, однако каждый раз по разному.

ЗАКЛЮЧЕНИЕ

В сравнении с другими подобными программами, данная имеет весь базовый функционал. Имеется.

В рамках курсовой работы был изучен набор функций WinAPI. Разработка велась на языке программирования C++ с использованием среды разработки VisualStudio 2019.

Для достижения данной цели были решены следующие задачи:

– осуществлена постановка игровой задачи;

– произведен анализ аналогов и программных средств разработки компьютерной игры;

– повторены и углублены знания в языке программирования C++;

– спроектировано приложение;

– составлены руководство пользователя и документация.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

[1] Microsoft Documentation [Электронный ресурс] Режим доступа: https://docs.microsoft.com/en-us/windows/win32/ – Дата доступа 28.10.2020.

[2] Основы программирования для Win32 API [Электронный ресурс] Режим доступа: https://dims.karelia.ru/win32/ – Дата доступа 20.11.2020.

[3] Мартин Р. Чистый код: создание, анализ и рефакторинг. Библиотека программиста – СПб.: Питер, 2013. – 464 с.: ил. – (Серия «Библиотека программиста»). ISBN 978-5–496-00487-9.

[4] MSDN – Windows API по-русски[Электронный ресурс] Режим доступа: http://narovol.narod.ru/_tbkp/New_MSDN_API/index_msdn.htm – Дата доступа 29.10.2020.

[5] Win 32 API по шагам [Электронный ресурс] Режим доступа: https://www.firststeps.ru/ – Дата доступа 02.11.2020.

[6] Справочник по функциям Windows API [Электронный ресурс] Режим доступа: http://rusproject.narod.ru/winapi/winapi.htm – Дата доступа 02.11.2020.

ПРИЛОЖЕНИЕ А
(Обязательное)
Исходный код программы

```
// Tanks2D.cpp : Определяет точку входа для приложения.
//

#include "framework.h"
#include "Tanks2D.h"
#include <windows.h>
#include <objidl.h>
#include "gdiplus.h"
#include <string>
#include <sstream>
#include <malloc.h>
#pragma comment(lib, "Msimg32.lib")

#define HEADER_HEIGHT 59
#define HEADER_WIDTH 16
#define MY_HEADER_HEIGHT 50
#define MAX_LOADSTRING 100
#define WINDOW_WIDTH 840 + HEADER_WIDTH
#define WINDOW_HEIGHT 600 + HEADER_HEIGHT
#define CELL_SIZE 60
#define MAP_HEIGHT 10
#define MAP_WIDTH 14
#define SHOT_SIZE 6
#define TO_REAL_SIZE 14

struct shot {
    int x;
    int y;
    BOOL* dir;
    BOOL isPlayer;
};

// Глобальные переменные:
HINSTANCE hInst;                                // текущий экземпляр
WCHAR szTitle[MAX_LOADSTRING];                  // Текст строки заголовка
WCHAR szWindowClass[MAX_LOADSTRING];            // имя класса главного окна
HBITMAP hBmpPlayer, hBmpEnemy, hBmpShoot, hBmpBack, hBmpBlack, hBmpHp;
```

```
RECT cRect;
RECT cellRect;
PAINTSTRUCT ps;
HDC hdc;
HDC hCompatibleDC;
HANDLE hOldBitmap;
POINT posPlayer, posEnemy;
BOOL dir[4] = { true, false, false, false };
BOOL dirEnemy[4] = { true, false, false, false };
int map[MAP_HEIGHT][MAP_WIDTH] =
{
    { 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0 },
    { 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0 },
    { 0, 0, 0, 0, 3, 1, 0, 0, 0, 0, 1, 0, 0, 0 },
    { 2, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0 },
    { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
    { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
    { 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2 },
    { 0, 0, 0, 1, 0, 0, 0, 0, 1, 3, 0, 0, 0, 0 },
    { 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0 },
    { 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0 }
};
shot* shots;
int shotsCount = 0;
int shotDelay = 0, moveDelay = 0, turnDelay = 0,
botShotDelay = 0, botMoveDelay = 0, botTurnDelay = 0,
botSpeed = 0;
int hpPlayer = 3, hpEnemy = 3;

// Отправить объявления функций, включенных в этот
модуль кода:
ATOM                MyRegisterClass(HINSTANCE
hInstance);
BOOL                InitInstance(HINSTANCE, int);
LRESULT CALLBACK    WndProc(HWND, UINT, WPARAM,
LPARAM);
INT_PTR CALLBACK    About(HWND, UINT, WPARAM, LPARAM);
VOID                TurnRight();
VOID                TurnLeft();
VOID                TurnRightEnemy();
VOID                TurnLeftEnemy();
VOID                MoveSprite(BOOL, int*, int*,
POINT*, int*, int*, BOOL*);
VOID                Shoot(int, int, BOOL*, BOOL);
VOID                Unshoot(int);
```

```
BOOL                IsStuck(int, int, int);
BOOL                IsEnemy(int, int, int, BOOL);
VOID                ShootPlayerNearby();
VOID                MoveEnemy();
VOID                    DrawBmp(HBITMAP, int, int, int,
int, int, int, int, int);

int APIENTRY wWinMain(_In_ HINSTANCE hInstance,
                      _In_opt_ HINSTANCE hPrevInstance,
                      _In_ LPWSTR    lpCmdLine,
                      _In_ int       nCmdShow)
{
    UNREFERENCED_PARAMETER(hPrevInstance);
    UNREFERENCED_PARAMETER(lpCmdLine);

    // TODO: Разместите код здесь.
    hBmpPlayer       =       (HBITMAP)LoadImageW(hInst,
L"tankBlue.bmp", IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE);
    if (!hBmpPlayer) {
        MessageBox(NULL, L"ÛÀÛ!", L"ÛÀÛ!", MB_OK);
        return 0;
    }

    hBmpEnemy        =       (HBITMAP)LoadImageW(hInst,
L"tankRed.bmp", IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE);
    if (!hBmpEnemy) {
        MessageBox(NULL, L"ÛÀÛ!", L"ÛÀÛ!", MB_OK);
        return 0;
    }

    hBmpShoot = (HBITMAP)LoadImageW(hInst, L"shot.bmp",
IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE);
    if (!hBmpShoot) {
        MessageBox(NULL, L"ÛÀÛ!", L"ÛÀÛ!", MB_OK);
        return 0;
    }

    hBmpBack         =       (HBITMAP)LoadImageW(hInst,
L"background.bmp",       IMAGE_BITMAP,       0,       0,
LR_LOADFROMFILE);
    if (!hBmpBack) {
        MessageBox(NULL, L"ÛÀÛ!", L"ÛÀÛ!", MB_OK);
        return 0;
    }
```

```
    hBmpBlack       =       (HBITMAP)LoadImageW(hInst,
L"BLACK.bmp", IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE);
    if (!hBmpBack) {
        MessageBox(NULL, L"ÛÀÛ!", L"ÛÀÛ!", MB_OK);
        return 0;
    }

    hBmpHp   =   (HBITMAP)LoadImageW(hInst,   L"hp.bmp",
IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE);
    if (!hBmpBack) {
        MessageBox(NULL, L"ÛÀÛ!", L"ÛÀÛ!", MB_OK);
        return 0;
    }

    posPlayer.x = 0;
    posPlayer.y = 9;
    posEnemy.x = 12;
    posEnemy.y = 1;

    // Инициализация глобальных строк
    LoadStringW(hInstance,    IDS_APP_TITLE,    szTitle,
MAX_LOADSTRING);
    LoadStringW(hInstance,  IDC_TANKS2D,  szWindowClass,
MAX_LOADSTRING);
    MyRegisterClass(hInstance);

    // Выполнить инициализацию приложения:
    if (!InitInstance (hInstance, nCmdShow))
    {
        return FALSE;
    }

    HACCEL  hAccelTable  =  LoadAccelerators(hInstance,
MAKEINTRESOURCE(IDC_TANKS2D));

    MSG msg;

    // Цикл основного сообщения:
    while (GetMessage(&msg, nullptr, 0, 0))
    {
        if              (!TranslateAccelerator(msg.hwnd,
hAccelTable, &msg))
        {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
```

```
        }
    }

    return (int) msg.wParam;
}



//
//  ФУНКЦИЯ: MyRegisterClass()
//
//  ЦЕЛЬ: Регистрирует класс окна.
//
ATOM MyRegisterClass(HINSTANCE hInstance)
{
    WNDCLASSEXW wcex;

    wcex.cbSize = sizeof(WNDCLASSEX);

    wcex.style          = CS_HREDRAW | CS_VREDRAW;
    wcex.lpfnWndProc    = WndProc;
    wcex.cbClsExtra     = 0;
    wcex.cbWndExtra     = 0;
    wcex.hInstance      = hInstance;
    wcex.hIcon                  =  LoadIcon(hInstance,
MAKEINTRESOURCE(IDI_TANKS2D));
    wcex.hCursor                =  LoadCursor(nullptr,
IDC_ARROW);
    wcex.hbrBackground                                  =
CreatePatternBrush((HBITMAP)LoadImage(NULL,
TEXT("background.bmp"),  0,  0,  0,  LR_LOADFROMFILE  |
LR_CREATEDIBSECTION));
    wcex.lpszMenuName                                   =
MAKEINTRESOURCEW(IDC_TANKS2D);
    wcex.lpszClassName  = szWindowClass;
    wcex.hIconSm            =  LoadIcon(wcex.hInstance,
MAKEINTRESOURCE(IDI_SMALL));

    return RegisterClassExW(&wcex);
}

//
//   ФУНКЦИЯ: InitInstance(HINSTANCE, int)
//
//      ЦЕЛЬ:  Сохраняет  маркер  экземпляра  и  создает
```

```
главное окно
//
//   КОММЕНТАРИИ:
//
//         В этой функции маркер экземпляра сохраняется
в глобальной переменной, а также
//        создается и выводится главное окно программы.
//
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
   hInst = hInstance; // Сохранить маркер экземпляра в
глобальной переменной

   HWND hWnd = CreateWindowW(szWindowClass, szTitle,
WS_OVERLAPPED | WS_BORDER | WS_SYSMENU,
               0,  0,  WINDOW_WIDTH,  WINDOW_HEIGHT  +
MY_HEADER_HEIGHT,    nullptr,    nullptr,    hInstance,
nullptr);

   if (!hWnd)
   {
      return FALSE;
   }
   SetTimer(hWnd, 1, 25, NULL);

   ShowWindow(hWnd, nCmdShow);
   UpdateWindow(hWnd);

   return TRUE;
}

//
//  ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)
//
//  ЦЕЛЬ: Обрабатывает сообщения в главном окне.
//
//  WM_COMMAND  - обработать меню приложения
//  WM_PAINT    - Отрисовка главного окна
//    WM_DESTROY    - отправить сообщение о выходе и
вернуться
//
//
int baseX = 0, baseY = WINDOW_HEIGHT - 59 - CELL_SIZE;
int   enemyX   =   WINDOW_WIDTH   -   2   *   CELL_SIZE   -
HEADER_WIDTH, enemyY = CELL_SIZE;
```

```
int const moveSpeed = CELL_SIZE;

LRESULT CALLBACK WndProc(HWND hWnd, UINT message,
WPARAM wParam, LPARAM lParam)
{
    switch (message)
    {
    case WM_COMMAND:
        {
            int wmId = LOWORD(wParam);
            // Разобрать выбор в меню:
            switch (wmId)
            {
            case IDM_ABOUT:
                DialogBox(hInst,
MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
                break;
            case IDM_EXIT:
                DestroyWindow(hWnd);
                break;
            default:
                return   DefWindowProc(hWnd,   message,
wParam, lParam);
            }
        }
        break;
    case WM_PAINT:
        {
        hdc = BeginPaint(hWnd, &ps);
        hCompatibleDC = CreateCompatibleDC(hdc);
        int newX, newY;

        for (int i = 0; i < shotsCount; i++)
        {
            newX  =  shots[i].x  +  CELL_SIZE  /  2  -
SHOT_SIZE / 2;
            newY  =  shots[i].y  +  CELL_SIZE  /  2  -
SHOT_SIZE / 2, SHOT_SIZE, SHOT_SIZE;
            DrawBmp(hBmpShoot,  newX,  newY,  SHOT_SIZE,
SHOT_SIZE, 0, 0, SHOT_SIZE, SHOT_SIZE);
            if (shots[i].dir[0] && newY > 0)
                shots[i].y--;
            if (shots[i].dir[1] && newX < WINDOW_WIDTH
- SHOT_SIZE)
                shots[i].x++;
```

```
            if (shots[i].dir[2] && newY < WINDOW_HEIGHT
- SHOT_SIZE)
                shots[i].y++;
            if (shots[i].dir[3] && newX > 0)
                shots[i].x--;

            if    (IsEnemy(newX,    newY,    SHOT_SIZE,
shots[i].isPlayer))
            {
                Unshoot(i);
                if (shots[i].isPlayer)
                    hpEnemy--;
                else
                    hpPlayer--;

                if (hpEnemy == 0 || hpPlayer == 0)
                {
                    PostMessage(FindWindow(NULL,
L"Tanks2D"), WM_QUIT, 0, 0);
                }
            }
            if (IsStuck(newX, newY, SHOT_SIZE))
                Unshoot(i);
        }

        DrawBmp(hBmpPlayer, baseX + TO_REAL_SIZE, baseY
+ TO_REAL_SIZE, CELL_SIZE - TO_REAL_SIZE, CELL_SIZE -
TO_REAL_SIZE, TO_REAL_SIZE, TO_REAL_SIZE, CELL_SIZE -
TO_REAL_SIZE, CELL_SIZE - TO_REAL_SIZE);

        DrawBmp(hBmpEnemy,   enemyX   +   TO_REAL_SIZE,
enemyY   +   TO_REAL_SIZE,   CELL_SIZE   -   TO_REAL_SIZE,
CELL_SIZE  -  TO_REAL_SIZE,  TO_REAL_SIZE,  TO_REAL_SIZE,
CELL_SIZE - TO_REAL_SIZE, CELL_SIZE - TO_REAL_SIZE);

        hOldBitmap     =     SelectObject(hCompatibleDC,
hBmpBlack);
        BitBlt(hdc,   0,   600,   WINDOW_WIDTH,   600   +
MY_HEADER_HEIGHT, hCompatibleDC, 0, 0, SRCCOPY);
        SelectObject(hCompatibleDC, hOldBitmap);

        for (int i = 0; i < hpPlayer; i++)
        {
            DrawBmp(hBmpHp, 16 + i * 20, 600 + 16, 18,
18, 0, 0, 18, 18);
```

```
        }

        for (int i = 0; i < hpEnemy; i++)
        {
            DrawBmp(hBmpHp, 824 - 18 - i * 20, 600 +
16, 18, 18, 0, 0, 18, 18);
        }

        EndPaint(hWnd, &ps);
        DeleteDC(hdc);
        DeleteDC(hCompatibleDC);
        }
        break;
    case WM_DESTROY:
        PostQuitMessage(0);
        break;
    case WM_KEYDOWN: {
        switch (wParam) {
        case VK_UP: {
            if (moveDelay == 0)
            {
                moveDelay = 10;
                MoveSprite(true,     &baseX,     &baseY,
&posPlayer, &hpPlayer, &moveDelay, dir);
            }
            break;
        }
        case VK_DOWN: {
            if (moveDelay == 0)
            {
                moveDelay = 10;
                MoveSprite(false,    &baseX,     &baseY,
&posPlayer, &hpPlayer, &moveDelay, dir);
            }
            break;
        }
        case VK_LEFT: {
            if (turnDelay == 0)
            {
                turnDelay = 10;
                TurnLeft();
            }
            break;
        }
        case VK_RIGHT: {
```

```
            if (turnDelay == 0)
            {
                turnDelay = 10;
                TurnRight();
            }
            break;
        }
        case VK_SPACE: {
            if (shotDelay == 0)
            {
                shotDelay = 40;
                Shoot(baseX, baseY, dir, true);
            }
        }
        }
        break;
    }
    case WM_GETMINMAXINFO: {
        LPMINMAXINFO lpMMI = (LPMINMAXINFO)lParam;
        lpMMI->ptMinTrackSize.y = WINDOW_HEIGHT;
        lpMMI->ptMinTrackSize.y = WINDOW_HEIGHT;
        lpMMI->ptMinTrackSize.x = WINDOW_WIDTH;
        lpMMI->ptMinTrackSize.x = WINDOW_WIDTH;
        break;
    }
    case WM_TIMER: {
        InvalidateRect(hWnd, NULL, TRUE);
        if (shotDelay != 0)
            shotDelay--;

        if (moveDelay != 0)
            moveDelay--;

        if (turnDelay != 0)
            turnDelay--;

        if (botTurnDelay != 0)
            botTurnDelay--;

        if (botShotDelay != 0)
            botShotDelay--;
        else
            ShootPlayerNearby();

        if (botMoveDelay != 0)
```

```
            botMoveDelay--;
        else
            if (botTurnDelay == 0)
                MoveEnemy();

        break;
    }
    //case WM_ERASEBKGND:
    //    return (LRESULT)1; // Say we handled it.
    default:
        return  DefWindowProc(hWnd,  message,  wParam,
lParam);
    }
    return 0;
}

// Обработчик сообщений для окна "О программе".
INT_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM
wParam, LPARAM lParam)
{
    UNREFERENCED_PARAMETER(lParam);
    switch (message)
    {
    case WM_INITDIALOG:
        return (INT_PTR)TRUE;

    case WM_COMMAND:
        if (LOWORD(wParam) == IDOK || LOWORD(wParam) ==
IDCANCEL)
        {
            EndDialog(hDlg, LOWORD(wParam));
            return (INT_PTR)TRUE;
        }
        break;
    }
    return (INT_PTR)FALSE;
}

VOID TurnRight()
{
    long      lpvBits[CELL_SIZE      *      CELL_SIZE],
result[CELL_SIZE * CELL_SIZE];
    int x, y, res;
    GetBitmapBits(hBmpPlayer,  sizeof(long)  *  CELL_SIZE
* CELL_SIZE, &lpvBits);
```

```
    for (int i = 0; i < CELL_SIZE * CELL_SIZE; i++)
    {
        x = i % CELL_SIZE;
        y = i / CELL_SIZE;
        res = (CELL_SIZE - y - 1) + (x) *CELL_SIZE;
        result[res] = lpvBits[i];
    }
    SetBitmapBits(hBmpPlayer, sizeof(long) * CELL_SIZE
* CELL_SIZE, &result);

    for (int i = 0; i < 4; i++)
    {
        if (dir[i])
        {
            dir[i] = false;
            dir[(i + 1) % 4] = true;
            break;
        }
    }
}

VOID TurnLeft()
{
    long      lpvBits[CELL_SIZE      *      CELL_SIZE],
result[CELL_SIZE * CELL_SIZE];
    int x, y, res;
    GetBitmapBits(hBmpPlayer, sizeof(long) * CELL_SIZE
* CELL_SIZE, &lpvBits);
    for (int i = 0; i < CELL_SIZE * CELL_SIZE; i++)
    {
        x = i % CELL_SIZE;
        y = i / CELL_SIZE;
        res = (CELL_SIZE - y - 1) + (x) * CELL_SIZE;
        result[i] = lpvBits[res];
    }
    SetBitmapBits(hBmpPlayer, sizeof(long) * CELL_SIZE
* CELL_SIZE, &result);

    for (int i = 0; i < 4; i++)
    {
        if (dir[i])
        {
            dir[i] = false;
            dir[(i + 3) % 4] = true;
            break;
```

```
        }
    }
}

VOID TurnRightEnemy()
{
    long      lpvBits[CELL_SIZE      *      CELL_SIZE],
result[CELL_SIZE * CELL_SIZE];
    int x, y, res;
    GetBitmapBits(hBmpEnemy, sizeof(long) * CELL_SIZE *
CELL_SIZE, &lpvBits);
    for (int i = 0; i < CELL_SIZE * CELL_SIZE; i++)
    {
        x = i % CELL_SIZE;
        y = i / CELL_SIZE;
        res = (CELL_SIZE - y - 1) + (x)*CELL_SIZE;
        result[res] = lpvBits[i];
    }
    SetBitmapBits(hBmpEnemy, sizeof(long) * CELL_SIZE *
CELL_SIZE, &result);

    for (int i = 0; i < 4; i++)
    {
        if (dirEnemy[i])
        {
            dirEnemy[i] = false;
            dirEnemy[(i + 1) % 4] = true;
            break;
        }
    }
}

VOID TurnLeftEnemy()
{
    long      lpvBits[CELL_SIZE      *      CELL_SIZE],
result[CELL_SIZE * CELL_SIZE];
    int x, y, res;
    GetBitmapBits(hBmpEnemy, sizeof(long) * CELL_SIZE *
CELL_SIZE, &lpvBits);
    for (int i = 0; i < CELL_SIZE * CELL_SIZE; i++)
    {
        x = i % CELL_SIZE;
        y = i / CELL_SIZE;
        res = (CELL_SIZE - y - 1) + (x)*CELL_SIZE;
        result[i] = lpvBits[res];
```

```
    }
    SetBitmapBits(hBmpEnemy, sizeof(long) * CELL_SIZE *
CELL_SIZE, &result);

    for (int i = 0; i < 4; i++)
    {
        if (dirEnemy[i])
        {
            dirEnemy[i] = false;
            dirEnemy[(i + 3) % 4] = true;
            break;
        }
    }
}

VOID MoveSprite(BOOL flag, int *adrX, int* adrY, POINT
*posPlayer, int* hp, int* delay, BOOL* dir)
{
    if (flag)
    {
        if (dir[0] && *adrY - CELL_SIZE >= 0)
            if (map[*adrY / CELL_SIZE - 1][*adrX /
CELL_SIZE] != 1 && (*adrY / CELL_SIZE - 1 != posEnemy.y
|| *adrX / CELL_SIZE != posEnemy.x))
            {
                *adrY -= CELL_SIZE;
                posPlayer->y--;
            }
        if (dir[1] && *adrX + CELL_SIZE <= WINDOW_WIDTH
- CELL_SIZE)
            if   (map[*adrY   /   CELL_SIZE][*adrX   /
CELL_SIZE + 1] != 1 && (*adrY / CELL_SIZE != posEnemy.y
|| *adrX / CELL_SIZE + 1 != posEnemy.x))
            {
                *adrX += CELL_SIZE;
                posPlayer->x++;
            }
        if   (dir[2]   &&   *adrY   +   CELL_SIZE   <=
WINDOW_HEIGHT - CELL_SIZE)
            if (map[*adrY / CELL_SIZE + 1][*adrX /
CELL_SIZE] != 1 && (*adrY / CELL_SIZE + 1 != posEnemy.y
|| *adrX / CELL_SIZE != posEnemy.x))
            {
                *adrY += CELL_SIZE;
                posPlayer->y++;
```

```
            }
        if (dir[3] && *adrX - CELL_SIZE >= 0)
            if   (map[*adrY   /   CELL_SIZE][*adrX   /
CELL_SIZE - 1] != 1 && (*adrY / CELL_SIZE != posEnemy.y
|| *adrX / CELL_SIZE - 1 != posEnemy.x))
            {
                *adrX -= CELL_SIZE;
                posPlayer->x--;
            }
    }
    else
    {
        if   (dir[0]   &&   *adrY   +   CELL_SIZE   <=
WINDOW_HEIGHT - CELL_SIZE)
            if  (map[*adrY  /  CELL_SIZE  +  1][*adrX  /
CELL_SIZE] != 1 && (*adrY / CELL_SIZE + 1 != posEnemy.y
|| *adrX / CELL_SIZE != posEnemy.x))
            {
                *adrY += CELL_SIZE;
                posPlayer->y++;
            }
        if (dir[1] && *adrX - CELL_SIZE >= 0)
            if   (map[*adrY   /   CELL_SIZE][*adrX   /
CELL_SIZE - 1] != 1 && (*adrY / CELL_SIZE != posEnemy.y
|| *adrX / CELL_SIZE - 1 != posEnemy.x))
            {
                *adrX -= CELL_SIZE;
                posPlayer->x--;
            }
        if (dir[2] && *adrY - CELL_SIZE >= 0)
            if  (map[*adrY  /  CELL_SIZE  -  1][*adrX  /
CELL_SIZE] != 1 && (*adrY / CELL_SIZE - 1 != posEnemy.y
|| *adrX / CELL_SIZE != posEnemy.x))
            {
                *adrY -= CELL_SIZE;
                posPlayer->y--;
            }
        if (dir[3] && *adrX + CELL_SIZE <= WINDOW_WIDTH
- CELL_SIZE)
            if   (map[*adrY   /   CELL_SIZE][*adrX   /
CELL_SIZE + 1] != 1 && (*adrY / CELL_SIZE != posEnemy.y
|| *adrX / CELL_SIZE + 1 != posEnemy.x))
            {
                *adrX += CELL_SIZE;
                posPlayer->x++;
```

```
            }
        }
        if (map[posPlayer->y][posPlayer->x] == 2)
        {
            (*hp)--;
        }

        if (map[posPlayer->y][posPlayer->x] == 3)
        {
            *delay = 120;
        }
}

VOID Shoot(int x, int y, BOOL* dir, BOOL isPlayer)
{
    shotsCount++;
    shot*    arr    =    (shot*)malloc(shotsCount    *
sizeof(shot));
    for (int i = 0; i < shotsCount - 1; i++)
    {
        arr[i] = shots[i];
    }

    shot shot;
    shot.x = x;
    shot.y = y;
    shot.dir = (BOOL*)malloc(4 * sizeof(BOOL));
    shot.dir[0] = dir[0];
    shot.dir[1] = dir[1];
    shot.dir[2] = dir[2];
    shot.dir[3] = dir[3];
    shot.isPlayer = isPlayer;
    arr[shotsCount - 1] = shot;
    free(shots);
    shots = arr;
}

VOID Unshoot(int pos)
{
    shotsCount--;
    shot*    arr    =    (shot*)malloc(shotsCount    *
sizeof(shot));
    int position = 0;
    while (position < pos)
    {
```

```
        arr[position] = shots[position];
        position++;
    }
    while (position < shotsCount)
    {
        arr[position] = shots[position + 1];
        position++;
    }
    free(shots[pos].dir);
    free(shots);
    shots = arr;
}

BOOL IsStuck(int x, int y, int size)
{
    if (x + size > WINDOW_WIDTH || x <= 0 || y + size>
WINDOW_HEIGHT || y <= 0
        || map[y / CELL_SIZE][x / CELL_SIZE] == 1 ||
map[(y + size) / CELL_SIZE][(x + size) / CELL_SIZE] ==
1)
        return true;
    else
        return false;
}

BOOL IsEnemy(int x, int y, int size, BOOL isPlayer)
{
    if (isPlayer)
    {
        if ((x + TO_REAL_SIZE <= enemyX + CELL_SIZE &&
y + TO_REAL_SIZE <= enemyY + CELL_SIZE &&
            x - TO_REAL_SIZE >= enemyX && y -
TO_REAL_SIZE >= enemyY) ||
            ((x + size) + TO_REAL_SIZE <= enemyX +
CELL_SIZE && (y + size) + TO_REAL_SIZE <= enemyY +
CELL_SIZE &&
                (x + size) - TO_REAL_SIZE >= enemyX &&
(y + size) - TO_REAL_SIZE >= enemyY))
            return true;
        else
            return false;
    }
    else
    {
        if ((x + TO_REAL_SIZE <= baseX + CELL_SIZE && y
```

```
+ TO_REAL_SIZE <= baseY + CELL_SIZE &&
            x   -   TO_REAL_SIZE   >=   baseX   &&   y   -
TO_REAL_SIZE >= baseY) ||
            ((x  +  size)  +  TO_REAL_SIZE  <=  baseX  +
CELL_SIZE  &&  (y  +  size)  +  TO_REAL_SIZE  <=  baseY  +
CELL_SIZE &&
                (x + size) - TO_REAL_SIZE >= baseX &&
(y + size) - TO_REAL_SIZE >= baseY))
            return true;
        else
            return false;
    }
}

VOID ShootPlayerNearby()
{
    if (posPlayer.x == posEnemy.x)
    {
        if (posPlayer.y < posEnemy.y)
        {
            for (int i = posEnemy.y; i <= posPlayer.y;
i++)
            {
                if (map[i][posPlayer.x] == 1)
                    return;
            }
            if (!dirEnemy[0])
            {
                if (botTurnDelay == 0)
                {
                    if (dirEnemy[1])
                        TurnLeftEnemy();
                    else
                        TurnRightEnemy();
                    botTurnDelay = 20;
                }
            }
            else
            {
                Shoot(enemyX, enemyY, dirEnemy, false);
                botShotDelay = 20;
            }
        }
        else
        {
```

```
            for (int i = posPlayer.y; i <= posEnemy.y;
i++)
            {
                if (map[i][posPlayer.x] == 1)
                    return;
            }
            if (!dirEnemy[2])
            {
                if (botTurnDelay == 0)
                {
                    if (dirEnemy[3])
                        TurnLeftEnemy();
                    else
                        TurnRightEnemy();
                    botTurnDelay = 20;
                }
            }
            else
            {
                Shoot(enemyX, enemyY, dirEnemy, false);
                botShotDelay = 20;
            }
        }
    }
    else
    {
        if (posPlayer.y == posEnemy.y)
        {
            if (posPlayer.x > posEnemy.x)
            {
                for   (int   i   =   posEnemy.x;   i   <=
posPlayer.x; i++)
                {
                    if (map[posPlayer.x][i] == 1)
                        return;
                }
                if (!dirEnemy[1])
                {
                    if (botTurnDelay == 0)
                    {
                        if (dirEnemy[2])
                            TurnLeftEnemy();
                        else
                            TurnRightEnemy();
                        botTurnDelay = 20;
```

```
                    }
                }
                else
                {
                    Shoot(enemyX,   enemyY,   dirEnemy,
false);
                    botShotDelay = 20;
                }
            }
            else
            {
                for   (int   i   =   posPlayer.x;   i   <=
posEnemy.x; i++)
                {
                    if (map[posPlayer.x][i] == 1)
                        return;
                }
                if (!dirEnemy[3])
                {
                    if (botTurnDelay == 0)
                    {
                        if (dirEnemy[0])
                            TurnLeftEnemy();
                        else
                            TurnRightEnemy();
                        botTurnDelay = 20;
                    }
                }
                else
                {
                    Shoot(enemyX,   enemyY,   dirEnemy,
false);
                    botShotDelay = 20;
                }
            }
        }
        return;
    }
}

VOID MoveEnemy()
{
    if (rand() % 3)
    {
        if (
```

```
            !(
                (dirEnemy[0]    &&    !(posEnemy.y    -
dirEnemy[0] >= 0)) ||
                (dirEnemy[1]    &&    !(posEnemy.x    +
dirEnemy[1] <= 13)) ||
                (dirEnemy[2]    &&    !(posEnemy.y    +
dirEnemy[2] <= 9)) ||
                (dirEnemy[3]    &&    !(posEnemy.x    -
dirEnemy[3] >= 0))
             )
           )
        {
            if   (map[posEnemy.y   -   dirEnemy[0]   +
dirEnemy[2]][posEnemy.x - dirEnemy[3] + dirEnemy[1]] !=
1 &&
                posEnemy.y - dirEnemy[0] + dirEnemy[2]
!= posPlayer.y &&
                posEnemy.x - dirEnemy[3] + dirEnemy[1]
!= posPlayer.x)
            {
                botMoveDelay = 20;
                botTurnDelay = 20;
                MoveSprite(true,    &enemyX,    &enemyY,
&posEnemy, &hpEnemy, &botMoveDelay, dirEnemy);
            }
        }
        else
        {
            if (rand() % 2)
            {
                TurnLeftEnemy();
                botTurnDelay = 20;
            }
            else
            {
                TurnRightEnemy();
                botTurnDelay = 20;
            }
        }
    }
    else
    {
        if (rand() % 2)
        {
            TurnLeftEnemy();
```

```
            botTurnDelay = 20;
        }
        else
        {
            TurnRightEnemy();
            botTurnDelay = 20;
        }
    }
}

VOID DrawBmp(HBITMAP hBmp, int baseX, int baseY, int
sizeX, int sizeY, int baseX2, int baseY2, int sizeX2,
int sizeY2)
{
    hOldBitmap = SelectObject(hCompatibleDC, hBmp);
    TransparentBlt(hdc, baseX, baseY, sizeX, sizeY,
hCompatibleDC, baseX2, baseY2, sizeX2, sizeY2, RGB(255,
255, 255));
    SelectObject(hCompatibleDC, hOldBitmap);
}
```

ВЕДОМОСТЬ ДОКУМЕНТОВ

Обозначение	Наименование	Дополнительные сведения
	Текстовые документы	
БГУИР КП 1–40 01 01 623 ПЗ	Пояснительная записка	46 с.
	Графические документы	
ГУИР.851006-01 СА	Поиск целей противника	Формат А1
	Схема алгоритма	

					БГУИР КП I- 40 01 01 623 ПЗ	
Изм	Лист	№ докум.	Подп.	Дата		
Разраб.		Одиноченко М.И.			Игровое приложение «Tanks 2D» Ведомость курсового проекта	Литера: Т; Лист: 46; Листов: 46
Провер.		Жиденко А.Л.				Кафедра ПОИТ гр. 851006