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

к курсовому проекту

на тему:

Игровое приложение «Xonix»

БГУИР КП 1-40 01 01 1 ПЗ

Студент: гр. 751003 Гринчик В.В.

Проверил: Базылев Е.Н.

Минск 2019

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

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

(подпись)

Лапицкая Н.В. 2019г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Гринчику Всеволоду Владимировичу* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема работы *Игровое приложение «Xonix»* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

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

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

2. Срок сдачи законченной работы *30.11.2019г.*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Исходные данные к работе *Язык программирования C++. Среда разработки Microsoft Visual Studio 2019. Реализация игрового приложения “Xonix” с использованием набора базовых функции интерфейсов программирования приложений операционных систем семейства Microsoft Windows(Windows API). Реализовать графический пользовательский интерфейс. Реализовать управление процессом игры с использованием функций обработки аппаратных сообщений интерфейса Windows API.*\_\_\_\_\_\_\_\_\_\_\_

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

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

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

*Введение*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*1 Анализ литературных источников\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2 Постановка задачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*3 Разработка программного средства\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*4 Руководство по установке и 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 *Базылев Е.Н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

7.Дата выдачи задания *04.09.2019г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

8. Календарный график работы над проектом на весь период проектирования

(с обозначением сроков выполнения и процентом от общего обьема работы):

*Раздел 1. Введение к 20.09.2018г. – 10 % готовности работы;\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 2 к 15.10.2018г. – 30% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 3 к 01.11.2018г. – 60% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 4 к 05.11.2018г. – 80% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Заключениe к 10.11.2018г. – 90% готовности работы; \_\_\_\_\_ оформление\_ пояснительной записки и графического материала к 15.11.2018г. –100%\_готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Защита курсового проекта с 15 .11.2018г. по 30.11.2018г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

РУКОВОДИТЕЛЬ *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* *Базылев Е.Н.*

*(подпись)*

Задание принял к исполнению *Гринчик В.В. 04.09.2019г.*

*(дата и подпись студента)*

Содержание

[Введение 5](#_Toc25619783)

[1 Анализ литературных источников 6](#_Toc25619784)

[1.1 Анализ существующих приложений 6](#_Toc25619785)

[2 моделирование предметной области 9](#_Toc25619786)

[2.1 Цель 9](#_Toc25619787)

[2.2 Задачи 9](#_Toc25619788)

[2.3 Обоснование выбора языка программирования 9](#_Toc25619789)

[3 Разработка программного средства 10](#_Toc25619790)

[3.1 Визуальное представление приложения 10](#_Toc25619791)

[3. 2 Схема алгоритмов решения задачи по ГОСТ 19.701-90 13](#_Toc25619792)

[4 Руководство по установке и использованию программного средства 21](#_Toc25619793)

[4.1 Начало работы 21](#_Toc25619794)

[Заключение 22](#_Toc25619795)

[Список использованной литературы 23](#_Toc25619796)

[5 ПРИЛОЖЕНИЕ 24](#_Toc25619797)

Введение

Компьютерная игра — компьютерная программа, служащая для организации игрового процесса (геймплея), обучения, ускорения психологического развития, а также для создания неформальных или же на оборот особо формальных(ролеплей) связей с партнёрами по игре.

В наше время сложно найти человека, который бы ни разу не играл в компьютерные игры, и с каждым днём становится всё больше так называемых геймеров - людей которые постоянно играют в игры, и даже зарабатывают на этом деньги. Будь то новейшая игра на современной игровой приставке или старая аркада на игровом автомате или эмуляторе. По некоторым играм, относящиеся к так называемой не стареющей классике проводятся ежегодные турниры. В популярные игры может одновременно играть более миллиона человек. Однако чаще всего подобные турниры ассоциируются с такими играми, как Pac-Man, Tetris, Pong. Но существует огромное число игр которые хотя и не имеют столь громких имен, однако также представлены на различных турнирах это BreakOut, Qix, Xonix, и т. д..

Однако чтобы вспомнить игры из детства вовсе не обязательно идти в зал игровых автоматов или устанавливать эмулятор. Многие игры старые игры воссоздаются на современных платформах и не смотря на свою “старость” зачастую пользуются успехом. И именно на основание утверждений описанных выше я принял решение поучаствовать в процессе воссоздания незаслуженно забытых игр разработать версию игры “Xonix” которая бы работала на современных представителях операционных систем семейства Microsoft Windows .

# Анализ литературных источников

## Анализ существующих приложений

Итак, первоначально игра Xonix была создана в 1984 году для платформы [PC](https://ru.wikipedia.org/wiki/IBM_PC-%D1%81%D0%BE%D0%B2%D0%BC%D0%B5%D1%81%D1%82%D0%B8%D0%BC%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) как клон игры [Qix](https://ru.wikipedia.org/wiki/Qix), появившейся ранее на [аркадных автоматах](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%BA%D0%B0%D0%B4%D0%BD%D1%8B%D0%B9_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82). Разработана Иланом Рабом и Дэни Катцем.

Игровое поле представляет собой сетку из квадратных или прямоугольных ячеек, поэтому игра легко реализуется в текстовом режиме экрана. Ячейки могут быть двух типов: условно «суша» и «море». По полю движутся управляемый игроком курсор и управляемые программой точки. Курсор может двигаться по вертикали и горизонтали, точки — во всех направлениях.

Курсор может свободно передвигаться по «суше. Выходя в «море», он оставляет за собой след, уязвимый для «морских» точек, при этом в «море» нельзя изменить направление движения курсора на противоположное, можно поворачивать только на 90 градусов в любую сторону, также в «море» нельзя останавливаться. Игра считается проигранной, если курсор точка или «морская» точка, пересечет след курсора, а также при попытке в «море» изменить направление движения на противоположное или остановиться. Как только курсор снова оказывается на «суше», след его превращается в новую «сушу». Если при этом в «море» появилась замкнутая область, не содержащая точек, то вся эта область также превращается в «сушу».

Проще говоря, игрок, управляя курсором, пытается отсечь куски «моря», превращая их в «сушу», а точки ему в этом мешают.

К современным аналогам можно отнести такие игры как:

**1.1.1 AirXonix** – популярный 3D ремейк Xonix.

Рисунок 1.1.1– Внешний вид AirXonix

Цель игрока в AirXonix - используя некое летающее устройство "заполнить" как можно большую часть игрового поля, избегая столкновения с находящимися на нем шарами.

**Плюсы:**

3D графика;

Размер программы;

Несколько уровней сложности.

**Минусы:**

* Старость (игра не обновлялась с 1999 года);
* Множество отступлений от классической версии Xonix;
* AirXonix – распространяется на коммерческой основе.

В целом, я считаю, что это лучшая модификация, созданная на основе каноничной игры, но из-за большого числа изменений её уже нельзя назвать тем самым Xonix.

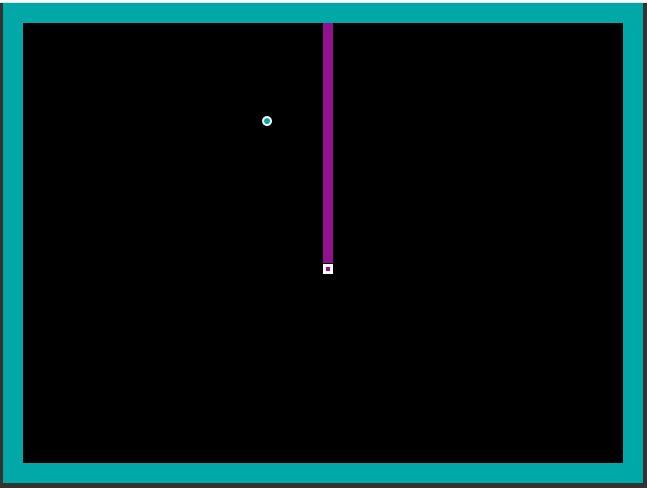
**1.1.2 xonix.pixel-tyumen** – онлайн версия Xonix.

Рисунок 1.1.2 – Внешний вид xonix.pixel-tyumen

Является полностью классической версией Xonix перенесенной на JavaScript.

**Плюсы:**

Доступность;

каноничность.

**Минусы:**

* Необходимость подключения к интернету;

Наиболее удачный каноничный ремейк Xonix. Главным недостатком которого я считаю отсутствие возможность скачивания приложения.

# моделирование предметной области

## Цель

Целью данного курсового проекта является разработка игрового программного средства «Xonix».

В процессе реализации программного средства планируется разработать игру с качественным графическим отображением процесса игры и корректной передачей команд игрока реализованных через интерфейс операционной системы Windows.

## Задачи

На основе цели работы было решено уделить особое внимание корректности передачи данных и внутренней логике программного средства.

Назначение игрового приложения заключается в возможности использования его как развлечения игроками разного уровня в связи, с чем пользователю доступны следующие возможности:

* возможность изменения скорости противников;
* возможность изменения числа противников.

## Обоснование выбора языка программирования

Для разработки программного средства использован язык программирования C++. Среда для разработки – Visual Studio 2019. Использование данной среды разработки дает возможность создания консольных и оконных приложений Windows. Выбор языка связан со следующими его преимуществами:

* язык программирования C++ претендует на объектную ориентированность
* возможность использования библиотеки windows;
* возможность взаимодействия с Windows API;
* унифицированная система типизации.

# Разработка программного средства

## Визуальное представление приложения

Главный вид программы имеет вид, представленный на рисунке 3.1.1 в нижней части приведен процент поля, захваченный игроком и максимальная скорость соперников.

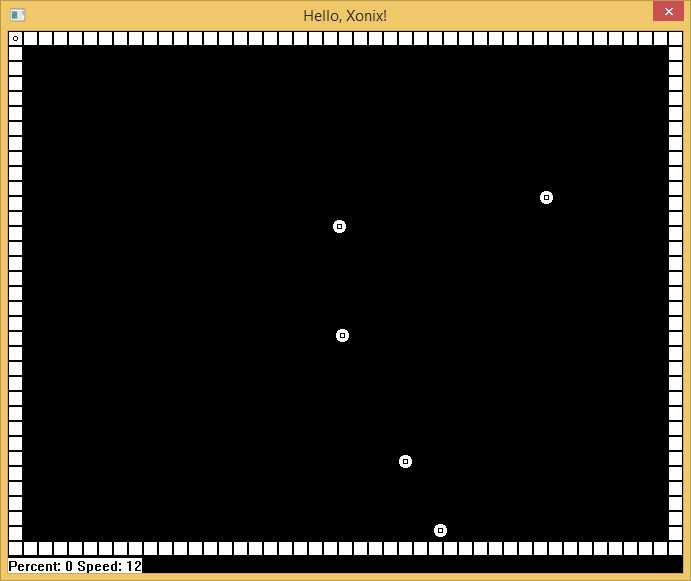


Рисунок 3.1. 1– Главное окно

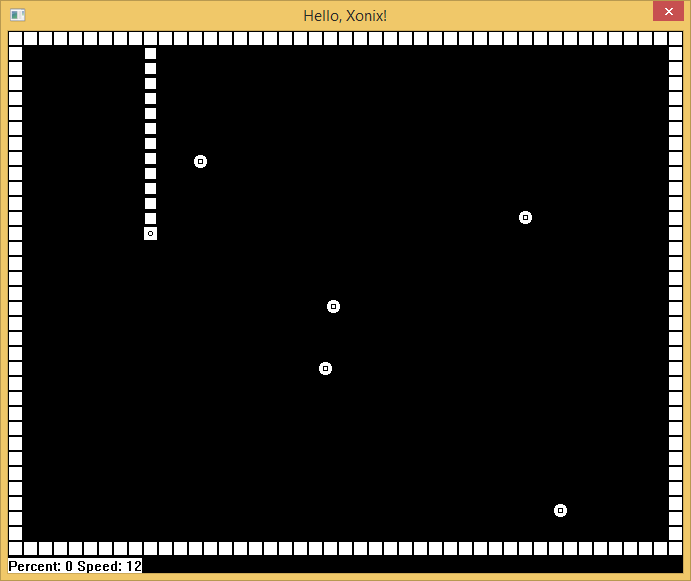


Рисунок 3.1.2– Движения игрока

При нажатии одной из клавиш управления курсора игровой объект начнет двигаться рис 3.1.2.

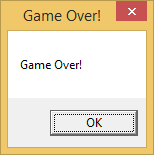


Рисунок 3.1.3– Сообщение о поражении

При касании одним из противников линии движения игрока появится окно с сообщением об ошибке рис 3.1.3.

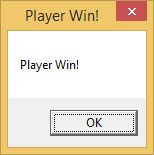


Рисунок 3.1.4– Сообщение о победе

При захвате более чем 70% поля появляется окно с сообщением о победе рис 3.1.4.

## 3. 2 Схема алгоритмов решения задачи по ГОСТ 19.701-90



Рисунок 3.2.1– Схема алгоритма функции движения курсора(часть1)



Рисунок 3.2.2– Схема алгоритма функции движения курсора(часть2)



Рисунок 3.2.3– Схема алгоритма функции движения курсора(часть3)



Рисунок 3.2.4– Схема алгоритма определения территории противника



Рисунок 3.2.5– Схема алгоритма обновления игрового поля



Рисунок 3.2.6– Схема алгоритма движения противников



Рисунок 3.2.7– Схема алгоритма прорисовки(часть1)



Рисунок 3.2.8– Схема алгоритма прорисовки(часть2)

# Руководство по установке и использованию программного средства

4.1 Начало работы

Для запуска игры необходимо открыть файлы Xonix.exe. После чего появляется окно игры, представленное на рисунке 5.1.

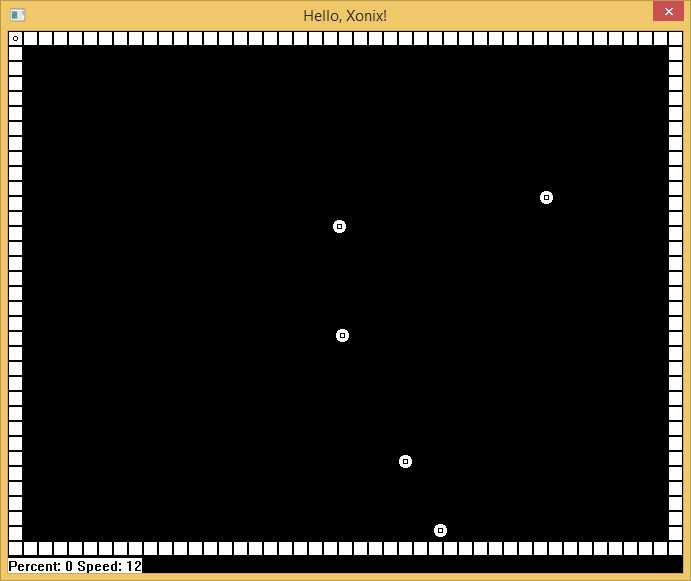


Рисунок 4.1.1 – окно игры

Управление курсором производится с помощью стрелок. Для увеличения, уменьшения максимальной скорости соперников используются клавиши \*, / соответственно. Для увеличения, уменьшения числа соперников используются клавиши +, - соответственно. Для перезапуска игры используется клавиша “end”. Для победы необходимо захватить 70% поля.

Заключение

В ходе разработки приложения был создан продукт, предоставляющий возможность играть в Xonix на различных уровнях скорости и с различным числом врагов.

При тестировании приложения не было выявлено случаев некорректной работы программы и появления ошибок.

Написанный код использую функций Windows API как для вывода на экран, так и для определения действий пользователя.

Итогами данной работы является опыт разработки приложений на языке программирования C++, изучение и применение основных функций Widwos API, а также опыт работы с много-поточностью и создания аналитических функций.

Список использованной литературы

[1] Рихтер Джеффри, Windows для профессионалов /Роберт Лафоре. – Санкт-Петербург: Классика Computer Science, 2001, 764 с

[2] Никита Культин Искусство программирования: в 4 т. /Никита Культин – Санкт-Петербург: БХВ-Петербург, 2009, 368 с.

[3] [Microsoft Developer Network](https://msdn.microsoft.com/ru-RU/)> Примеры [Электронный ресурс] – Режим доступа: https://code.msdn.microsoft.com. – Дата доступа: 24.11.2019.

[5] Блох Д.: Эффективное программирование, 2001– 464 с.

[6] ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – Введ. 01.01.1992. – М.: Изд-во стандартов, 1991.

[7] ГОСТ 7. 53–2001. Издания. Международная стандартная нумерация книг [Текст]. – Взамен ГОСТ 7.53–86 ; введ. 2002–07–01. – Минск: Межгос. совет по стандартизации, метрологии и сертификации ; М.: Изд-во стандартов, cop. 2002. – 3 с. – 05.10.2018

# ПРИЛОЖЕНИЕ

#include <windows.h>

#include <wingdi.h>

typedef struct GameObject

{

int delta\_x;

int delta\_y;

int x;

int y;

} GameObject;

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);

void MyRedrawWindow(HWND hWnd);

void InitializeGame(GameObject\* player, GameObject\* enemy, int start\_x, int start\_y, int edge);

void InitializeEnemy(GameObject\* objects, int edge);

float MovePlayer(GameObject\* object, GameObject\* enemy, float enemyPercent);

void MoveEnemy(GameObject\* objects);

void DrawGameField(GameObject\* object, GameObject\* enemy, float enemyPercent, int enemyMaxSpeed, HDC hdc);

float UpdateField();

void CheckCell(int y, int x);

//константы приложения

#define PercentMessage "Percent: "

#define SpeedMessage "Speed: "

#define OverMessage "Game Over!"

#define WinMessage "Player Win!"

#define CELL\_SIZE 15

#define Height 35

#define Width 45

#define WIN\_PERCENT 70

#define STATISTIC\_PIXEL 20

const int MaxCountEnemy = Width / 2;

const int MaxStackSize = Width \* Height \* 2 + 1;

const int Edge = 1;

//размеры окна в пикселях

const int pixelWidth = 16 + (Width) \* CELL\_SIZE;

const int pixelHeight = 16 +(Height) \* CELL\_SIZE + STATISTIC\_PIXEL\*2;

//флаги

bool GameEnd = false;

bool PlayerWin = false;

//изменяемые параметры

int CountEnemy =5;

int EnemyMaxSpeed = (12 % (CELL\_SIZE));

class stack

{

int obj[MaxStackSize];

public:

int top;

stack()

{

top = 0;

}

void push(int i)

{

obj[top] = i;

top++;

}

int pop()

{

return(obj[--top]);

}

};

//Глобальные переменные

float EnemyPercent = 100;

int gameField[Height][Width];

GameObject\* Enemys = new GameObject[MaxCountEnemy];

GameObject Player;

int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPTSTR lpCmdLine, int nCmdShow)

{

WNDCLASSEX wcex; HWND hWnd; MSG msg;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = NULL;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = NULL;

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

wcex.hbrBackground = NULL;

wcex.lpszMenuName = NULL;

wcex.lpszClassName = "XonixClass";

wcex.hIconSm = wcex.hIcon;

RegisterClassEx(&wcex);

hWnd = CreateWindow("XonixClass", "Hello, Xonix!",

WS\_OVERLAPPED | WS\_SYSMENU, CW\_USEDEFAULT, CW\_USEDEFAULT, pixelWidth,

pixelHeight, NULL, NULL, hInstance, NULL);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return (int)msg.wParam;

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message,

WPARAM wParam, LPARAM lParam)

{

PAINTSTRUCT ps;

HDC hdc; //создаём контекст устройства

switch (message)

{

case WM\_CREATE:

{

// инициализация игры

InitializeGame(&Player, Enemys, 0, 0, Edge);

SetTimer(hWnd, 1, 1000/15, NULL);

return 0;

}

case WM\_TIMER:

{

if (!GameEnd) {

EnemyPercent = MovePlayer(&Player,

Enemys, EnemyPercent);

MoveEnemy(Enemys);

}

else

{

GameEnd = false;

if (PlayerWin)

{

PlayerWin = false;

MessageBox(hWnd, WinMessage,

WinMessage, MB\_OK | MB\_APPLMODAL);

EnemyPercent = 100;

InitializeGame(&Player, Enemys,

0, 0, Edge);

}

else {

EnemyPercent = 100;

InitializeGame(&Player, Enemys,

0, 0, Edge);

MessageBox(hWnd, OverMessage, OverMessage, MB\_OK | MB\_APPLMODAL);

}

}

MyRedrawWindow(hWnd);

break;

}

case WM\_KEYDOWN:

{

switch (wParam)

{

case VK\_LEFT:

Player.delta\_x = -1;

Player.delta\_y = 0;

break;

case VK\_UP:

Player.delta\_x = 0;

Player.delta\_y = -1;

break;

case VK\_RIGHT:

Player.delta\_x = 1;

Player.delta\_y = 0;

break;

case VK\_DOWN:

Player.delta\_x = 0;

Player.delta\_y = 1;

break;

default:

break;

}

MyRedrawWindow(hWnd);

break;

}

case WM\_KEYUP:

{

switch (wParam)

{

case VK\_ADD:

if (CountEnemy < MaxCountEnemy)

{

CountEnemy++;

InitializeGame(&Player, Enemys, 0, 0, 1);

}

break;

case VK\_SUBTRACT:

if (CountEnemy > 0)

{

CountEnemy--;

InitializeGame(&Player, Enemys, 0, 0, 1);

}

break;

case VK\_MULTIPLY:

if (EnemyMaxSpeed < CELL\_SIZE)

{

EnemyMaxSpeed++;

}

break;

case VK\_DIVIDE:

if (EnemyMaxSpeed > 1)

{

EnemyMaxSpeed--;

}

break;

case VK\_END:

InitializeGame(&Player, Enemys, 0, 0, 1);

break;

default:

break;

}

}

case WM\_ERASEBKGND:

return 1;

case WM\_PAINT:

{

hdc = BeginPaint(hWnd, &ps);

ps.fErase = false;

DrawGameField(&Player, Enemys,

EnemyPercent, EnemyMaxSpeed, hdc);

EndPaint(hWnd, &ps);

break;

}

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message,

wParam, lParam);

}

return 0;

}

void MyRedrawWindow(HWND hWnd)

{

InvalidateRect(hWnd, NULL, TRUE);

UpdateWindow(hWnd);

}

void InitializeGame(GameObject\* player, GameObject\* enemy,

int start\_x, int start\_y, int edge)

{

//инициализация края

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

for (int j = 0; j < Width; j++)

if (i < edge || j < edge || i >= Height - edge

|| j >= Width - edge) gameField[i][j] = 1;

else gameField[i][j] = 0;

//Инициализация игрока

player->x = start\_x;

player->y = start\_y;

player->delta\_x = 0;

player->delta\_y = 0;

InitializeEnemy(enemy, edge);

}

void InitializeEnemy(GameObject\* objects, int edge)

{

//Инициализация противников

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

{

objects[i].x = (edge + rand() %

(Width - edge \* 2))\* CELL\_SIZE;

objects[i].y = (edge + rand() %

(Height - edge \* 2))\* CELL\_SIZE;

objects[i].delta\_x = EnemyMaxSpeed - rand() % (EnemyMaxSpeed \* 2 + 1);

objects[i].delta\_y = EnemyMaxSpeed - rand() % (EnemyMaxSpeed \* 2 + 1);

}

}

float MovePlayer(GameObject\* object, GameObject\* enemy, float enemyPercent)

{

if ((object->delta\_x != 0) || (object->delta\_y != 0)) {

object->x += object->delta\_x;

if ((object->x) < 0)

{

object->x = 0;

}

else if ((object->x) >= Width)

{

object->x = Width - 1;

}

object->y += object->delta\_y;

if ((object->y) < 0)

{

object->y = 0;

}

else if ((object->y) >= Height)

{

object->y = Height - 1;

}

//проверка конечной точки

switch (gameField[object->y][object->x])

{

case 0:

{

gameField[object->y][object->x] = 2;

break;

}

case 1:

{

//проверяем наличие "хвоста"

if (gameField[object->y- object->delta\_y] [object->x - object->delta\_x] == 2) {

object->delta\_x = 0;

object->delta\_y = 0;

//определение вражеской територии

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

CheckCell(enemy[i].x / CELL\_SIZE, enemy[i].y / CELL\_SIZE);

enemyPercent = UpdateField();

//проверка на победу игрока

if (enemyPercent < 100 - WIN\_PERCENT)

{

GameEnd = true;

PlayerWin = true;

}

}

break;

}

case 2:

{

GameEnd = true;

break;

}

}

}

return enemyPercent;

}

void CheckCell(int x, int y)

{

stack CordinateStack;

CordinateStack.push(x);

CordinateStack.push(y);

while (CordinateStack.top != 0) {

if (gameField[y][x] == 0) gameField[y][x] = -1;

if (gameField[y - 1][x] == 0)

{

CordinateStack.push(x);

CordinateStack.push(y);

y--;

}

else if (gameField[y + 1][x] == 0)

{

CordinateStack.push(x);

CordinateStack.push(y);

y++;

}

else if (gameField[y][x - 1] == 0)

{

CordinateStack.push(x);

CordinateStack.push(y);

x--;

}

else if (gameField[y][x + 1] == 0)

{

CordinateStack.push(x);

CordinateStack.push(y);

x++;

}

else

{

y = CordinateStack.pop();

x = CordinateStack.pop();

}

}

}

float UpdateField()

{

int ColEnemyCell= 0;

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

for (int j = 0; j < Width; j++)

if (gameField[i][j] == -1)

{

gameField[i][j] = 0;

ColEnemyCell++;

}

else gameField[i][j] = 1;

//процент вражеских територии

return ((float)ColEnemyCell / ((Width-1) \* (Height-1)))\*100;

}

void MoveEnemy(GameObject\* objects)

{

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

{

objects[i].x += objects[i].delta\_x;

if ((gameField[objects[i].y / CELL\_SIZE]

[objects[i].x / CELL\_SIZE] == 1)||

(gameField[objects[i].y / CELL\_SIZE]

[objects[i].x / CELL\_SIZE+1] == 1))

{

objects[i].delta\_x = -objects[i].delta\_x;

objects[i].x += objects[i].delta\_x;

}

objects[i].y += objects[i].delta\_y;

if ((gameField[objects[i].y / CELL\_SIZE]

[objects[i].x / CELL\_SIZE] == 1)

|| (gameField[objects[i].y / CELL\_SIZE + 1] [objects[i].x / CELL\_SIZE ] == 1))

{

objects[i].delta\_y = -objects[i].delta\_y;

objects[i].y += objects[i].delta\_y;

}

//проверка конечной точки

if (gameField[objects[i].y / CELL\_SIZE] [objects[i].x / CELL\_SIZE] == 2) {

GameEnd = true;

}

}

}

void DrawGameField(GameObject\* object, GameObject\* enemy, float enemyPercent, int enemyMaxSpeed, HDC hdc)

{

//буферизация

HDC BuffHdc = CreateCompatibleDC(hdc);

HBITMAP BuffHbm = CreateCompatibleBitmap(hdc, pixelWidth, pixelHeight);

HANDLE BuffHan = SelectObject(BuffHdc, BuffHbm);

//перерисовка игрового поля

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

{

for (int j = 0; j < Width; j++) {

if (gameField[i][j] == 1)

{

Rectangle(BuffHdc, j \* CELL\_SIZE, i \* CELL\_SIZE, j \* CELL\_SIZE + CELL\_SIZE, i \* CELL\_SIZE + CELL\_SIZE);

}

if (gameField[i][j] == 2)

{

Rectangle(BuffHdc, j \* CELL\_SIZE, i \* CELL\_SIZE, j \* CELL\_SIZE + CELL\_SIZE, i \* CELL\_SIZE + CELL\_SIZE);

Rectangle(BuffHdc, j \* CELL\_SIZE + CELL\_SIZE/10, i \* CELL\_SIZE + CELL\_SIZE / 10, j \* CELL\_SIZE + CELL\_SIZE - CELL\_SIZE / 10, i \* CELL\_SIZE + CELL\_SIZE - CELL\_SIZE / 10);

}

}

}

//Draw Player

Rectangle(BuffHdc, object->x \* CELL\_SIZE, object->y \* CELL\_SIZE, object->x \* CELL\_SIZE + CELL\_SIZE, object->y \* CELL\_SIZE + CELL\_SIZE);

Ellipse(BuffHdc, object->x \* CELL\_SIZE+ CELL\_SIZE / 3, object->y \* CELL\_SIZE + CELL\_SIZE / 3, object->x \* CELL\_SIZE + CELL\_SIZE - CELL\_SIZE / 3, object->y \* CELL\_SIZE + CELL\_SIZE - CELL\_SIZE / 3);

//Перерисовка врагов

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

{

Ellipse(BuffHdc, enemy[i].x , enemy[i].y ,

enemy[i].x + CELL\_SIZE, enemy[i].y + CELL\_SIZE);

Rectangle(BuffHdc, enemy[i].x + CELL\_SIZE / 3, enemy[i].y + CELL\_SIZE / 3, enemy[i].x + CELL\_SIZE - CELL\_SIZE / 3, enemy[i].y + CELL\_SIZE - CELL\_SIZE / 3);

}

//Вывод статистики

char\* stat = new char(3);

char message [100] = PercentMessage;

\_itoa(100 - enemyPercent, stat, 10);

strcat(message, stat);

strcat(message, " ");

strcat(message, SpeedMessage);

\_itoa(enemyMaxSpeed, stat, 10);

strcat(message, stat);

TextOut(BuffHdc, 0, (Height)\* CELL\_SIZE +2 , message, strlen(message));

BitBlt(hdc, 0, 0, pixelWidth, pixelHeight, BuffHdc, 0, 0, SRCCOPY);

//освобождаем память

SelectObject(BuffHdc, BuffHan);

DeleteObject(BuffHbm);

DeleteObject(BuffHdc);

}