Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: компьютерные системы и сети (КСиС)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

Игровое приложение «Линии»

БГУИР КП 1-40 01 01 209 ПЗ

Студент: гр. 551004 Ермошин М.А.

Руководитель: асс. Базылев Е.Н.

Минск 2017

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc499065647)

[Версии 7](#_Toc499065648)

[1. Анализ задания и постановка задачи 8](#_Toc499065649)

[2. Теоретическая часть 10](#_Toc499065650)

[3. Проектирование программы 14](#_Toc499065651)

[3.1 Пользовательские функции 14](#_Toc499065652)

[3.2 Хранение данных 15](#_Toc499065653)

[3.3 Воспроизведение анимации 15](#_Toc499065654)

[3.4 Подсчёт проведённого в игре времени 15](#_Toc499065655)

[3.5 Начисление очков 16](#_Toc499065656)

[4. Реализация программы 17](#_Toc499065657)

[4.1 Реализация главного окна игры 17](#_Toc499065658)

[4.2 Реализация перемещения шариков 18](#_Toc499065659)

[4.3 Реализация новой игры 19](#_Toc499065660)

[4.4 Составление шаров в линию 20](#_Toc499065661)

[4.5 Реализация сохранения и загрузки игры 20](#_Toc499065662)

[5. Тестирование 21](#_Toc499065663)

[6. Руководство пользователя 22](#_Toc499065664)

[ЗАКЛЮЧЕНИЕ 23](#_Toc499065665)

[СПИСОК ЛИТЕРАТУРЫ 24](#_Toc499065666)

[Листинг программы 25](#_Toc499065667)

[5.0.cpp 25](#_Toc499065668)

[5.0.h 41](#_Toc499065669)

[PathFinder.cpp 42](#_Toc499065670)

[PathFinder.h 48](#_Toc499065671)

[Ведомость 49](#_Toc499065672)

# ВВЕДЕНИЕ

Стремительное развитие и широкое распространение вычислительной техники послужили предпосылками к появлению нового раздела науки, названного информатикой. Сегодня информатика не только научная и учебная дисциплина. Можно с полным основанием утверждать, что она превратилась в динамично развивающуюся отрасль народного хозяйства. Всё больше расширяется сфера применения компьютеров. ЭВМ применяются теперь не только в таких традиционных областях как кибернетика, финансы, экономика. Сегодня компьютеры приходят даже в те сферы, где главную роль всегда играл человек: здравоохранение, культура, образование.

Данный курсовой проект предусматривает написание простейшего windows-приложения, используя набор базовых функций winapi.

**Windows API** (англ. *application programming interfaces*) — общее наименование целого набора базовых функций интерфейсов программирования приложений операционных систем семейств Windows и Windows NT корпорации «Майкрософт». Является самым прямым способом взаимодействия приложений с Windows. Для создания программ, использующих Windows API, «Майкрософт» выпускает SDK, который называется Platform SDK и содержит документацию, набор библиотек, утилит и других инструментальных средств.

Windows API был изначально спроектирован для использования в программах, написанных на языке C (или C++). Работа через Windows API — это наиболее близкий к системе способ взаимодействия с ней из прикладных программ. Более низкий уровень доступа, необходимый только для драйверов устройств, в текущих версиях Windows предоставляется через Windows Driver Model.

# Версии

* **Win16** — первая версия Windows API для 16-разрядных версий Windows. Изначально назывался просто Windows API, затем стал называться Win16 для отличия от Win32.
* **Win32s** — подмножество Win32, устанавливаемое на семейство 16-разрядных систем Windows 3.x и реализующее ограниченный набор функций Win32 API для этих систем.
* **Win32** — 32-разрядный API для современных версий Windows. Самая популярная ныне версия. Базовые функции этого API реализованы в DLL **kernel32.dll** и **advapi32.dll**; базовые модули GUI — в **user32.dll** и **gdi32.dll**. Win32 появился вместе с Windows NT и затем был перенесён (в несколько ограниченном виде) в системы серии Windows 9x. В современных версиях Windows, происходящих от Windows NT, работу Win32 GUI обеспечивают два модуля: csrss.exe (*Client/Server Runtime Subsystem*), работающий в пользовательском режиме, и win32k.sys в режиме ядра. Работу же системных Win32 API обеспечивает ядро — ntoskrnl.exe

**Win64** — 64-разрядная версия Win32, содержащая дополнительные функции для использования на 64-разрядных компьютерах. Win64 API можно найти только в 64-разрядных версиях Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows Server 2008 R2 и Windows 7 и т.д.

# 1. Анализ задания и постановка задачи

В нашем случае предметной областью является мини-игра «LINES». Игра существует уже давно, поэтому ничего нового изобретать нам не придётся, нужно лишь реализовать приложение.

Правила игры предельно просты. Имеется игровое поле – 9х9 ячеек в которых лежат шарики. Необходимо переставлять шарики одного цвета в линии из 5 и более по горизонтали или вертикали. В начале шариков мало, но в течение игры их количество увеличивается.

Смысл игры заключается в том, что необходимо набрать как можно больше очков, из одноцветных шаров фигуры.

Шары перемещаются по полю мышью: cначала выбирается шар, который надо переместить, затем – пустую позицию, на которую вы бы хотели его поставить. Если вместо пустой позиции вы выберете занятую, то шар стоящий на ней станет выбранным.

Шар не всегда может попасть туда куда вы бы его хотели поставить. После каждого перемещения появляются ещё три шара.

Игра не была бы достаточно интересной, не будь в ней реализован подсчёт игрового времени, анимации и звукового сопровождения, и возможности сохранения с последующей загрузкой.

Чтобы не загромождать игровое поле, очков и игрового времени вынесен на специальную строку состояния, которая будет располагаться за пределами игрового поля, вверху окна.

Следовательно, можно сделать вывод, что в нашей программе мы будем реализовывать:

1. Игровое поле 9\*9 со случайной генерацией чисел
2. Перемещение шаров по полю
3. Уничтожение шаров
4. Анимация
5. Звуковое сопровождение
6. Подсчёт игрового времени
7. Запись текущего состояния в файл
8. Загрузка сохранённой игры из файла

# 2. Теоретическая часть

В данной части пояснительной записки хочется рассказать о стандартных процедурах и функциях, которые использовались в моей программе.

1)MyRegisterClass – функция регистрации класса окна. Является одной из основных функций. В ней мы задаём стили, иконки, стиль курсора, цвет фона и многое другое. В нашей программе полностью оформленная эта функция выглядит так:

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEX 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 = (HICON) LoadImage(hInstance, MAKEINTRESOURCE(IDI\_MY50), IMAGE\_ICON, 48, 48, LR\_DEFAULTCOLOR);

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

wcex.hbrBackground = nullptr;

wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_MY50); wcex.lpszClassName = szWindowClass;

wcex.hIconSm = (HICON)LoadImage(hInstance, MAKEINTRESOURCE(IDI\_MY50), IMAGE\_ICON, 16, 16, LR\_DEFAULTCOLOR);

return RegisterClassEx(&wcex);

}

2) CreateWindow – функция создания окна. В ней мы указываем заголовок окна, его стили, размеры и др. Без использования этой функции наша программа в принципе невозможна. В нашем проекте эта функция выглядит следующим образом:

hWnd = CreateWindow(szWindowClass, szTitle, WS\_CAPTION | WS\_OVERLAPPED | WS\_MINIMIZEBOX | WS\_SYSMENU,

CW\_USEDEFAULT, 0, 426, 520, NULL, NULL, hInstance, NULL);

где hWnd – указатель на дескриптор создаваемого нами окна.

Ещё некоторые функции окна:

ShowWindow – функция, которая в зависимости от переданых ей параметров либо показывает, либо скрывает окно.

UpdateWindow – функция обновления окна.

3) PlaySound (MY\_SOUND1,NULL, SND\_FILENAME | SND\_ASYNC) – функция для воспроизведения звуковых файлов в формате WAV.

Первый параметр функции задает либо имя файла на диске, либо строку имени звукового ресурса. Если при вызове PlaySound первый параметр равен NULL, это вызывает прекращение воспроизведения любого «играемого» файла. Второй параметр используется только при воспроизведении звука из звукового ресурса. Если же функция вызвана для воспроизведения звука из файла, то второй параметр устанавливается в NULL. Третий параметр управляет загрузкой и воспроизведением звука.

4) SetTimer – функция присоединения стандартного таймера Windows. Первый параметр, hWnd, содержит дескриптор окна, ассоциированного с данным таймером. Второй параметр, nIDEvent, позволяет указывать идентификатор таймера, которым может быть произвольное целое число (но не нуль). Третий параметр, uElapse, задает интервал, который может находиться в пределах (теоретически) от 1 до 4 294 967 295 мс, что составляет около 50 дней. Это значение определяет темп, с которым Windows будет посылать программе сообщения WM\_TIMER. Сообщения WM\_TIMER направляются либо оконной процедуре для окна hWnd, если параметр IpTimeProc равен NULL, либо функции обратного вызова с адресом IpTimeProc — в противном случае.

5) ImageList\_Add – функция добавления изображений в список. Первый аргумент этой функции: хэндл списка изображений. Второй аргумент представляет собой хэндл добавляемого в список изображения. Третий аргумент – хэндл монохромного bitmap’a, который содержит маски.

ImageList\_Draw – функция прорисовки изображения хранящегося в списке. Первый аргумент хэндл списка изображений, второй – индекс изображения, третий – хэндл контекста, на которых будет копироваться изображение, четвёртый и пятый – координаты в этом контексте, начиная с которых будет скопировано изображение, и, наконец, шестой – именно флаги прорисовки.

6) GetObject – функция получающая размеры изображения. Первый параметр дескриптор графического объекта. Второй параметр размер буфера для информации об объекте. Третий параметр адрес буфера для информации об объекте. Функция является универсальной, поэтому в параметре hgdiobj может быть указан дескриптор одного из таких объектов, как перо, кисть, шрифт, палитра или растр типа Н BITMAP.

7) BitBlt – Функция BitBlt передает прямоугольный блок пикселов с поверхности устройства-источника hdcSrc на поверхность устройства-приемника hdcDest. Исходный прямоугольник определяется параметрами nXSrc, nYSrc, nWidth, nHeight в логической системе координат контекста устройства-источника. Приемный прямоугольник определяется параметрами nXDest, nYDest, nWidth, nHeight в логической системе координат контекста устройства-приемника. Оба контекста устройства должны поддерживать растровые операции RC\_BITBLT.

8) DestroyWindow – функция разрушения окна. Вызывается нами единствееный раз при активации пункта меню “Выход”. Полностю разрушает окно, дескриптор которг передан ей в параметре.

9) rand() – функция для рандомного (случайного) заполнения. В нашем проекте используется для случайного заполнения поля фишками. Для того, чтобы это заполнение было каждый раз новое, нужно использовать следующую конструкцию:

srand ((unsigned)time(NULL));

10) DrawImage – функция рисования изображения. Используется для рисования наших игровых фишек.

11) ::CreateFile – используется для создания и/или открытия файла. Файл нам необходим для хранение таблицы рекордов.

12) ::ReadFile – функция чтения файла.

# 3. Проектирование программы

## 3.1 Пользовательские функции

Для того, чтобы наша программа имела законченный вид, нам недостаточно одних стандартных процедур и функций. Нам потребуется реализовать свои собственные.

При проектированиии игры использовались следующие функции:

1. void Paint(HDC hdc,HDC hdcMem);

Функция вывода на экран всей графики.

POINT \* len(int StartX, int StartY,int &length);

Функция подсчёта шаров в одной линии. Осуществляет поиск ряда по вертикали и горизонтали от текущей позиции, задаваемой параметрами StartX и StartY. Результат возвращается в переменную length.

1. void EndPathfinder (void);

Функция очистки динамического массива, содержащего координаты точек проходимого пути для шарика.

1. int FindPath (int startX, int startY, int targetX, int targetY)

Функция поиска пути по алгоритму A-star. В начале работы просматриваются узлы, смежные с начальной; выбирается тот из них, который имеет минимальное значение *f(x)*, после чего этот узел раскрывается. На каждом этапе алгоритм оперирует с множеством путей из начальной точки до всех ещё не раскрытых (листовых) вершин графа («множеством частных решений»), которое размещается в [очереди с приоритетом](http://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_%D1%81_%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%BE%D0%BC). Приоритет пути определяется по значению *f(x) = g(x) + h(x)*. Алгоритм продолжает свою работу до тех пор, пока значение *f(x)* целевой вершины не окажется меньшим, чем любое значение в очереди (либо пока всё дерево не будет просмотрено). Из множественных решений выбирается решение с наименьшей стоимостью.

1. void dest\_len(POINT \* PatchDestroy);

Функция очистки массива из координат точек шаров расположенных в одной линии.

1. void create\_bubbles(HWND hWnd);

Функция добавления шариков в массив. Осуществляет случайное заполненение поля шариками, также осужествляет проверку на завершение игры (игра завершается когда заполняется всё поле). На каждом ходе функция добавляет три шарика, в случае попадания шарика в занятую ячейку осуществляется поиск свободной.

## 3.2 Хранение данных

Данные хранятся в статическом массиве типа int размером 9х9. Младшее слово используется для хранения номера шарика, а старший для хранения текущего состояния (используется для анимации активации, уничтожения, и появления шарика).

При сохранении игры осуществляется запись содержимого массива хранящего порядковый номер, и состояние шарика. Затем дописываются очки набранные игроком и время проведённое в игре. При загрузки производятся обратные действия.

## 3.3 Воспроизведение анимации

Для воспроизведения анимации используется стандартный таймер Windows, который через заданный промежуток времени посылает сообщение WM\_TIMER и через этот промежуток времени осуществляется загрузка другого bitmap'а из Image\_List и так повторятся циклически. В моём случае период срабатывания таймера равен 50ms.

## 3.4 Подсчёт проведённого в игре времени

Для подсчёта времени используется функция GetTickCount(), которая возвращается количество миллисекунд с момента запуска системы. Для расчёта времени работы при создании окна записывается в переменную T1 время запуска программы. Проведённое время в игре рассчитывается как T1-T2, где T2 текущее количество миллисекунд с момента запуска системы.

## 3.5 Начисление очков

Начисление очков производится за каждый уничтоженный шарик по единице.

Полный листинг всех процедур и функций представлен в приложении В «ЛИСТИНГ ПРОГРАММЫ».

# 4. Реализация программы

## 4.1 Реализация главного окна игры

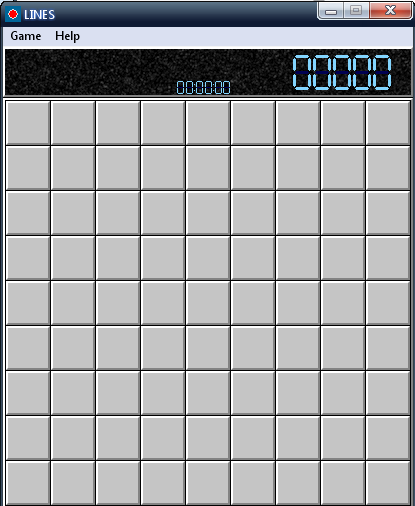


Рис. 1 – «Главное окно программы»

## 4.2 Реализация перемещения шариков

Совершим произвольное количество ходов и увидим, что все шарики перемещаются совершенно правильно.

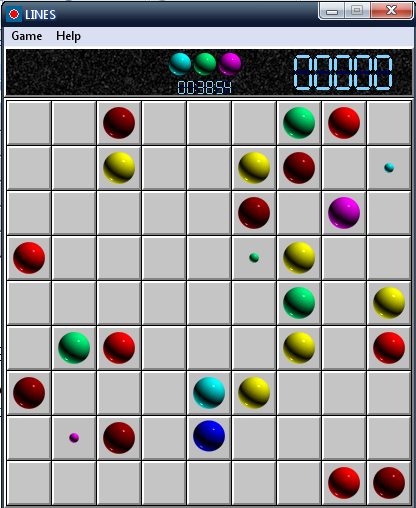


Рис. 2 «Реализация перемещения шаров»

## 4.3 Реализация новой игры

На вкладке меню «Игра» выберем пункт «Новая игра» и увидим следующее:

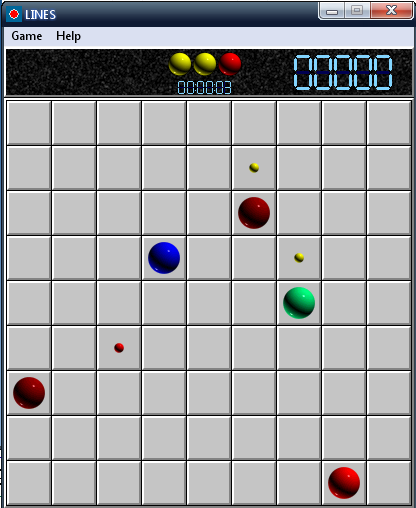


Рис. 3 «Новая игра»

## 4.4 Составление шаров в линию

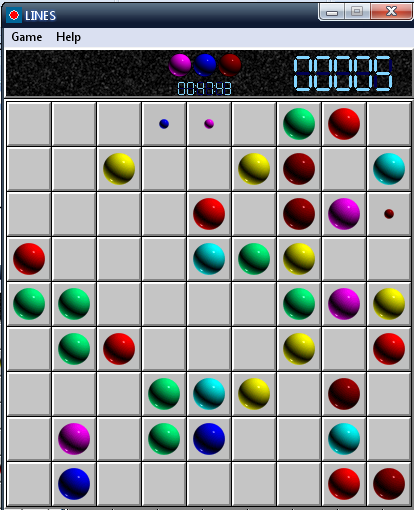
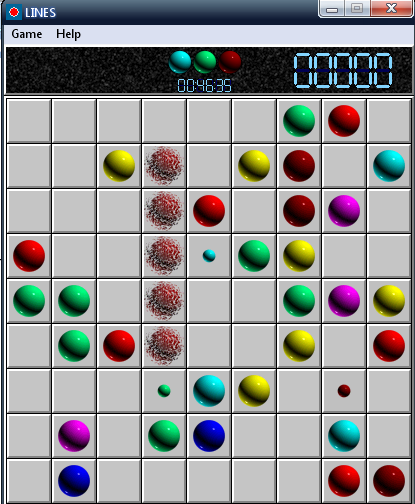
****

Рис. 4 «Реализация уничтожения шаров»

## 4.5 Реализация сохранения и загрузки игры

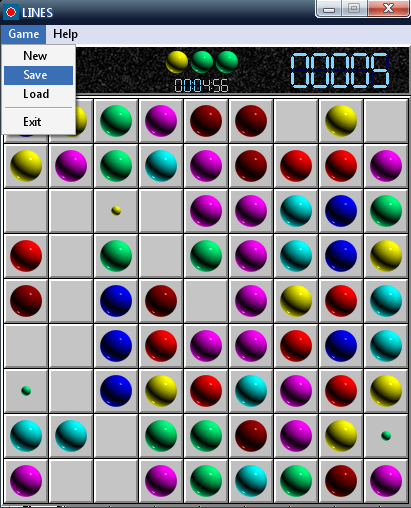
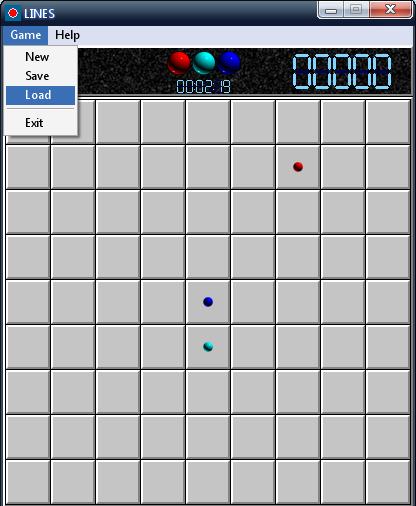
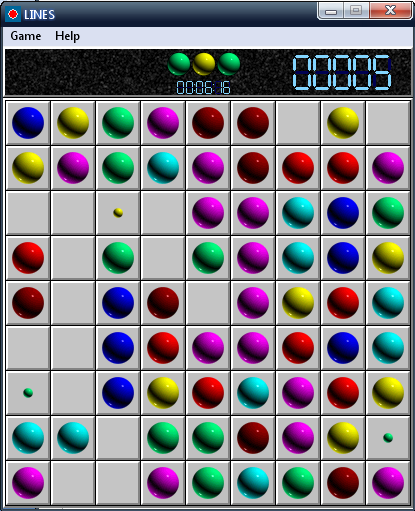


Рис. 5 «Сохранение и загрузка»

# 5. Тестирование

**Тест 1. Корректный и стабильный запуск программы**

При каждом запуске приложение готово выполнять поставленную задачу. Никаких ошибок не происходит. Тест пройден.

**Тест 2. Перемещение шарика**

При клике на шарик с последующим кликом на свободную клетку шарик перемещается при наличии свободного пути.Тест пройден.

**Тест 3. Попытка переместить шарик в закрытую область**

При попытке переместить шарик через непроходимую область ничего не происходит. Тест пройден.

**Тест 4. Новая игра**

После запуска новой игры поле обновилось таймер и очки обнулированы, можно начинать играть. Тест пройден.

**Тест 5. Выход**

После выбора в меню выхода или по нажатию на крестик игра закрывается. Тест пройден.

**Тест 6. Сохранение с последующей загрузкой**

После сохранения и загрузки игры информация на таймере количество очков, расположение и цвет шаров восстановились. Тест пройден.

**Тест 6. Составление в ряд шаров**

После составления в ряд по вертикали или горизонтали пяти и более шаров олного цвета происходит их уничтожение с последующим увеличением очков.

Можно сделать вывод, что программа прошла тестирование. Ее использование не вызвало проблем, все действия она выполняла стабильно и верно.

# 6. Руководство пользователя

После запуска приложения необходимо либо выбрать пункт меню Game -> New для начала новой игры, либо Game -> Load для загрузки предыдущей сохранённой игры.

Управление в игре осуществляется левыми кликами мыши по квадратам поля.

При необходимости сохранения игры необходимо выбрать пункт меню Game -> Save.

Для выхода из игры необходимо нажать крестик окна приложения либо выбрать пункт меню Game -> Exit.

# ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы была решена поставленная задача. Задача представляет собой windows-приложение, эмулирующее игру «LINES» с использованием WinAPI. Приложение имеет интуитивно-понятный интерфейс. В ходе тестирования приложение показало себя стабильным и устойчивым к некорректным действиям пользователя.

Для разработки приложения использовался язык высокого уровня C++. Средой разработки была выбрана среда Microsoft Visual Studio 2015.

# СПИСОК ЛИТЕРАТУРЫ

1. Win32 API Щупак
2. Win32API Основы программирования
3. Саймон Р.Windows 2000 API.1085стр

# Листинг программы

## 5.0.cpp

#include "stdafx.h"

#include "5.0.h"

#include "PathFinder.h"

#include <commctrl.h>

#include <mmsystem.h>

#include <ctime>

#define MAX\_LOADSTRING 100

#define TEMER\_TIME 2

HINSTANCE hInst;

TCHAR szTitle[MAX\_LOADSTRING];

TCHAR szWindowClass[MAX\_LOADSTRING];

HIMAGELIST himl, himl2, himl3, himl4;

DWORD T1, T2;

TIME my\_time = {0};

int startX, startY;

POINT fetus[3];

BOOL START = FALSE;

int SCORE = 0, COUNT\_BUBBLES = 0;

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

void Paint(HDC hdc, HDC hdcMem)

{

BITMAP bm;

HDC hDC = CreateCompatibleDC(hdc);

SelectObject(hDC, (HBITMAP)hBitmapBKG);

GetObject(hBitmapBKG, sizeof(bm), &bm);

BitBlt(hdcMem, 0, 0, bm.bmWidth, bm.bmHeight, hDC, 0, 0, SRCCOPY);

if (START)

{

for (int i = 0; i < 3; i++)

{

ImageList\_Draw(himl3, LOWORD(walkability[fetus[i].x][fetus[i].y]) / 22, hdcMem, 163 + i \* 25, 5, ILD\_TRANSPARENT);

}

}

int mul = 10000;

for (int i = 0; i < 5; i++)

{

ImageList\_Draw(himl4, (SCORE / mul) % 10, hdcMem, 290 + i \* 20, 8, ILD\_TRANSPARENT);

mul /= 10;

}

ImageList\_Draw(himl2, my\_time.hour / 10, hdcMem, 174, 34, ILD\_TRANSPARENT);

ImageList\_Draw(himl2, my\_time.hour % 10, hdcMem, 183, 34, ILD\_TRANSPARENT);

ImageList\_Draw(himl2, my\_time.min / 10, hdcMem, 193, 34, ILD\_TRANSPARENT);

ImageList\_Draw(himl2, my\_time.min % 10, hdcMem, 202, 34, ILD\_TRANSPARENT);

ImageList\_Draw(himl2, my\_time.sec / 10, hdcMem, 212, 34, ILD\_TRANSPARENT);

ImageList\_Draw(himl2, my\_time.sec % 10, hdcMem, 220, 34, ILD\_TRANSPARENT);

for (int i = 0; i < 9; i++)

{

for (int j = 0; j < 9; j++)

{

if (walkability[i][j])

{

ImageList\_Draw(himl, LOWORD(walkability[i][j]) + HIWORD(walkability[i][j]) - 1, hdcMem, 45 \* i + 5, 45 \* j + 56, ILD\_NORMAL);

}

}

}

DeleteDC(hDC);

}

int APIENTRY \_tWinMain(HINSTANCE hInstance,

HINSTANCE hPrevInstance,

LPTSTR lpCmdLine,

int nCmdShow)

{

MSG msg;

LoadString(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadString(hInstance, IDC\_MY50, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

if (!InitInstance(hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_MY50));

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

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int)msg.wParam;

}

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEX 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 = (HICON)LoadImage(hInstance, MAKEINTRESOURCE(IDI\_MY50), IMAGE\_ICON, 48, 48, LR\_DEFAULTCOLOR);

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

wcex.hbrBackground = nullptr;

wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_MY50);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = (HICON)LoadImage(hInstance, MAKEINTRESOURCE(IDI\_MY50), IMAGE\_ICON, 16, 16, LR\_DEFAULTCOLOR);

return RegisterClassEx(&wcex);

}

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance;

HWND hWnd = CreateWindow(szWindowClass, szTitle, WS\_CAPTION | WS\_OVERLAPPED | WS\_MINIMIZEBOX | WS\_SYSMENU,

CW\_USEDEFAULT, 0, 426, 520, NULL, NULL, hInstance, NULL);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

void CreateBubbles(HWND hWnd)

{

int x, y;

srand((unsigned)time(nullptr));

if (!START)

{

for (int i = 0; i < 5; i++)

{

do

{

x = rand() % 9;

y = rand() % 9;

} while (walkability[x][y]);

walkability[x][y] = (rand() % 7) \* 22 + 1;

COUNT\_BUBBLES++;

}

}

else

{

for (int i = 0; i < 3; i++)

{

walkability[fetus[i].x][fetus[i].y] &= zero;

}

}

for (int i = 0; i < 3; i++)

{

do

{

x = rand() % 9;

y = rand() % 9;

}

while (walkability[x][y]);

fetus[i].x = x;

fetus[i].y = y;

walkability[x][y] = walkability\_fetus;

walkability[x][y] += (rand() % 7) \* 22 + 1;

COUNT\_BUBBLES++;

if (COUNT\_BUBBLES == mapWidth \* mapHeight)

{

COUNT\_BUBBLES = 0;

PlaySound(MY\_SOUND\_CANT\_MOVE, nullptr, SND\_FILENAME | SND\_ASYNC);

START = FALSE;

if (MessageBox(nullptr, \_T("GAME OVER"), \_T(""), MB\_OK))

PostMessage(hWnd, WM\_COMMAND, ID\_GAME\_NEW, 0);

break;

}

PlaySound(MY\_SOUND\_FETUS, nullptr, SND\_FILENAME | SND\_ASYNC);

}

}

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

{

int wmId;

PAINTSTRUCT ps;

static HANDLE hFile = nullptr;

static int x, y;

static BITMAP bm;

static bool ANIMATE\_RUN = false;

DWORD NumberOfBytesRead;

static HDC hdc;

static HDC hdcMem;

static HBITMAP hBmpFrame;

static int COUNT\_RUN = 0;

static int COUNT\_FETUS = 0;

static int COUNT\_DESTROY = 0;

static int LENGTH\_DESTROY = 0;

static POINT\* PatchDestroy;

switch (message)

{

case WM\_LBUTTONDOWN:

{

if (HIWORD(lParam) <= 56 || COUNT\_RUN || COUNT\_FETUS || COUNT\_DESTROY || !START)

break;

if (ANIMATE\_RUN)

walkability[x][y] &= zero;

startX = x;

startY = y;

x = (LOWORD(lParam) - 5) / 45;

y = (HIWORD(lParam) - 56) / 45;

if (HIWORD(walkability[x][y]) ^ HIWORD(walkability\_fetus) && walkability[x][y])

{

ANIMATE\_RUN = true;

PlaySound(MY\_SOUND\_MOVE, nullptr, SND\_FILENAME | SND\_ASYNC);

}

else if (ANIMATE\_RUN)

{

ANIMATE\_RUN = false;

path = FindPath(startX, startY, x, y);

if (path == found)

{

COUNT\_RUN = pathLength;

}

}

break;

}

case WM\_COMMAND:

{

wmId = LOWORD(wParam);

switch (wmId)

{

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

case ID\_FILE\_32771: //save

{

walkability[x][y] &= zero;

hFile = CreateFile(FILE\_PATH, GENERIC\_WRITE, FILE\_SHARE\_READ, nullptr, CREATE\_ALWAYS,

FILE\_ATTRIBUTE\_NORMAL, nullptr);

if (hFile)

{

WriteFile(hFile, walkability, sizeof(walkability), &NumberOfBytesRead, nullptr);

WriteFile(hFile, &SCORE, sizeof(int), &NumberOfBytesRead, nullptr);

WriteFile(hFile, &my\_time, sizeof(TIME), &NumberOfBytesRead, nullptr);

CloseHandle(hFile);

}

else

MessageBox(hWnd,\_T("Невозможно открыть файл"),\_T("Ошибка"), MB\_ICONWARNING | MB\_OK);

break;

}

case ID\_FILE\_32772: //load

{

ANIMATE\_RUN = false;

hFile = CreateFile(FILE\_PATH,GENERIC\_READ,FILE\_SHARE\_READ, nullptr,OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL, nullptr);

if (hFile)

{

ReadFile(hFile, walkability, sizeof(walkability), &NumberOfBytesRead, nullptr);

ReadFile(hFile, &SCORE, sizeof(int), &NumberOfBytesRead, nullptr);

ReadFile(hFile, &my\_time, sizeof(TIME), &NumberOfBytesRead, nullptr);

CloseHandle(hFile);

COUNT\_BUBBLES = 0;

int k = 0;

for (int i = 0; i < 9; i++)

{

for (int j = 0; j < 9; j++)

{

if (walkability[i][j])

{

COUNT\_BUBBLES++;

if (!(HIWORD(walkability[i][j]) ^ HIWORD(walkability\_fetus)))

{

fetus[k].x = i;

fetus[k++].y = j;

}

}

}

}

START = TRUE;

T1 = GetTickCount() - (my\_time.hour \* 3600 + my\_time.min \* 60 + my\_time.sec) \* 1000;

}

else

{

MessageBox(hWnd, \_T("Невозможно открыть файл"), \_T("Ошибка"), MB\_ICONWARNING | MB\_OK);

}

break;

}

case ID\_GAME\_NEW: //new

{

ANIMATE\_RUN = false;

START = FALSE;

for (int i = 0; i < 9; i++)

{

for (int j = 0; j < 9; j++)

{

walkability[i][j] = 0;

}

}

COUNT\_BUBBLES = 0;

SCORE = 0;

CreateBubbles(hWnd);

START = TRUE;

T1 = GetTickCount();

break;

}

default:

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

}

break;

}

case WM\_CREATE:

{

himl = ImageList\_Create(40, 40, ILC\_COLOR24, 154, 0);

himl2 = ImageList\_Create(7, 13, ILC\_COLOR32 | ILC\_MASK, 10, 0);

himl3 = ImageList\_Create(25, 25, ILC\_COLOR24 | ILC\_MASK, 7, 0);

himl4 = ImageList\_Create(18, 35, ILC\_COLOR32 | ILC\_MASK, 10, 0);

ImageList\_Add(himl, hBitmapNum1, (HBITMAP)nullptr);

ImageList\_Add(himl, hBitmapNum2, (HBITMAP)nullptr);

ImageList\_Add(himl, hBitmapNum3, (HBITMAP)nullptr);

ImageList\_Add(himl, hBitmapNum4, (HBITMAP)nullptr);

ImageList\_Add(himl, hBitmapNum5, (HBITMAP)nullptr);

ImageList\_Add(himl, hBitmapNum6, (HBITMAP)nullptr);

ImageList\_Add(himl, hBitmapNum7, (HBITMAP)nullptr);

ImageList\_Add(himl2, hBitmapNum8, hBitmapNum12);

ImageList\_Add(himl3, hBitmapNum9, hBitmapNum13);

ImageList\_Add(himl4, hBitmapNum10, hBitmapNum11);

SetTimer(hWnd, TEMER\_TIME, 50, nullptr);

hdc = GetDC(hWnd);

hdcMem = CreateCompatibleDC(hdc);

GetObject(hBitmapBKG, sizeof(bm), &bm);

hBmpFrame = CreateCompatibleBitmap(hdc, bm.bmWidth, bm.bmHeight);

SelectObject(hdcMem, (HBITMAP)hBmpFrame);

break;

}

case WM\_ERASEBKGND:

break;

case WM\_TIMER:

{

switch (wParam)

{

case TEMER\_TIME:

if (START)

{

T2 = GetTickCount();

my\_time.hour = (T2 - T1) / (1000 \* 3600);

my\_time.min = (T2 - T1) / (1000 \* 60) - my\_time.hour \* 60;

my\_time.sec = (T2 - T1) / 1000 - my\_time.hour \* 3600 - my\_time.min \* 60;

}

if (ANIMATE\_RUN)

{

walkability[x][y] = ((((walkability[x][y] >> 16) + 1) % 7) << 16) + (walkability[x][y] & 0x0000ffff);

}

if (COUNT\_FETUS)

{

for (int i = 0; i < 3; i++)

{

if (!(HIWORD(walkability[fetus[i].x][fetus[i].y])))

{

do

{

x = rand() % 9;

y = rand() % 9;

}

while (walkability[x][y]);

walkability[x][y] = walkability\_fetus;

walkability[x][y] += (rand() % 7) \* 22 + 1;

fetus[i].x = x;

fetus[i].y = y;

}

walkability[fetus[i].x][fetus[i].y] = ((((walkability[fetus[i].x][fetus[i].y] >> 16) - 1)) << 16) +

(walkability[fetus[i].x][fetus[i].y] & 0x0000ffff);

}

COUNT\_FETUS--;

if (!COUNT\_FETUS)

{

CreateBubbles(hWnd);

}

}

if (COUNT\_DESTROY)

{

for (int i = 0; i < LENGTH\_DESTROY; i++)

{

walkability[PatchDestroy[i].x][PatchDestroy[i].y] = ((17 - COUNT\_DESTROY) << 16) +

(walkability[PatchDestroy[i].x][PatchDestroy[i].y] & 0x0000ffff);

}

COUNT\_DESTROY--;

if (!COUNT\_DESTROY)

{

for (int i = 0; i < LENGTH\_DESTROY; i++)

{

walkability[PatchDestroy[i].x][PatchDestroy[i].y] = 0;

}

dest\_len(PatchDestroy);

SCORE += LENGTH\_DESTROY;

}

}

if (COUNT\_RUN)

{

if (COUNT\_RUN == pathLength)

{

if (!walkability[pathBank[0].x][pathBank[0].y] ||

!(HIWORD(walkability[x][y]) ^ HIWORD(walkability\_fetus)))

{

walkability[pathBank[0].x][pathBank[0].y] = walkability[startX][startY] & zero;

}

else

{

walkability[pathBank[1].x][pathBank[1].y] = walkability[startX][startY] & zero;

COUNT\_RUN--;

}

walkability[startX][startY] = 0;

}

else

{

if (!walkability[pathBank[pathLength - COUNT\_RUN].x][pathBank[pathLength - COUNT\_RUN].y] ||

!(HIWORD(walkability[x][y]) ^ HIWORD(walkability\_fetus)))

{

walkability[pathBank[pathLength - COUNT\_RUN].x][pathBank[pathLength - COUNT\_RUN].y] =

walkability[pathBank[pathLength - COUNT\_RUN - 1].x][pathBank[pathLength - COUNT\_RUN - 1].y];

walkability[pathBank[pathLength - COUNT\_RUN - 1].x][pathBank[pathLength - COUNT\_RUN - 1].y] = 0;

}

else

{

walkability[pathBank[pathLength - COUNT\_RUN + 1].x][pathBank[pathLength - COUNT\_RUN + 1].y] =

walkability[pathBank[pathLength - COUNT\_RUN - 1].x][pathBank[pathLength - COUNT\_RUN - 1].y];

walkability[pathBank[pathLength - COUNT\_RUN - 1].x][pathBank[pathLength - COUNT\_RUN - 1].y] = 0;

COUNT\_RUN--;

}

}

COUNT\_RUN--;

if (COUNT\_RUN <= 0)

{

if (PatchDestroy = len(x, y, LENGTH\_DESTROY))

{

COUNT\_DESTROY = 10;

COUNT\_BUBBLES -= LENGTH\_DESTROY;

PlaySound(MY\_SOUND\_DESTROY, nullptr, SND\_FILENAME | SND\_ASYNC);

}

COUNT\_FETUS = 4;

path = 0;

EndPathfinder();

}

}

break;

}

Paint(hdc, hdcMem);

BitBlt(hdc, 0, 0, bm.bmWidth, bm.bmHeight, hdcMem, 0, 0, SRCCOPY);

break;

}

case WM\_DESTROY:

{

ImageList\_Destroy(himl);

ImageList\_Destroy(himl2);

ImageList\_Destroy(himl3);

ImageList\_Destroy(himl4);

ReleaseDC(hWnd, hdc);

DeleteDC(hdcMem);

KillTimer(hWnd, TEMER\_TIME);

PostQuitMessage(0);

break;

}

default:

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

}

return 0;

}

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;

}

## 5.0.h

#pragma once

#include "resource.h"

#define FILE\_PATH \_T("data.dat")

HBITMAP hBitmapBKG = (HBITMAP)LoadImage(nullptr, \_T("data//BKG.BMP"), IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum1 = (HBITMAP)LoadImage(nullptr, \_T("data//BALLS1.BMP"), IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum2 = (HBITMAP)LoadImage(nullptr, \_T("data//BALLS2.BMP"), IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum3 = (HBITMAP)LoadImage(nullptr, \_T("data//BALLS3.BMP"), IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum4 = (HBITMAP)LoadImage(nullptr, \_T("data//BALLS4.BMP"), IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum5 = (HBITMAP)LoadImage(nullptr, \_T("data//BALLS5.BMP"), IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum6 = (HBITMAP)LoadImage(nullptr, \_T("data//BALLS6.BMP"), IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum7 = (HBITMAP)LoadImage(nullptr, \_T("data//BALLS7.BMP"), IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum8 = (HBITMAP)LoadImage(nullptr, \_T("data//TIME.BMP"), IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum9 = (HBITMAP)LoadImage(nullptr, \_T("data//NEXT.BMP"), IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum10 = (HBITMAP)LoadImage(nullptr, \_T("data//DIGITS.BMP"), IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum11 = (HBITMAP)LoadImage(nullptr, \_T("data//DIGTR.BMP"), IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum12 = (HBITMAP)LoadImage(nullptr, \_T("data//TIMTR.BMP"), IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum13 = (HBITMAP)LoadImage(nullptr, \_T("data//NEXTTR.BMP"), IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);

TCHAR MY\_SOUND\_MOVE[] = \_T("data//move.wav");

TCHAR MY\_SOUND\_DESTROY[] = \_T("data//destroy.wav");

TCHAR MY\_SOUND\_FETUS[] = \_T("data//fetus.wav");

TCHAR MY\_SOUND\_CANT\_MOVE[] = \_T("data//cantmove.wav");

## PathFinder.cpp

#include "stdafx.h"

#include "PathFinder.h"

int onClosedList = 10;

int openList[mapWidth \* mapHeight + 2];

int whichList[mapWidth + 1][mapHeight + 1];

int openX[mapWidth \* mapHeight + 2];

int openY[mapWidth \* mapHeight + 2];

int parentX[mapWidth + 1][mapHeight + 1];

int parentY[mapWidth + 1][mapHeight + 1];

int Fcost[mapWidth + 1][mapHeight + 1];

int xPath;

int yPath;

//-----------------------------------------------------------------------------

// Реализация алгоритма A\*(A-star)

//-----------------------------------------------------------------------------

int FindPath(int startX, int startY, int targetX, int targetY)

{

int a, b, m, temp, path, tempx, newOpenListItemID = 0;

if (startX == targetX && startY == targetY)

{

return found;

}

if (walkability[targetX][targetY] != walkable &&

HIWORD(walkability[targetX][targetY]) != HIWORD(walkability\_fetus))

{

xPath = startX;

yPath = startY;

return nonexistent;

}

for (int x = 0; x < mapWidth; x++)

{

for (int y = 0; y < mapHeight; y++)

{

whichList[x][y] = 0;

}

}

onClosedList = 2;

int onOpenList = 1;

pathLength = notStarted;

int numberOfOpenListItems = 1;

openList[1] = 1;

openX[1] = startX;

openY[1] = startY;

do

{

if (numberOfOpenListItems != 0)

{

int parentXval = openX[openList[1]];

int parentYval = openY[openList[1]];

whichList[parentXval][parentYval] = onClosedList;

numberOfOpenListItems--;

openList[1] = openList[numberOfOpenListItems + 1];

int v = 1;

do

{

int u = v;

if (2 \* u + 1 <= numberOfOpenListItems)

{

if (Fcost[openX[openList[u]]][openY[openList[u]]] >=

Fcost[openX[openList[2 \* u]]][openY[openList[2 \* u]]])

v = 2 \* u;

if (Fcost[openX[openList[v]]][openY[openList[v]]] >=

Fcost[openX[openList[2 \* u + 1]]][openY[openList[2 \* u + 1]]])

v = 2 \* u + 1;

}

else

{

if (2 \* u <= numberOfOpenListItems)

{

if (Fcost[openX[openList[u]]][openY[openList[u]]] >=

Fcost[openX[openList[2 \* u]]][openY[openList[2 \* u]]])

v = 2 \* u;

}

}

if (u != v)

{

temp = openList[u];

openList[u] = openList[v];

openList[v] = temp;

}

else

break;

} while (true);

for (a = parentXval - 1; a <= parentXval + 1; a++)

{

for (b = parentYval - 1 + abs(parentXval - a); b <= parentYval + 1 - abs(parentXval - a); b++)

{

if (a != -1 && b != -1 && a != mapWidth && b != mapHeight)

{

if (whichList[a][b] != onClosedList)

{

if (walkability[a][b] == walkable || HIWORD(walkability[a][b]) == HIWORD(walkability\_fetus))

{

if (whichList[a][b] != onOpenList)

{

newOpenListItemID++;

m = numberOfOpenListItems + 1;

openList[m] = newOpenListItemID;

openX[newOpenListItemID] = a;

openY[newOpenListItemID] = b;

Fcost[a][b] = 10 \* (abs(a - targetX) + abs(b - targetY));

parentX[a][b] = parentXval;

parentY[a][b] = parentYval;

while (m != 1)

{

if (Fcost[openX[openList[m]]][openY[openList[m]]] <=

Fcost[openX[openList[m / 2]]][openY[openList[m / 2]]])

{

temp = openList[m / 2];

openList[m / 2] = openList[m];

openList[m] = temp;

m = m / 2;

}

else

break;

}

numberOfOpenListItems++;

whichList[a][b] = onOpenList;

}

else

{

if (Fcost[parentXval][parentYval] < Fcost[a][b])

{

parentX[a][b] = parentXval;

parentY[a][b] = parentYval;

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

{

if (openX[openList[x]] == a && openY[openList[x]] == b)

{

m = x;

while (m != 1)

{

if (Fcost[openX[openList[m]]][openY[openList[m]]] <

Fcost[openX[openList[m / 2]]][openY[openList[m / 2]]])

{

temp = openList[m / 2];

openList[m / 2] = openList[m];

openList[m] = temp;

m = m / 2;

}

else

break;

}

break;

}

}

}

}

}

}

}

}

}

}

else

{

path = nonexistent;

break;

}

if (whichList[targetX][targetY] == onOpenList)

{

path = found;

break;

}

} while (true);

if (path == found)

{

int pathX = targetX;

int pathY = targetY;

do

{

tempx = parentX[pathX][pathY];

pathY = parentY[pathX][pathY];

pathX = tempx;

pathLength = pathLength + 1;

} while (pathX != startX || pathY != startY);

pathBank = (POINT\*)realloc(pathBank,

pathLength \* sizeof(POINT));

pathX = targetX;

pathY = targetY;

int cellPosition = pathLength;

do

{

cellPosition--;

pathBank[cellPosition].x = pathX;

pathBank[cellPosition].y = pathY;

tempx = parentX[pathX][pathY];

pathY = parentY[pathX][pathY];

pathX = tempx;

} while (pathX != startX || pathY != startY);

}

return path;

}

void EndPathfinder()

{

if (pathBank)

{

free(pathBank);

pathBank = nullptr;

}

}

POINT\* len(int StartX, int StartY, int& length)

{

int north = 1, south = 1, west = 1, east = 1;

POINT\* PatchDestroy = nullptr;

length = 0;

while (StartY - north != -1 && walkability[StartX][StartY] == walkability[StartX][StartY - north])

north++;

while (StartY + south != mapHeight && walkability[StartX][StartY] == walkability[StartX][StartY + south])

south++;

while (StartX - west != -1 && walkability[StartX][StartY] == walkability[StartX - west][StartY])

west++;

while (StartX + east != mapWidth && walkability[StartX][StartY] == walkability[StartX + east][StartY])

east++;

int height = north + south - 1;

int width = west + east - 1;

if (height >= 5 && width >= 5 && height >= width || height >= 5)

{

length = height;

PatchDestroy = (POINT\*)malloc(sizeof(POINT) \* height);

StartY -= north - 1;

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

{

PatchDestroy[i].x = StartX;

PatchDestroy[i].y = StartY + i;

}

}

else if (width >= 5)

{

length = width;

PatchDestroy = (POINT\*)malloc(sizeof(POINT) \* width);

StartX -= west - 1;

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

{

PatchDestroy[i].x = StartX + i;

PatchDestroy[i].y = StartY;

}

}

return PatchDestroy;

}

void dest\_len(POINT\* PatchDestroy)

{

if (PatchDestroy)

{

free(PatchDestroy);

}

}

## PathFinder.h

#pragma once

typedef struct

{

BYTE hour, min, sec;

} TIME;

const int notfinished = 0, notStarted = 0;

const int nonexistent = 2;

const int walkable = 0;

const int walkability\_fetus = 1376256;

const int mapWidth = 9, mapHeight = 9;

const int zero = 0x0000ffff;

const int found = 1;

static int path = 0;

extern int walkability[mapWidth][mapHeight];

extern int pathLength;

extern POINT\* pathBank;

int FindPath(int startX, int startY, int targetX, int targetY);

void EndPathfinder();

POINT\* len(int StartX, int StartY, int& length);

void dest\_len(POINT\* PatchDestroy);

# Ведомость

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | Дополнительные сведения | | | |
|  | | | | Текстовые документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР КП 1–40 01 01 209 ПЗ | | | | Пояснительная записка | | | | 47 с. | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | | Графические документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| ГУИР 551004 209 СА | | | | А\*  Схема алгоритма. | | | | Формат А1 | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  |  |  |  |  | БГУИР КП 1-40 01 01 209 Д1 | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Игровое приложение «Линии»  Ведомость курсового  проекта |  | | | | Лист | Листов |
| Разраб. | | Ермошин М.А. |  | 09.09.17 | Т |  | |  | 48 | 48 |
| Пров. | | Базылев Е.Н. |  | 01.12.17 | Кафедра ПОИТ  гр. 551004 | | | | | |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |