­Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Конструирование программного обеспечения (КПО)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

Программное средство «Игровое приложение танки»

БГУИР КП 1-40 01 01 014 ПЗ

Студент: гр. 151004 Дубовский А.В.

Руководитель: асс. Шостак Е.В.

Минск 2022

содержание

[Введение 3](#_Toc122570595)

[1 Аналитический обзор литературы 4](#_Toc122570596)

[1.1 Обзор существующих аналогов 4](#_Toc122570597)

[1.2 Перечень функциональных требований 6](#_Toc122570598)

[1.3 Состав и параметры технических и программных средств 6](#_Toc122570599)

[2 Моделирование предметной области 7](#_Toc122570600)

[2.1 Связный список 7](#_Toc122570601)

[3 Проектирование программного средства 9](#_Toc122570602)

[3.1 Структура программы 9](#_Toc122570603)

[3.2 Проектирование интерфейса программного средства 9](#_Toc122570604)

[3.3 Проектирование функционала программного средства 9](#_Toc122570605)

[4 Разработка программного средства 13](#_Toc122570606)

[4.1 Проверка столкновения танка 13](#_Toc122570607)

[4.2 Движение танка 14](#_Toc122570608)

[4.3 Заполнение карты 15](#_Toc122570609)

[4.4 Выстрел танка 16](#_Toc122570610)

[5 Тестирование программного средства 17](#_Toc122570611)

[5.1 Запуск игры 17](#_Toc122570612)

[5.2 Конец игры 19](#_Toc122570613)

[6 Руководство пользователя 21](#_Toc122570614)

[6.1 Минимальные системные требования 21](#_Toc122570615)

[6.2 Установка 21](#_Toc122570616)

[6.3 Игровое окно 21](#_Toc122570617)

[Заключение 23](#_Toc122570618)

[Список использованной литературы 24](#_Toc122570619)

[Приложение А 25](#_Toc122570620)

# Введение

Данный курсовой проект посвящён обновлению легендарной игры «Танчики», для преобщения молодёжи к класике.

Первая версия «Танчиков» выпущина в 1980г компание «Namco».

В основе игрового процесса «Танчиков» лежит перемещение по карте с последующим отстрелом врагов.

В 1985г компания «Namco» выпустила обновлённую версию танчиков для консоли «Famicon».

В 2004г компаания «Бука» выпустила 3D версию «Танчиков».

# Аналитический обзор литературы

## Обзор существующих аналогов

### Игророве приложение «Танчики Денди 1990»

Программное средство, изображённое на рисунке 1.1, является наиболее известной итерацией игрового приложения танчики. Ввиду это являеться основным конкурентом разрабатываемого приложения.

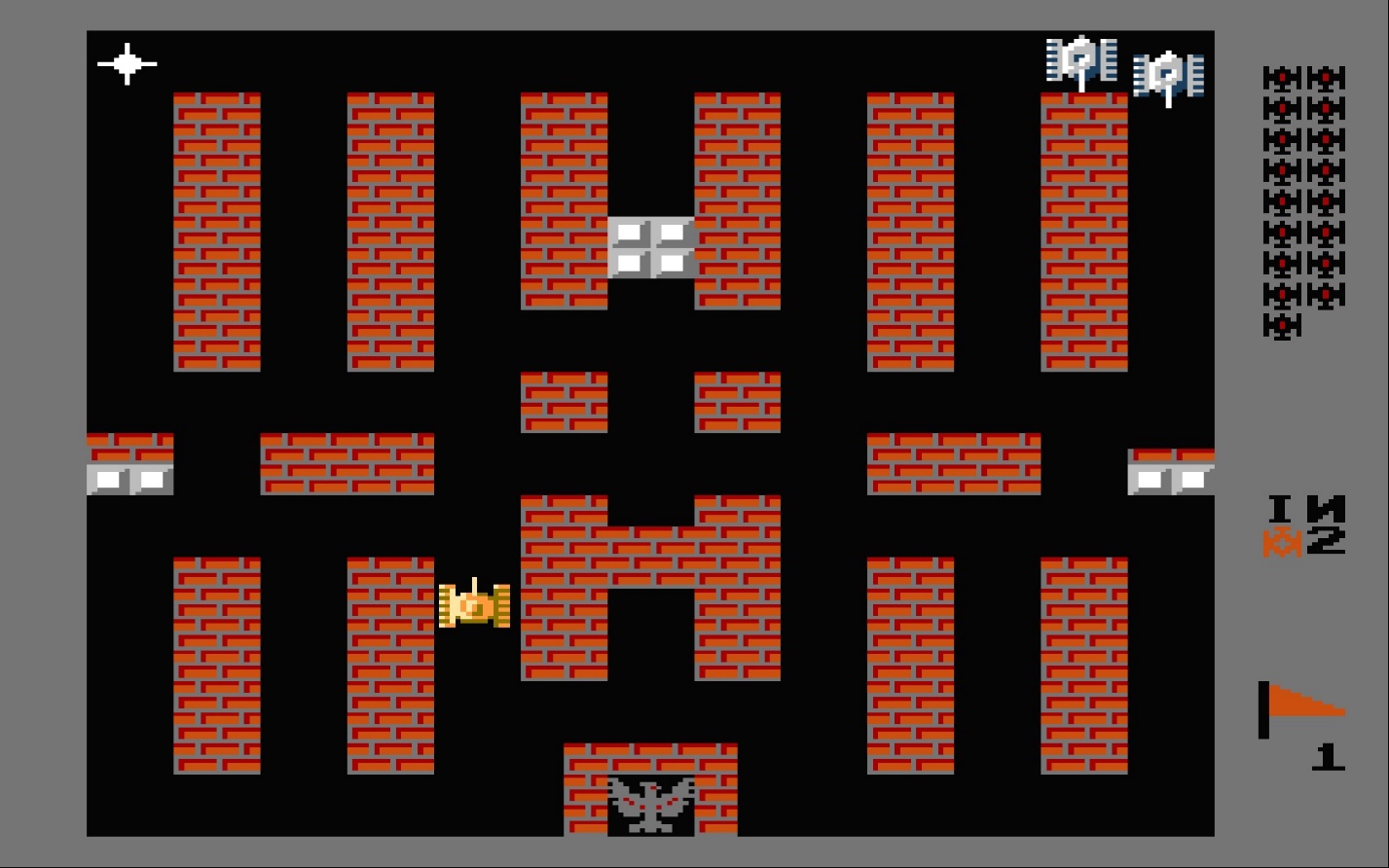


Рисунок 1.1 – Игророве приложение «Танчики Денди 1990»

Список выполняемых функций:

* управление танком;
* сражение с ботами.

Рассмотрим достоинства данного приложения:

* наличие 14 уровней;
* разнообразие врагов и препядствий;
* разруемость препядствий;

Рассмотрим недостатки данного приложения:

* запуск только на платформе Денди.

### Игровое приложение «Реалистичный симулятр танковых сражений»

Программное средство, изображённое на рисунке 1.2, сохраняет привычный гемплей танчиков, превнося в него 3D.



Рисунок 1.2 – Игровое приложение «Реалистический симулятр танковых сражений»

Список выполняемых функций:

* управление танком;
* сражение с ботами;
* Растановка танков на карте.

Рассмотрим достоинства данного приложения:

* 3D графика;
* использование браузера, что позволяет запускать игровое приложение

на любом устройстве.

Рассмотрим недостатки данного приложения:

* невозможность играть при отсутствии подключения к интернету;
* низкая производительность.

## Перечень функциональных требований

После анализа аналогов был составлен следующий список выполняемых функций:

* управление танком;
* сражение с ботами;
* перезапуск игры;
* перезарядка танка;
* отрисовка карты;
* счётчик убийств.

Для разработки программного средства будет использоваться язык программирования C и среда разработки Visual Studio 2022, а также WINAPI и .

WINAPI используется с целью создания графического интерфейса и дополнительных потоков.

## Состав и параметры технических и программных средств

Приложение должно функционировать на персональных компьютерах со следующими характеристиками:

* процессор Intel Celeron N4020 или лучше;
* оперативная память 128 MB или лучше;
* свободное место на диске в размере 3 мегабайта или больше.

Приложение должно функционировать в окружении операционной системы Windows XP или новее.

В данном разделе указаны минимальные технические требования для запуска программного средства. Для эксплуатации в реальных могут потребоваться более мощные технические средства. Программное средство должно корректно функционировать и на более мощном оборудовании.

# Моделирование предметной области

В данном программном средстве будут использоваться следующие структуры данных:

* Связный список.

## Связный список

Однонаправленный связный список — базовая динамическая структура данных, состоящая из узлов, каждый из которых содержит как собственно данные, так и одну на следующий узел списка. Принципиальным преимуществом перед массивом является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера, а порядок обхода списка всегда явно задаётся его внутренними связями.

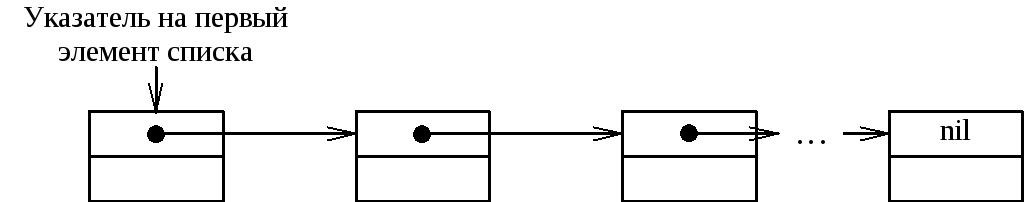


Рисунок 2.1 – Иллюстрация однонаправленного связного списка

По односвязному списку можно передвигаться только в сторону конца списка. Узнать адрес предыдущего элемента, опираясь на содержимое текущего узла, невозможно.

В двусвязном, или двунаправленном связном, списке ссылки в каждом узле указывают на предыдущий и на последующий узел.

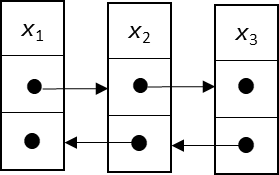


Рисунок 2.2 – Иллюстрация однонаправленного связного списка

Как и односвязный список, двусвязный допускает только последовательный доступ к элементам, но при этом даёт возможность перемещения в обе стороны. В таком списке проще производить удаление и перестановку элементов, так как легко получить доступ ко всем элементам списка, ссылки которых направлены на изменяемый элемент.

При работе со списком вводятся дополнительные ссылки на первый и последний элемент списка. Будем называть их head («голова») и tail («хвост»).

Операции вставки в конкретное место списка и удаления определённого элемента списка выполняются за O(1) при условии, что на вход даётся ссылка на узел (идущий перед точкой вставки или предшествующий узлу, который будет удалён). Если такая ссылка не предоставлена, то операции работают за O(n).

# Проектирование программного средства

## Структура программы

При разработке приложения будет использовано 5 модулей:

* GameZone – модуль графического вывода и обработки сообщений

отправленыных окну

* Logic – модуль ии и обработки взаимодействия объектов
* Shell – модуль подпрограмм и данных снарядов
* Tank – модуль подпрограмм и данных танков
* Wall – модуль подпрограмм и данных стен

## Проектирование интерфейса программного средства

### Главное окно

Лаунчер (главное меню приложения) должен состоять из простого окна с двумя кнопками, которые осуществляют запуск крестиков-ноликов и змейки соответственно. Макет главного окна приложения представлен на рисунке 3.1. 1,2,3,4 индикаторы смерти ботов.



Рисунок 3.1 – Главное окно

## Проектирование функционала программного средства

При создании игры очень важно сразу определить задачу и цели, а также хорошо составить рабочие алгоритмы. Искать ошибки в коде придётся в любом случае, но хорошо написанный алгоритм упрощает это в разы. В игре должны быть использованы следующие алгоритмы:

* обработчик нажитий пользователя;
* обработчик поведения ИИ;
* схема работы лаунчера;

### Обработчик нажатий пользователя

Обработчик нажатий пользователя необходим, чтобы позволять пользователю осуществлять управление танком и перезагрузкой карты. При создании будет использоваться оконая процедура GameWindowProc. Блок-схема алгоритма данной функции приведена на рисунке 3.2.



Рисунок 3.2 – Обработчик нажатий пользователя

### Обработчик повеведия ИИ

Обработчик повеведия ИИ необходим, чтобы организовать самостоятельное движение и выстрелы со стороны противников. При создании будет использоваться процедура Game, вынесенная в отдельный поток. Блок-схема алгоритма данной функции приведена на рисунке 3.3.



Рисунок 3.3 – Обработчик поведения ИИ

### Проверка попадания снаряда

Проверка попадания снаряда необходимо, чтобы избежать прохождения снарядов через объекты и их правильное движение по карте. Блок-схема алгоритма данной функции приведена на рисунке 3.4.



Рисунок 3.4 – Проверка попадания снаряда

# Разработка программного средства

## Проверка столкновения танка

Основой танка является алгоритм проверки столкновения с объектами.

Вызов необходимой процедуры осуществляется каждый раз, когда игрок нажимает на клавиши управления. Сперва происходит проверка жив ли танк, потом взависимости от направления движения выбираются координаты проверки столкновений со стенами и выходы за границы поля Код процедуры будет предоставлен ниже.

void MoveAndCheckingCollisionsTank(unsigned char orient, unsigned char numb) {

if (0 != tank[numb].hp) {

char flag = 1;

int i;

switch (orient)

{

case UP:

if ((tank[numb].y - tank[numb].speed) > ZONE\_TOP - 1) {

for (i = 0; i < numbWall; i++) {

if ((wall[i].y + WALL\_TOP < tank[numb].y +

TANK\_TOP + tank[numb].speed) &&

(tank[numb].y + TANK\_TOP < wall[i].y +

WALL\_BOTTOM + tank[numb].speed)) {

if (((wall[i].x + WALL\_LEFT < tank[numb].x

+ TANK\_LEFT) && (tank[numb].x +

TANK\_LEFT < wall[i].x + WALL\_RIGHT))

|| ((wall[i].x + WALL\_LEFT <

tank[numb].x + TANK\_RIGHT) &&

(tank[numb].x + TANK\_RIGHT < wall[i].x

+ WALL\_RIGHT))) {

flag = 0;

}

}

}

if (flag) {

MoveTank(UP, numb);

}

}

break;

case RIGHT:

if ((tank[numb].x + tank[numb].speed) < ZONE\_RIGHT + 1) {

for (i = 0; i < numbWall; i++) {

if ((wall[i].x + WALL\_LEFT < tank[numb].x +

TANK\_RIGHT + tank[numb].speed + 2) &&

(tank[numb].x + TANK\_RIGHT < wall[i].x +

WALL\_RIGHT + tank[numb].speed + 2)) {

if (((wall[i].y + WALL\_TOP < tank[numb].y

+ TANK\_TOP) && (tank[numb].y +

TANK\_TOP < wall[i].y + WALL\_BOTTOM))

|| ((wall[i].y + WALL\_TOP <

tank[numb].y + TANK\_BOTTOM) &&

(tank[numb].y + TANK\_BOTTOM <

wall[i].y + WALL\_BOTTOM))) {

flag = 0;

}

}

}

if (flag) {

MoveTank(RIGHT, numb);

}

}

break;

case DOWN:

if ((tank[numb].y + tank[numb].speed) < ZONE\_BOTTOM + 1) {

for (i = 0; i < numbWall; i++) {

if ((wall[i].y + WALL\_TOP < tank[numb].y +

TANK\_BOTTOM + tank[numb].speed + 2) &&

(tank[numb].y + TANK\_BOTTOM < wall[i].y +

WALL\_BOTTOM + tank[numb].speed + 2)) {

if (((wall[i].x + WALL\_LEFT < tank[numb].x

+ TANK\_LEFT) && (tank[numb].x +

TANK\_LEFT < wall[i].x + WALL\_RIGHT))

|| ((wall[i].x + WALL\_LEFT <

tank[numb].x + TANK\_RIGHT) &&

(tank[numb].x + TANK\_RIGHT <

wall[i].x + WALL\_RIGHT))) {

flag = 0;

}

}

}

if (flag) {

MoveTank(DOWN, numb);

}

}

break;

case LEFT:

if ((tank[numb].x - tank[numb].speed) > ZONE\_LEFT - 1) {

for (i = 0; i < numbWall; i++) {

if ((wall[i].x + WALL\_LEFT < tank[numb].x +

TANK\_LEFT + tank[numb].speed) &&

(tank[numb].x + TANK\_LEFT < wall[i].x +

WALL\_RIGHT + tank[numb].speed)) {

if (((wall[i].y + WALL\_TOP < tank[numb].y

+ TANK\_TOP) && (tank[numb].y +

TANK\_TOP < wall[i].y + WALL\_BOTTOM))

|| ((wall[i].y + WALL\_TOP <

tank[numb].y + TANK\_BOTTOM) &&

(tank[numb].y + TANK\_BOTTOM <

wall[i].y + WALL\_BOTTOM))) {

flag = 0;

}

}

}

if (flag) {

MoveTank(LEFT, numb);

}

}

break;

}

}

}

## Движение танка

После проверки столкновений необходимо осуществить движение танка.

Сначала проводиться проверка совподения направления танка и направления заданного клавишей, если совподение произошло танк движится, иначе поворачивает. Код функций будет предоставлен ниже.

void MoveTank(unsigned char orient, unsigned char numb) {

tank[numb].colour = !tank[numb].colour;

if (orient == tank[numb].orient) {

switch (orient)

{

case UP:

tank[numb].y -= tank[numb].speed;

break;

case RIGHT:

tank[numb].x += tank[numb].speed;

break;

case DOWN:

tank[numb].y += tank[numb].speed;

break;

case LEFT:

tank[numb].x -= tank[numb].speed;

break;

default:

break;

}

}

else {

tank[numb].orient = orient;

}

}

## Заполнение карты

Для игры требуеться создать карту.

Карта храниться ввиде массива стен, задаваемых функциями HorizontalLineWall() и VerticalLineWall(). Код функций будет предоставлен ниже.

void InitializationWall(){

hbrWall = CreateSolidBrush(RGB(0,0,100));

hpNoBorder = CreatePen(PS\_SOLID,0,RGB(0,0,100));

hpBorder = CreatePen(PS\_SOLID, 0, RGB(0, 0, 0));

SetRect(&base\_wall[0],4 ,4 ,30,17);

SetRect(&base\_wall[1],4 ,17,17,30);

numbWall = 0;

numbWall = HorizontalLineWall(numbWall,14, 81, 60);

numbWall = VerticalLineWall(numbWall, 4, 120, 180);

numbWall = VerticalLineWall(numbWall, 4, 120, 480);

numbWall = HorizontalLineWall(numbWall, 3, 240, 180);

numbWall = VerticalLineWall(numbWall, 3, 344, 232);

numbWall = HorizontalLineWall(numbWall, 3, 240, 636);

numbWall = VerticalLineWall(numbWall, 3, 344, 480);

numbWall = HorizontalLineWall(numbWall, 3, 480, 180);

numbWall = VerticalLineWall(numbWall, 3, 480, 232);

numbWall = HorizontalLineWall(numbWall, 3, 480, 636);

numbWall = VerticalLineWall(numbWall, 3, 480, 480);

numbWall = VerticalLineWall(numbWall, 4, 720, 180);

numbWall = VerticalLineWall(numbWall, 4, 720, 480);

numbWall = HorizontalLineWall(numbWall, 14, 81, 780);

}

## Выстрел танка

Для возможности прохождения игры необходима возможность уничтожения противников или игрока.

Сначала проверяеться окончание перезорядки и жив ли танк потом выделяется память для нового элемента списка. Потом исходя из положения танка выбирается место появления снаряда. Код данной функции будет приведён ниже.

void Shot(unsigned char numb) {

if (0==tank[numb].timeToReload && 0!=tank[numb].hp) {

tank[numb].timeToReload = tank[numb].reload;

SHELL\* now = NULL;

while (NULL == now) {

now = (SHELL\*)(malloc(sizeof(SHELL)));

}

now->next = NULL;

now->damage = tank[numb].damage;

switch (tank[numb].orient)

{

case UP:

now->orient = tank[numb].orient;

now->x = tank[numb].x + baseTank.orient[UP][3].left;

now->y = tank[numb].y + baseTank.orient[UP][3].top

- SHELL\_SIZE;

break;

case RIGHT:

now->orient = tank[numb].orient;

now->x = tank[numb].x +

baseTank.orient[RIGHT][3].right;

now->y = tank[numb].y + baseTank.orient[RIGHT][3].top;

break;

case DOWN:

now->orient = tank[numb].orient;

now->x = tank[numb].x + baseTank.orient[DOWN][3].left;

now->y = tank[numb].y +

baseTank.orient[DOWN][3].bottom;

break;

case LEFT:

now->orient = tank[numb].orient;

now->x = tank[numb].x + baseTank.orient[LEFT][3].left

- SHELL\_SIZE;

now->y = now->y = tank[numb].y +

baseTank.orient[LEFT][3].top;

break;

}

finShell->next = now;

finShell = now;

}

}

# Тестирование программного средства

## Запуск игры

Таблица 5.1 – Игровое окно

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка корректности поведения программы при запуске |
| Исходный набор данных | Запуск прогаммы |
| Ожидаемый результат | Открытие игрового окна |
| Полученный результат |  |

Таблица 5.2 – ИИ

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка роботоспособности ИИ |
| Исходный набор данных | Запуск программы |
| Ожидаемый результат | Танчики врагов начали движение и ведение огня |
| Полученный результат |  |

Таблица 5.3 – Индикатор убитых врагов

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка работы индикатора убитых врагов |
| Исходный набор данных | Запуск программы, убит 1 враг |
| Ожидаемый результат | 1 враг закрашен крестом |
| Полученный результат |  |

## Конец игры

Таблица 5.4 – Победа игрока

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка корректности поведения программы при победе игрока |
| Исходный набор данных | Победа игрока |
| Ожидаемый результат | Сообщение о победе |
| Полученный результат |  |

Таблица 5.5 – Змейка (касание хвоста крипера)

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка корректности поведения программы при проигрыше игрока |
| Исходный набор данных | Попадания врага в игрока |
| Ожидаемый результат | Конец игры |
| Полученный результат |  |

# Руководство пользователя

## Минимальные системные требования

Для успешного запуска программного средства «Танчики» и комфортной работы с ним необходимо соответствие минимальным системным требованиям:

* процессор Intel Celeron N4020 или лучше;
* оперативная память 128 MB или лучше;
* свободное место на диске 3 мегабайта или больше;
* операционная система Windows XP или новее.

## Установка

На установочном диске находится исполняемый файл программы и несколько библиотек. Для открытия программного средства необходимо дважды кликнуть левой кнопкой мыши по исполняемому файлу, предварительно перенеся его и сопутствующие библиотеки на компьютер.

## Игровое окно

При запуске перед пользователем открывается игровое окно. Игровое окно изображено на рисунке 6.1.

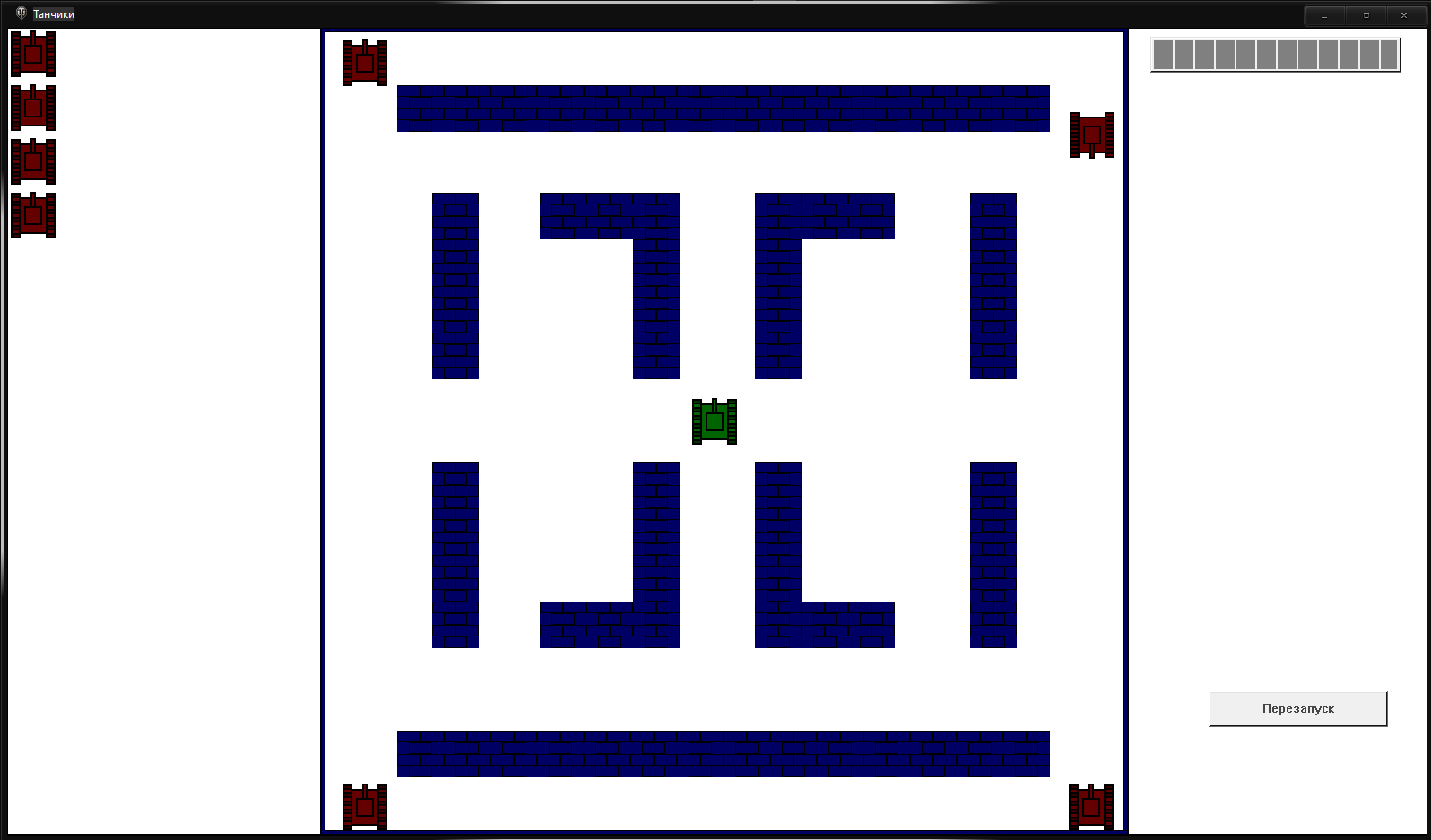


Рисунок 6.1 – Игровое окно

Слева от игрового поля находятся индикаторы танков которые надо уничтожить. При уничтожении танка индикатор закрашиваеться крестом.

Справа от игрового поля расположен индикатор перезарядки. И кнопка перезапускающая игровое поле.

Поцентру расположено игровое поле. Игрок осуществляет управление танком зелёного цвета при помощи клавиатуры.

* WASD – клавиши управления движением;
* Space – клавиша выстрела из танка;
* R – клавиша перезапуска игрового окна.

Задача пользователя уничтожить танки противника, оставшись при этом живым.

# Заключение

По итогу работы над курсовым проектом было разработано программное средство «Игровое приложение танки», обладающее графическим интерфейсом для взаимодействия с пользователем.

Данное программное средство позволяет пользователю сыграть в «Танчики». Среди преимуществ программы можно отметить интуитивно понятный интерфейс, простоту реализации, а также низкие системные требования.

При разработке данного средства были успешно выполнены следующие поставленные задачи:

* управление танком;
* сражение с ботами;
* перезапуск игры;
* перезарядка танка;
* отрисовка карты;
* счётчик убийств.

Приложение прошло все этапы тестирования и продемонстрировало корректную быструю работу.

Бесценным результатом курсового проектирования является полученный опыт работы с WINAPI. ы

B перспективе данное программное средство может быть усовершенствовано (оптимизация кода, добавление новых функций, генерирование кода по составленной схеме и наоборот и т.п.).

# Список использованной литературы

[1] Википедия – свободная энциклопедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/. – Дата доступа: 03.12.2022.

[2] Программирование на C и C++ [Электронный ресурс]. – Режим доступа: http://www.c-cpp.ru/. – Дата доступа: 03.12.2022.

[3] Wikiwand [Электронный ресурс]. – Режим доступа: https://wikiwand.com/ru/. – Дата доступа: 03.12.2022.

[4] msDN Library for Visual Studio 2008 - ENU

[5] Записи с лекций по WINAPI, Оношко Д.Е.

Приложение А

(обязательное)

GameZone.c

#include <Windows.h>

#include <CommCtrl.h>

#include <process.h>

#include "Tank.h"

#include "Wall.h"

#include "Shell.h"

#include "Logic.h"

#include "resource.h"

#define KEY\_W 87

#define KEY\_D 68

#define KEY\_S 83

#define KEY\_A 65

#define KEY\_R 82

#define KEY\_SPACE 32

#define FrameTime 33

#define GameSpeed 50

#define ReloadTime 125

#define BUTTON\_REST 1

// Класс игрового окна

const wchar\_t\* const GameZonelassClass = L"Танчики";

HDC hdcBack;

HBITMAP hbmBack;

RECT rcClient,rcText;

HANDLE thGame, thDraw, thReload;

HWND hwndGameWindow,hmndProgressBar,hwndRest;

HFONT hfText;

char T0=0, T1=0;

char F0=1, F1=1, F2=1;

void InitializeBackBuffer(HWND hWnd, int w, int h)

{

HDC hdcWindow;

hdcWindow = GetDC(hWnd);

hdcBack = CreateCompatibleDC(hdcWindow);

hbmBack = CreateCompatibleBitmap(hdcWindow, w, h);

SaveDC(hdcBack);

SelectObject(hdcBack, hbmBack);

ReleaseDC(hWnd, hdcWindow);

}

void FinalizeBackBuffer()

{

if (hdcBack)

{

RestoreDC(hdcBack, -1);

DeleteObject(hbmBack);

DeleteDC(hdcBack);

hdcBack = 0;

}

}

void Draw() {

while (F0)

{

Sleep(FrameTime);

InvalidateRect(hwndGameWindow, NULL, 0);

}

F0 = 1;

}

void Game() {

while (F1)

{

T1 = 1;

while (T0)

{

Sleep(10);

}

CheckingCollisionsShell();

T1 = 0;

AI();

CheckGame();

Sleep(GameSpeed);

}

F1 = 1;

}

void Reload() {

while (F2)

{

TankReload();

if (0 != tank[PLAYER].timeToReload) {

SendMessage(hmndProgressBar, PBM\_STEPIT, 0, 0);

Sleep(ReloadTime);

SendMessage(hmndProgressBar, PBM\_STEPIT, 0, 0);

Sleep(ReloadTime);

SendMessage(hmndProgressBar, PBM\_STEPIT, 0, 0);

Sleep(ReloadTime);

SendMessage(hmndProgressBar, PBM\_STEPIT, 0, 0);

Sleep(ReloadTime);

SendMessage(hmndProgressBar, PBM\_STEPIT, 0, 0);

Sleep(ReloadTime);

SendMessage(hmndProgressBar, PBM\_STEPIT, 0, 0);

Sleep(ReloadTime);

SendMessage(hmndProgressBar, PBM\_STEPIT, 0, 0);

Sleep(ReloadTime);

SendMessage(hmndProgressBar, PBM\_STEPIT, 0, 0);

Sleep(ReloadTime);

}

else {

Sleep(ReloadTime \* 8);

}

}

F2 = 1;

}

// Оконная процедура

LRESULT CALLBACK GameZoneProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam) {

PAINTSTRUCT ps;

HDC hdc;

// Обработка сообщений

switch (uMsg) {

case WM\_CREATE:

GetClientRect(hWnd, &rcClient);

InitializeBackBuffer(hWnd, rcClient.right, rcClient.bottom);

SetRect(&rcText, 500, 400, 1100, 500);

break;

case WM\_SIZE:

GetClientRect(hWnd, &rcClient);

FinalizeBackBuffer();

InitializeBackBuffer(hWnd, rcClient.right, rcClient.bottom);

break;

case WM\_COMMAND:

if (BUTTON\_REST == LOWORD(wParam)) {

FinalizeTank();

FinalizeShell();

FinalizeWall();

game = 1;

InitializationWall();

InitializationShell();

InitializationTank(5);

InitializationAI();

SetFocus(hwndGameWindow);

}

break;

case WM\_KEYDOWN:

switch (wParam) {

case KEY\_W:

MoveAndCheckingCollisionsTank(UP, PLAYER);

break;

case KEY\_D:

MoveAndCheckingCollisionsTank(RIGHT, PLAYER);

break;

case KEY\_S:

MoveAndCheckingCollisionsTank(DOWN, PLAYER);

break;

case KEY\_A:

MoveAndCheckingCollisionsTank(LEFT, PLAYER);

break;

case KEY\_SPACE:

Shot(PLAYER);

break;

case KEY\_R:

SendMessage(hwndGameWindow, WM\_COMMAND, BUTTON\_REST, 0);

break;

}

break;

// Риссования

case WM\_PAINT:

Rectangle(hdcBack, ZONE\_LEFT, ZONE\_TOP, ZONE\_RIGHT + 60, ZONE\_BOTTOM + 60);

if (game && 0!=tank[PLAYER].hp) {

DrawTank(hdcBack);

T0 = 1;

while (T1) {

Sleep(10);

}

DrawShell(hdcBack);

T0 = 0;

DrawWall(hdcBack);

}

else {

if (0==tank[PLAYER].hp) {

SaveDC(hdcBack);

SetTextColor(hdcBack,RGB(255,0,0));

SelectObject(hdcBack, hfText);

DrawText(hdcBack, L"YOU DIED", 8, &rcText, DT\_CENTER);

ReleaseDC(hdcBack, -1);

}

else {

SaveDC(hdcBack);

SetTextColor(hdcBack, RGB(0,255,0));

SelectObject(hdcBack, hfText);

DrawText(hdcBack, L"YOU WIN", 7, &rcText, DT\_CENTER);

ReleaseDC(hdcBack, -1);

}

}

Rectangle(hdcBack, 0, 0, ZONE\_LEFT, ZONE\_BOTTOM + 60);

Rectangle(hdcBack, ZONE\_RIGHT + 60, ZONE\_TOP, 1600, 900);

DrawAllTank(hdcBack);

hdc = BeginPaint(hWnd, &ps);

BitBlt(hdc, 0, 0, rcClient.right, rcClient.bottom, hdcBack, 0, 0, SRCCOPY);

EndPaint(hWnd, &ps);

break;

case WM\_DESTROY:

F0 = 0;

F1 = 0;

F2 = 0;

while (!F0 || !F1 || !F2) {

Sleep(500);

}

FinalizeBackBuffer();

PostQuitMessage(0);

break;

}

return DefWindowProc(hWnd, uMsg, wParam, lParam);

}

INT WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {

MSG msg;

WNDCLASSEX wcex;

// Регестрация оконого класса

memset(&wcex, 0, sizeof wcex);

wcex.cbSize = sizeof wcex;

wcex.style = CS\_GLOBALCLASS;

wcex.lpfnWndProc = GameZoneProc;

wcex.hCursor = LoadCursor(0, IDC\_ARROW);

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_ICON2));

wcex.hIconSm = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_ICON2));

wcex.hbrBackground = (HBRUSH)(COLOR\_BACKGROUND + 1);

wcex.lpszClassName = GameZonelassClass;

RegisterClassEx(&wcex);

InitializationWall();

InitializationShell();

InitializationTank(5);

InitializationAI();

hfText = CreateFont(100,0,0,FW\_BOLD,0,0,0,0,0,0,0,0,0,0);

thDraw = (HANDLE)\_beginthread(Draw, 0, 0);

thGame = (HANDLE)\_beginthread(Game, 0, 0);

thReload = (HANDLE)\_beginthread(Reload, 0, 0);

// Создание окна

hwndGameWindow = CreateWindowEx(0, GameZonelassClass, GameZonelassClass,

WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 0, 0, 1600, 940, 0, 0, 0, NULL);

hmndProgressBar = CreateWindowEx(PBS\_SMOOTH, PROGRESS\_CLASS, L"Reload",

WS\_CHILD | WS\_VISIBLE, ZONE\_RIGHT+85, 10, 280, 40, hwndGameWindow,

0, 0, NULL);

hwndRest = CreateWindowEx(BS\_PUSHBUTTON, L"BUTTON", L"Перезапуск",

WS\_VISIBLE | WS\_CHILD, ZONE\_RIGHT + 150, ZONE\_BOTTOM - 100,

200, 40, hwndGameWindow, (HMENU)BUTTON\_REST, 0, NULL);

SendMessage(hmndProgressBar, PBM\_SETRANGE, 0, MAKELPARAM(0, 8));

SendMessage(hmndProgressBar, PBM\_SETSTEP, 1, 0);

SendMessage(hmndProgressBar, PBM\_DELTAPOS, 8, 0);

while (GetMessage(&msg, 0, 0, 0))

{

DispatchMessage(&msg);

}

DeleteObject(hfText);

FinalizeTank();

FinalizeShell();

FinalizeWall();

return msg.wParam;

}

**Logic.h**

#pragma once

extern char game;

void MoveAndCheckingCollisionsTank(

unsigned char orient, unsigned char numb);

void Shot(unsigned char numb);

void CheckingCollisionsShell();

void InitializationAI();

void AI();

void CheckGame();

Logic.c

#include "Logic.h"

#include "Wall.h"

#include "Tank.h"

#include "Shell.h"

#include <time.h>

#define NIL 255

#define colision 25

typedef struct \_GRID\_MOVE {

unsigned char cell[4];

short int x;

short int y;

}GRID\_MOVE;

GRID\_MOVE gridMove[19];

char game = 1;

void Shot(unsigned char numb) {

if (0==tank[numb].timeToReload && 0!=tank[numb].hp) {

tank[numb].timeToReload = tank[numb].reload;

SHELL\* now = NULL;

while (NULL == now) {

now = (SHELL\*)(malloc(sizeof(SHELL)));

}

now->next = NULL;

now->damage = tank[numb].damage;

switch (tank[numb].orient)

{

case UP:

now->orient = tank[numb].orient;

now->x = tank[numb].x + baseTank.orient[UP][3].left;

now->y = tank[numb].y + baseTank.orient[UP][3].top - SHELL\_SIZE;

break;

case RIGHT:

now->orient = tank[numb].orient;

now->x = tank[numb].x + baseTank.orient[RIGHT][3].right;

now->y = tank[numb].y + baseTank.orient[RIGHT][3].top;

break;

case DOWN:

now->orient = tank[numb].orient;

now->x = tank[numb].x + baseTank.orient[DOWN][3].left;

now->y = tank[numb].y + baseTank.orient[DOWN][3].bottom;

break;

case LEFT:

now->orient = tank[numb].orient;

now->x = tank[numb].x + baseTank.orient[LEFT][3].left - SHELL\_SIZE;

now->y = now->y = tank[numb].y + baseTank.orient[LEFT][3].top;

break;

}

finShell->next = now;

finShell = now;

}

}

void MoveAndCheckingCollisionsTank(unsigned char orient, unsigned char numb) {

if (0 != tank[numb].hp) {

char flag = 1;

int i;

switch (orient)

{

case UP:

if ((tank[numb].y - tank[numb].speed) > ZONE\_TOP - 1) {

for (i = 0; i < numbWall; i++) {

if ((wall[i].y + WALL\_TOP < tank[numb].y +

TANK\_TOP + tank[numb].speed) &&

(tank[numb].y + TANK\_TOP < wall[i].y +

WALL\_BOTTOM + tank[numb].speed)) {

if (((wall[i].x + WALL\_LEFT < tank[numb].x

+ TANK\_LEFT) && (tank[numb].x +

TANK\_LEFT < wall[i].x + WALL\_RIGHT))

|| ((wall[i].x + WALL\_LEFT <

tank[numb].x + TANK\_RIGHT) &&

(tank[numb].x + TANK\_RIGHT <

wall[i].x + WALL\_RIGHT))) {

flag = 0;

}

}

}

if (flag) {

MoveTank(UP, numb);

}

}

break;

case RIGHT:

if ((tank[numb].x + tank[numb].speed) < ZONE\_RIGHT + 1) {

for (i = 0; i < numbWall; i++) {

if ((wall[i].x + WALL\_LEFT < tank[numb].x +

TANK\_RIGHT + tank[numb].speed + 2) &&

(tank[numb].x + TANK\_RIGHT < wall[i].x +

WALL\_RIGHT + tank[numb].speed + 2)) {

if (((wall[i].y + WALL\_TOP < tank[numb].y

+ TANK\_TOP) && (tank[numb].y +

TANK\_TOP < wall[i].y + WALL\_BOTTOM)) || ((wall[i].y + WALL\_TOP < tank[numb].y + TANK\_BOTTOM) && (tank[numb].y + TANK\_BOTTOM < wall[i].y + WALL\_BOTTOM))) {

flag = 0;

}

}

}

if (flag) {

MoveTank(RIGHT, numb);

}

}

break;

case DOWN:

if ((tank[numb].y + tank[numb].speed) < ZONE\_BOTTOM + 1) {

for (i = 0; i < numbWall; i++) {

if ((wall[i].y + WALL\_TOP < tank[numb].y +

TANK\_BOTTOM + tank[numb].speed + 2) &&

(tank[numb].y + TANK\_BOTTOM < wall[i].y +

WALL\_BOTTOM + tank[numb].speed + 2)) {

if (((wall[i].x + WALL\_LEFT < tank[numb].x

+ TANK\_LEFT) && (tank[numb].x +

TANK\_LEFT < wall[i].x + WALL\_RIGHT))

|| ((wall[i].x + WALL\_LEFT <

tank[numb].x + TANK\_RIGHT) &&

(tank[numb].x + TANK\_RIGHT <

wall[i].x + WALL\_RIGHT))) {

flag = 0;

}

}

}

if (flag) {

MoveTank(DOWN, numb);

}

}

break;

case LEFT:

if ((tank[numb].x - tank[numb].speed) > ZONE\_LEFT - 1) {

for (i = 0; i < numbWall; i++) {

if ((wall[i].x + WALL\_LEFT < tank[numb].x +

TANK\_LEFT + tank[numb].speed) &&

(tank[numb].x + TANK\_LEFT < wall[i].x +

WALL\_RIGHT + tank[numb].speed)) {

if (((wall[i].y + WALL\_TOP < tank[numb].y

+ TANK\_TOP) && (tank[numb].y +

TANK\_TOP < wall[i].y + WALL\_BOTTOM))

|| ((wall[i].y + WALL\_TOP <

tank[numb].y + TANK\_BOTTOM) &&

(tank[numb].y + TANK\_BOTTOM <

wall[i].y + WALL\_BOTTOM))) {

flag = 0;

}

}

}

if (flag) {

MoveTank(LEFT, numb);

}

}

break;

}

}

}

void CheckingCollisionsShell() {

unsigned char flag, kill;

int i;

SHELL\* next, \* pred = headShell;

for (SHELL\* now = headShell->next; NULL != now; ) {

flag = 1;

kill = 255;

switch (now->orient)

{

case UP:

if (now->y > ZONE\_TOP - 20) {

for (i = 0; i < numbWall; i++) {

if ((wall[i].y + WALL\_TOP < now->y) && (now->y <

wall[i].y + WALL\_BOTTOM)) {

if (((wall[i].x + WALL\_LEFT < now->x) &&

(now->x < wall[i].x + WALL\_RIGHT))

|| ((wall[i].x + WALL\_LEFT < now->x

+ SHELL\_SIZE) && (now->x +

SHELL\_SIZE < wall[i].x +

WALL\_RIGHT))) {

flag = 0;

}

}

}

for (i = 0; i < numbTank; i++) {

if ((tank[i].y + baseTank.orient[UP][0].top <

now->y) && (now->y < tank[i].y +

baseTank.orient[UP][1].bottom)) {

if (((tank[i].x +

baseTank.orient[UP][0].left <

now->x) && (now->x < tank[i].x +

baseTank.orient[UP][1].right)) ||

((tank[i].x +

baseTank.orient[UP][0].left <

now->x + SHELL\_SIZE) && (now->x +

SHELL\_SIZE < tank[i].x +

baseTank.orient[UP][1].right))) {

flag = 0;

kill = i;

}

}

}

}

else {

flag = 0;

}

break;

case RIGHT:

if (now->x < ZONE\_RIGHT + 80 - SHELL\_SIZE) {

for (i = 0; i < numbWall; i++) {

if ((wall[i].x + WALL\_LEFT < now->x+SHELL\_SIZE)

&& (now->x+SHELL\_SIZE < wall[i].x +

WALL\_RIGHT)) {

if (((wall[i].y + WALL\_TOP < now->y) &&

(now->y < wall[i].y + WALL\_BOTTOM))

|| ((wall[i].y + WALL\_TOP < now->y + SHELL\_SIZE) && (now->y + SHELL\_SIZE < wall[i].y + WALL\_BOTTOM))) {

flag = 0;

}

}

}

for (i = 0; i < numbTank; i++) {

if ((tank[i].x + baseTank.orient[UP][0].left <

now->x + SHELL\_SIZE) && (now->x +

SHELL\_SIZE < tank[i].x +

baseTank.orient[UP][1].right)) {

if (((tank[i].y +

baseTank.orient[UP][0].top < now->y) &&

(now->y < tank[i].y +

baseTank.orient[UP][1].bottom)) ||

((tank[i].y + baseTank.orient[UP][0].top

< now->y + SHELL\_SIZE) && (now->y +

SHELL\_SIZE < tank[i].y +

baseTank.orient[UP][1].bottom))) {

flag = 0;

kill = i;

}

}

}

}

else {

flag = 0;

}

break;

case DOWN:

if (now->y < ZONE\_BOTTOM + 80 - SHELL\_SIZE) {

for (i = 0; i < numbWall; i++) {

if ((wall[i].y + WALL\_TOP < now->y +

SHELL\_SIZE)&& (now->y + SHELL\_SIZE <

wall[i].y + WALL\_BOTTOM)) {

if (((wall[i].x + WALL\_LEFT < now->x) &&

(now->x < wall[i].x + WALL\_RIGHT)) || ((wall[i].x + WALL\_LEFT < now->x+SHELL\_SIZE) && (now->x + SHELL\_SIZE < wall[i].x + WALL\_RIGHT))) {

flag = 0;

}

}

}

for (i = 0; i < numbTank; i++) {

if ((tank[i].y + baseTank.orient[UP][0].top <

now->y + SHELL\_SIZE) && (now->y +

SHELL\_SIZE < tank[i].y +

baseTank.orient[UP][1].bottom)) {

if (((tank[i].x +

baseTank.orient[UP][0].left <

now->x) && (now->x < tank[i].x +

baseTank.orient[UP][1].right)) ||

((tank[i].x +

baseTank.orient[UP][0].left <

now->x + SHELL\_SIZE) &&

(now->x + SHELL\_SIZE < tank[i].x +

baseTank.orient[UP][1].right))) {

flag = 0;

kill = i;

}

}

}

}

else {

flag = 0;

}

break;

case LEFT:

if (now->x > ZONE\_LEFT - 20) {

for (i = 0; i < numbWall; i++) {

if ((wall[i].x + WALL\_LEFT < now->x) &&

(now->x < wall[i].x + WALL\_RIGHT)) {

if (((wall[i].y + WALL\_TOP < now->y) &&

(now->y < wall[i].y + WALL\_BOTTOM))

|| ((wall[i].y + WALL\_TOP < now->y+SHELL\_SIZE) && (now->y + SHELL\_SIZE < wall[i].y + WALL\_BOTTOM))) {

flag = 0;

}

}

}

for (i = 0; i < numbTank; i++) {

if ((tank[i].x + baseTank.orient[UP][0].left <

now->x) && (now->x < tank[i].x +

baseTank.orient[UP][1].right)) {

if (((tank[i].y +

baseTank.orient[UP][0].top < now->y) && (now->y < tank[i].y + baseTank.orient[UP][1].bottom)) || ((tank[i].y + baseTank.orient[UP][0].top < now->y + SHELL\_SIZE) && (now->y + SHELL\_SIZE < tank[i].y + baseTank.orient[UP][1].bottom))) {

flag = 0;

kill = i;

}

}

}

}

else {

flag = 0;

}

break;

}

if (flag) {

MoveShell(now);

}

else {

next = now->next;

DeleteShell(now, pred);

now = next;

if (255!=kill) {

tank[kill].hp = 0;

tank[kill].x = 0;

tank[kill].y = 0;

}

}

pred = now;

if (NULL != now) {

now = now->next;

}

}

}

void AddElemGridMove(unsigned char numb, unsigned char numb0, unsigned char numb1, unsigned char numb2, unsigned char numb3, short int x, short int y) {

gridMove[numb].cell[0] = numb0;

gridMove[numb].cell[1] = numb1;

gridMove[numb].cell[2] = numb2;

gridMove[numb].cell[3] = numb3;

gridMove[numb].x = x;

gridMove[numb].y = y;

}

void InitializationAI() {

AddElemGridMove(0 , NIL, 1 , 2 , NIL, 370 , 10 );

AddElemGridMove(1 , NIL, NIL, 6 , 0 , 1180, 10);

AddElemGridMove(2 , 0 , 3 , 7 , NIL, 370 , 120);

AddElemGridMove(3 , NIL, 4 , 8 , 2 , 540 , 120);

AddElemGridMove(4 , NIL, 5 , 9 , 3 , 770 , 120);

AddElemGridMove(5 , NIL, 6 , 10 , 4 , 1010, 120);

AddElemGridMove(6 , 1 , NIL, 11 , 5 , 1180, 120);

AddElemGridMove(7 , 2 , 8 , 12 , NIL, 370 , 400);

AddElemGridMove(8 , 3 , 9 , 13 , 7 , 540 , 400);

AddElemGridMove(9 , 4 , 10 , 14 , 8 , 770 , 400);

AddElemGridMove(10, 5 , 11 , 15 , 9 , 1010, 400);

AddElemGridMove(11, 6 , NIL, 16 , 10 , 1180, 400);

AddElemGridMove(12, 7 , 13 , 17 , NIL, 370 , 700);

AddElemGridMove(13, 8 , 14 , NIL, 12 , 540 , 700);

AddElemGridMove(14, 9 , 15 , NIL, 13 , 770 , 700);

AddElemGridMove(15, 10 , 16 , NIL, 14 , 1010, 700);

AddElemGridMove(16, 11 , NIL, 18 , 15 , 1180, 700);

AddElemGridMove(17, 12 , 18 , NIL, NIL, 370 , 840);

AddElemGridMove(18, 16 , NIL, NIL, 17 , 1180, 840);

}

void WayAI() {

unsigned char cell;

for (int i = 1; i < numbTank; i++) {

if (0 < tank[i].hp && 0 == tank[i].move) {

cell = NIL;

while (NIL == cell) {

cell = gridMove[tank[i].cell].cell[rand() % 4];

}

tank[i].move = 1;

tank[i].cell = cell;

}

}

}

void MoveAI() {

for (int i = 1; i < numbTank; i++) {

if (tank[i].y != gridMove[tank[i].cell].y && tank[i].x != gridMove[tank[i].cell].x) {

if (tank[i].y > gridMove[tank[i].cell].y) {

MoveAndCheckingCollisionsTank(UP, i);

}

else {

if (tank[i].x < gridMove[tank[i].cell].x) {

MoveAndCheckingCollisionsTank(RIGHT, i);

}

else {

if (tank[i].y < gridMove[tank[i].cell].y) {

MoveAndCheckingCollisionsTank(DOWN, i);

}

else {

if (tank[i].x > gridMove[tank[i].cell].x) {

MoveAndCheckingCollisionsTank(LEFT, i);

}

}

}

}

}

else {

tank[i].move = 0;

}

if (tank[i].x - colision < tank[PLAYER].x && tank[PLAYER].x < tank[i].x + colision) {

if (tank[i].y > tank[PLAYER].y) {

tank[i].orient = UP;

}

else {

tank[i].orient = DOWN;

}

Shot(i);

}

else {

if (tank[i].y - colision < tank[PLAYER].y && tank[PLAYER].y < tank[i].y + colision) {

if (tank[i].x > tank[PLAYER].x) {

tank[i].orient = LEFT;

}

else {

tank[i].orient = RIGHT;

}

Shot(i);

}

}

}

}

void AI() {

srand(time(NULL));

WayAI();

MoveAI();

}

void CheckGame() {

game = 0;

for (int i = 1; i < numbTank; i++) {

if (0 != tank[i].hp) {

game = 1;

}

}

}

Tank.h

#pragma once

#include <Windows.h>

#define UP 0

#define RIGHT 1

#define DOWN 2

#define LEFT 3

#define TANK\_TOP 5

#define TANK\_BOTTOM 53

#define TANK\_RIGHT 54

#define TANK\_LEFT 5

#define SIZE\_TANK 60

#define ZONE\_TOP 0

#define ZONE\_BOTTOM 840

#define ZONE\_RIGHT 1190

#define ZONE\_LEFT 350

#define PLAYER 0

typedef struct \_BASE\_TANK {

RECT orient[4][6];

}BASE\_TANK;

typedef struct \_TANK {

short int x;

short int y;

unsigned char orient;

unsigned char colour;

unsigned char timeToReload;

unsigned char cell;

unsigned char move;

unsigned char hp;

unsigned char speed;

unsigned char damage;

unsigned char reload;

}TANK;

extern BASE\_TANK baseTank;

extern TANK tank[5];

extern unsigned char numbTank;

void InitializationTank(unsigned char numb);

void FinalizeTank();

void DrawTank(HDC hdc);

void DrawAllTank(HDC hdc);

void MoveTank(unsigned char orient, unsigned char numb);

void TankReload();

Tank.c

#include "Tank.h"

#define ENEMY 1

#define BASE\_SPEED 5

#define BASE\_DAMAGE 1

#define BASE\_HP 1

#define BASE\_RELOAD 2

BASE\_TANK baseTank;

TANK tank[5];

unsigned char numbTank;

HBRUSH hbrPlayer, hbrPalayerLight, hbrEnemy, hbrEnemyLight;

HPEN hpCross, hpTank;

void InitializationTank(unsigned char numb) {

hbrPlayer = CreateSolidBrush(RGB(0,100,0));

hbrPalayerLight = CreateSolidBrush(RGB(0,50,0));

hbrEnemy = CreateSolidBrush(RGB(100,0,0));

hbrEnemyLight = CreateSolidBrush(RGB(50,0,0));

hpCross = CreatePen(PS\_SOLID, 10, RGB(255, 100, 100));

hpTank = CreatePen(PS\_SOLID, 2, RGB(0, 0, 0));

numbTank = numb;

int i;

//гусеница 1

SetRect(&baseTank.orient[UP][0], 5 , 5 , 15, 53);

//гусеница 2

SetRect(&baseTank.orient[UP][1], 44, 5, 54, 53);

//корпус

SetRect(&baseTank.orient[UP][2], 14, 10, 45, 50);

//орудие

SetRect(&baseTank.orient[UP][3], 27, 4 , 32, 21);

//башня

SetRect(&baseTank.orient[UP][4], 20, 20, 39, 40);

//траки

SetRect(&baseTank.orient[UP][5], 5 , 5 , 15, 10);

for (i = 0; i < 6; i++) {

SetRect(&baseTank.orient[RIGHT][i], SIZE\_TANK - baseTank.orient[UP][i].bottom, baseTank.orient[UP][i].left ,

SIZE\_TANK - baseTank.orient[UP][i].top , baseTank.orient[UP][i].right);

}

for (i = 0; i < 6; i++) {

SetRect(&baseTank.orient[DOWN][i], SIZE\_TANK - baseTank.orient[UP][i].right , SIZE\_TANK - baseTank.orient[UP][i].bottom,

SIZE\_TANK - baseTank.orient[UP][i].left , SIZE\_TANK - baseTank.orient[UP][i].top );

}

for (i = 0; i < 6; i++) {

SetRect(&baseTank.orient[LEFT][i], baseTank.orient[UP][i].top , SIZE\_TANK - baseTank.orient[UP][i].right,

baseTank.orient[UP][i].bottom, SIZE\_TANK - baseTank.orient[UP][i].left);

}

memset(tank,0, sizeof(TANK)\* numbTank);

tank[PLAYER].x = ZONE\_LEFT+410;

tank[PLAYER].y = 410;

tank[1].x = 370;

tank[1].y = 10;

tank[2].x = 1180;

tank[2].y = 10;

tank[3].x = 370;

tank[3].y = 840;

tank[4].x = 1180;

tank[4].y = 840;

tank[1].cell = 0;

tank[2].cell = 1;

tank[3].cell = 17;

tank[4].cell = 18;

for (i = 0; i < numbTank; i++) {

tank[i].speed = BASE\_SPEED;

tank[i].damage = BASE\_DAMAGE;

tank[i].hp = BASE\_HP;

tank[i].reload = BASE\_RELOAD;

}

}

void FinalizeTank() {

DeleteObject(hbrPlayer);

DeleteObject(hbrPalayerLight);

DeleteObject(hbrEnemy);

DeleteObject(hbrEnemyLight);

}

void DrawTank(HDC hdc) {

SaveDC(hdc);

SelectObject(hdc, hpTank);

int i, x, y, k, n, j;

unsigned char orient;

for (j = 0; j < numbTank; j++) {

orient = tank[j].orient;

if (tank[j].colour) {

if (j) {

SelectObject(hdc, hbrEnemyLight);

}

else {

SelectObject(hdc, hbrPalayerLight);

}

}

else {

if (j) {

SelectObject(hdc, hbrEnemy);

}

else {

SelectObject(hdc, hbrPlayer);

}

}

// Основные прямоугольники танка

for (i = 0; i < 2; i++) {

Rectangle(hdc, tank[j].x + baseTank.orient[orient][i].left, tank[j].y + baseTank.orient[orient][i].top,

tank[j].x + baseTank.orient[orient][i].right, tank[j].y + baseTank.orient[orient][i].bottom);

}

if (j) {

SelectObject(hdc, hbrEnemy);

}

else {

SelectObject(hdc, hbrPlayer);

}

for (i = 2; i < 5; i++) {

Rectangle(hdc, tank[j].x + baseTank.orient[orient][i].left, tank[j].y + baseTank.orient[orient][i].top,

tank[j].x + baseTank.orient[orient][i].right, tank[j].y + baseTank.orient[orient][i].bottom);

}

// переделать

x = 0;

i = 5;

y = 0;

if (tank[j].colour) {

if (j) {

SelectObject(hdc, hbrEnemy);

}

else {

SelectObject(hdc, hbrPlayer);

}

}

else {

if (j) {

SelectObject(hdc, hbrEnemyLight);

}

else {

SelectObject(hdc, hbrPalayerLight);

}

}

for (n = 0; n < 2; n++) {

switch (orient)

{

case UP:

case DOWN:

y = 0;

break;

case RIGHT:

case LEFT:

x = 0;

break;

}

for (k = 0; k < 6; k++) {

Rectangle(hdc, tank[j].x + baseTank.orient[orient][i].left + x, tank[j].y + baseTank.orient[orient][i].top + y,

tank[j].x + baseTank.orient[orient][i].right + x, tank[j].y + baseTank.orient[orient][i].bottom + y);

switch (orient)

{

case UP:

y += 9;

break;

case RIGHT:

x -= 9;

break;

case DOWN:

y -= 9;

break;

case LEFT:

x += 9;

break;

default:

break;

}

}

switch (orient)

{

case UP:

x += 39;

break;

case RIGHT:

y += 39;

break;

case DOWN:

x -= 39;

break;

case LEFT:

y -= 39;

break;

default:

break;

}

}

}

RestoreDC(hdc, -1);

}

void DrawAllTank(HDC hdc) {

int i, x, y, tankX = 0, tankY = 0, n, k;

SaveDC(hdc);

for (int j = 1; j < numbTank; j++) {

SelectObject(hdc, hpTank);

if (tank[j].colour) {

SelectObject(hdc, hbrEnemyLight);

}

else {

SelectObject(hdc, hbrEnemy);

}

// Основные прямоугольники танка

for (i = 0; i < 2; i++) {

Rectangle(hdc, baseTank.orient[UP][i].left + tankX,

baseTank.orient[UP][i].top + tankY,

baseTank.orient[UP][i].right + tankX,

baseTank.orient[UP][i].bottom + tankY);

}

SelectObject(hdc, hbrEnemy);

for (i = 2; i < 5; i++) {

Rectangle(hdc, baseTank.orient[UP][i].left + tankX,

baseTank.orient[UP][i].top + tankY,

baseTank.orient[UP][i].right + tankX,

baseTank.orient[UP][i].bottom + tankY);

}

// переделать

x = 0;

i = 5;

y = 0;

if (tank[j].colour) {

SelectObject(hdc, hbrEnemy);

}

else {

SelectObject(hdc, hbrEnemyLight);

}

for (n = 0; n < 2; n++) {

y = 0;

for (k = 0; k < 6; k++) {

Rectangle(hdc, baseTank.orient[UP][i].left + x +

tankX, baseTank.orient[UP][i].top + y + tankY,

baseTank.orient[UP][i].right + x + tankX,

baseTank.orient[UP][i].bottom + y + tankY);

y += 9;

}

x += 39;

}

if (0 == tank[j].hp) {

SelectObject(hdc, hpCross);

MoveToEx(hdc, baseTank.orient[UP][0].left + tankX,

baseTank.orient[UP][0].top + tankY, NULL);

LineTo(hdc, baseTank.orient[UP][1].right + tankX,

baseTank.orient[UP][1].bottom + tankY);

MoveToEx(hdc, baseTank.orient[UP][0].left + tankX,

baseTank.orient[UP][0].bottom + tankY, NULL);

LineTo(hdc, baseTank.orient[UP][1].right + tankX,

baseTank.orient[UP][1].top + tankY);

}

tankY += 60;

}

RestoreDC(hdc, -1);

}

void MoveTank(unsigned char orient, unsigned char numb) {

tank[numb].colour = !tank[numb].colour;

if (orient == tank[numb].orient) {

switch (orient)

{

case UP:

tank[numb].y -= tank[numb].speed;

break;

case RIGHT:

tank[numb].x += tank[numb].speed;

break;

case DOWN:

tank[numb].y += tank[numb].speed;

break;

case LEFT:

tank[numb].x -= tank[numb].speed;

break;

default:

break;

}

}

else {

tank[numb].orient = orient;

}

}

void TankReload() {

for (int i = 0; i < numbTank; i++) {

if (0 != tank[i].timeToReload) {

tank[i].timeToReload--;

}

}

}

Shell.h

#pragma once

#include <Windows.h>

#define SHELL\_SIZE 4

typedef struct \_SHELL {

short int x;

short int y;

unsigned char damage;

unsigned char orient;

struct \_SHELL\* next;

}SHELL;

extern SHELL\* finShell;

extern SHELL\* headShell;

void InitializationShell();

void FinalizeShell();

void DrawShell(HDC hdc);

void MoveShell(SHELL\* now);

void DeleteShell(SHELL\* now, SHELL\* pred);

Shell.c

#include "Shell.h"

#define UP 0

#define RIGHT 1

#define DOWN 2

#define LEFT 3

#define BASE\_SHELL\_SPEED 20

RECT baseShell;

SHELL \*headShell;

SHELL\* finShell;

HBRUSH hbrShell;

void InitializationShell() {

SetRect(&baseShell, 0, 0, SHELL\_SIZE + 1, SHELL\_SIZE + 1);

headShell = NULL;

while (NULL == headShell) {

headShell = (SHELL\*)(malloc(sizeof(SHELL)));

}

memset(headShell, 0, sizeof(SHELL));

finShell = headShell;

hbrShell = CreateSolidBrush(RGB(100, 0, 0));

}

void FinalizeShell() {

for (SHELL\*pred = headShell; NULL != headShell;) {

headShell = headShell->next;

free(pred);

pred = headShell;

}

DeleteObject(hbrShell);

}

void DrawShell(HDC hdc) {

SaveDC(hdc);

SelectObject(hdc, hbrShell);

for (SHELL\* now = headShell->next; NULL != now; now = now->next) {

Rectangle(hdc, baseShell.left + now->x, baseShell.left + now->y,

baseShell.right + now->x, baseShell.bottom + now->y);

}

RestoreDC(hdc, -1);

}

void MoveShell(SHELL \*now) {

switch (now->orient)

{

case UP:

now->y -= BASE\_SHELL\_SPEED;

break;

case RIGHT:

now->x += BASE\_SHELL\_SPEED;

break;

case DOWN:

now->y += BASE\_SHELL\_SPEED;

break;

case LEFT:

now->x -= BASE\_SHELL\_SPEED;

break;

}

}

void DeleteShell(SHELL\* now, SHELL\* pred) {

pred->next = now->next;

if (now == finShell) {

finShell = pred;

finShell->next = NULL;

}

free(now);

}

Wall.h

#pragma once

#include <Windows.h>

#define WALL\_TOP 4

#define WALL\_RIGHT 56

#define WALL\_BOTTOM 56

#define WALL\_LEFT 4

typedef struct \_WALL {

short int x;

short int y;

}WALL;

extern RECT base\_wall[2];

extern unsigned char numbWall;

extern WALL wall[68];

void InitializationWall();

void FinalizeWall();

void DrawWall(HDC hdc);

Wall.c

#include"Wall.h"

#define ZONE\_TOP 0

#define ZONE\_BOTTOM 840

#define ZONE\_RIGHT 1190

#define ZONE\_LEFT 350

RECT base\_wall[2];

unsigned char numbWall;

WALL wall[68];

HPEN hpNoBorder, hpBorder;

HBRUSH hbrWall;

unsigned char HorizontalLineWall(unsigned char StartInd,unsigned char numb,

short int x, short int y) {

x += ZONE\_LEFT;

y += ZONE\_TOP;

wall[StartInd].x = x;

wall[StartInd].y = y;

for (int i = StartInd+1; i < StartInd+numb; i++) {

wall[i].x = wall[i - 1].x + 52;

wall[i].y = y;

}

return StartInd += numb;

}

unsigned char VerticalLineWall(unsigned char StartInd, unsigned char numb,

short int x, short int y) {

x += ZONE\_LEFT;

y += ZONE\_TOP;

wall[StartInd].x = x;

wall[StartInd].y = y;

for (int i = StartInd + 1; i < StartInd + numb; i++) {

wall[i].x = x;

wall[i].y = wall[i - 1].y + 52;

}

return StartInd += numb;

}

void InitializationWall(){

hbrWall = CreateSolidBrush(RGB(0,0,100));

hpNoBorder = CreatePen(PS\_SOLID,0,RGB(0,0,100));

hpBorder = CreatePen(PS\_SOLID, 0, RGB(0, 0, 0));

SetRect(&base\_wall[0],4 ,4 ,30,17);

SetRect(&base\_wall[1],4 ,17,17,30);

numbWall = 0;

numbWall = HorizontalLineWall(numbWall,14, 81, 60);

numbWall = VerticalLineWall(numbWall, 4, 120, 180);

numbWall = VerticalLineWall(numbWall, 4, 120, 480);

numbWall = HorizontalLineWall(numbWall, 3, 240, 180);

numbWall = VerticalLineWall(numbWall, 3, 344, 232);

numbWall = HorizontalLineWall(numbWall, 3, 240, 636);

numbWall = VerticalLineWall(numbWall, 3, 344, 480);

numbWall = HorizontalLineWall(numbWall, 3, 480, 180);

numbWall = VerticalLineWall(numbWall, 3, 480, 232);

numbWall = HorizontalLineWall(numbWall, 3, 480, 636);

numbWall = VerticalLineWall(numbWall, 3, 480, 480);

numbWall = VerticalLineWall(numbWall, 4, 720, 180);

numbWall = VerticalLineWall(numbWall, 4, 720, 480);

numbWall = HorizontalLineWall(numbWall, 14, 81, 780);

}

void FinalizeWall() {

DeleteObject(hbrWall);

DeleteObject(hpNoBorder);

DeleteObject(hpBorder);

}

void DrawWall(HDC hdc) {

SaveDC(hdc);

SelectObject(hdc, hbrWall);

Rectangle(hdc, ZONE\_LEFT, ZONE\_TOP, ZONE\_RIGHT + 60, ZONE\_TOP + 5);

Rectangle(hdc, ZONE\_RIGHT + 55, ZONE\_TOP, ZONE\_RIGHT + 60,

ZONE\_BOTTOM + 60);

Rectangle(hdc, ZONE\_LEFT, ZONE\_BOTTOM + 55, ZONE\_RIGHT + 60,

ZONE\_BOTTOM + 60);

Rectangle(hdc, ZONE\_LEFT, ZONE\_TOP, ZONE\_LEFT + 5, ZONE\_BOTTOM + 60);

int x, y, i, j, n;

for (i = 0; i < numbWall; i++) {

y = 0;

SelectObject(hdc, hpBorder);

for (j = 0; j < 2; j++) {

x = 0;

for (n = 0; n < 2; n++) {

Rectangle(hdc, wall[i].x + base\_wall[0].left + x,

wall[i].y + base\_wall[0].top + y,

wall[i].x + base\_wall[0].right + x,

wall[i].y + base\_wall[0].bottom + y);

x = 26;

}

x = 13;

y += 13;

Rectangle(hdc, wall[i].x + base\_wall[0].left + x,

wall[i].y + base\_wall[0].top + y, wall[i].x +

base\_wall[0].right + x, wall[i].y +

base\_wall[0].bottom + y);

y += 13;

}

y = 0;

SelectObject(hdc, hpNoBorder);

for (j = 0; j < 2; j++) {

x = 0;

for (n = 0; n < 2; n++) {

Rectangle(hdc, wall[i].x + base\_wall[1].left + x,

wall[i].y + base\_wall[1].top + y,

wall[i].x + base\_wall[1].right + x,

wall[i].y + base\_wall[1].bottom + y);

x = 39;

}

y += 26;

}

}

RestoreDC(hdc, -1);

}