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 курсовой работе
на тему

Игровое приложение Flappy Bird

БГУИР КР 1-40 01 01 601 ПЗ

Выполнил: студент гр. 851006 Барыко И.Ю.
Руководитель: Жиденко А.Л.

Минск 2020

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой ПОИТ
Лапицкая Н.В.

_____
(подпись)
_____2020 г.

ЗАДАНИЕ
по курсовому проектированию

Студенту Барыко Илье Юрьевичу, 851006

1. Тема работы «Игровое приложение Flappy Bird»
2. Срок сдачи студентом законченной работы: 10.12.2020 г.
3. Исходные данные к работе: Документация по WinAPI

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

Введение.

1. Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству;

2. Разработка программного средства;

3. Тестирование программного средства;

4. Руководство пользователя;

Список используемой литературы

Заключение

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

1. Схема программы на А1

6. Консультант по курсовой работе Жиденко А.Л.

7. Дата выдачи задания 05.09.2020 г.

8. Календарный график работы над курсовой работой на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):
раздел 1 к 20.09.2020 – 15 % готовности работы;
разделы 2, 3 к 13.10.2020 – 30 % готовности работы;
разделы 4, 5 к 02.11.2020 – 60 % готовности работы;
раздел 6 к 26.11.2020 – 90 % готовности работы;
оформление пояснительной записки и графического материала к 01.12.2020 – 100 % готовности работы.
Защита курсовой работы с 01.12.2020 по 10.12.2020 г.

РУКОВОДИТЕЛЬ_____А.Л.Жиденко
(подпись)

Задание приняла к исполнению Барыко И.Ю. _____
(дата и подпись студента)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данная курсовая работа посвящена разработке игрового приложения “Flappy Bird” с использованием возможностей языка C++.

Для разработки приложения в рамках курсового проекта ставятся следующие задачи:

– изучение разработки приложений на WinAPI;

– более углубленное изучение языка C++.

В этой пояснительной записке отображены следующие этапы написания курсовой работы:

1. Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству.

2. Моделирование предметной области и разработка функциональных требований.

3. Проектирование программного средства;

4. Тестирование, проверка работоспособности и анализ полученных результатов;

5. Руководство по установке и использованию.

1 АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ

1.1 Анализ существующих аналогов

В результате анализа существующих приложений, реализованных в качестве игрового приложения “Flappy Bird” было выявлено, было установлено, что подобные аналоги проектируемого программного средства пользуются большим спросом у современных разработчиков, но лишь единицы написаны с использованием WinAPI. Среди них представлены как мобильные, так и десктопные версии. В качестве аналогов были взяты приложения из App Store.

1.1.1 “Faby Bird”

“Faby Bird” находится на первом месте, в качестве аналога игры “Flappy Bird” в App Store. На рисунке 1.1 представлен интерфейс данного приложения.

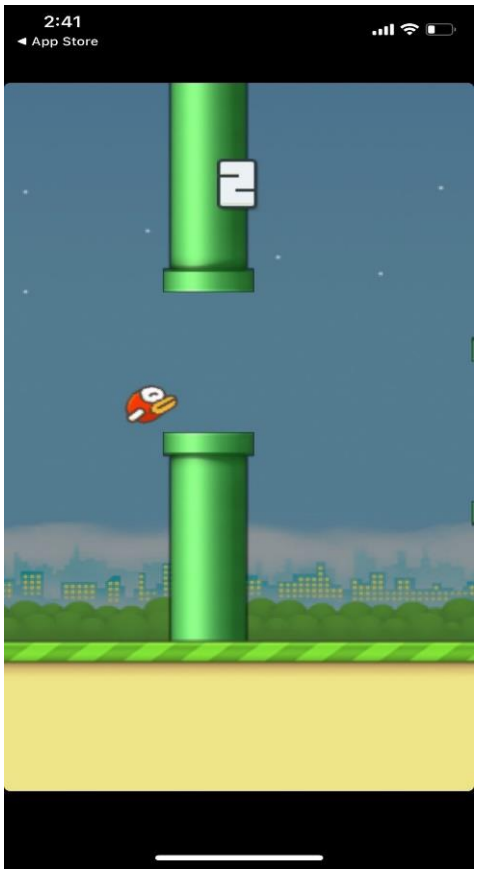

Рисунок 1.1 – Интерфейс приложения «Faby Bird»

Достоинства:
– анимация;
– аудиосопровождение.
Недостатки:
– реклама;
– нет масштабируемости.

1.1.2 “Great Bird”

“Great Bird” находится на втором месте, в качестве аналога игры “Flappy Bird” в App Store. На рисунке 1.2 представлен интерфейс данного приложения.

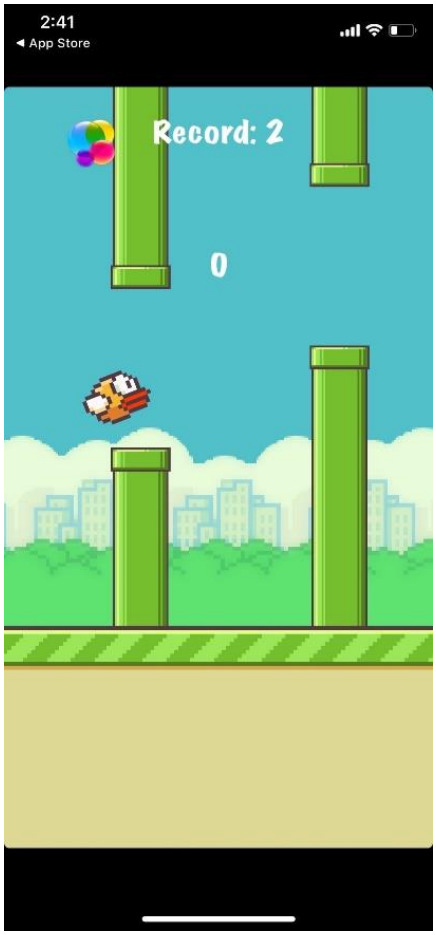

Рисунок 1.2 – Интерфейс приложения «Great Bird»

Достоинства:
– анимация;
– отображение текущего счёта.

Недостатки:

– реклама;

– нет масштабируемости;

– отсутствие главного меню.

1.2 Постановка задачи

Для того, чтобы данное приложение можно было считать аналогом игры “Flappy Bird”, способного к конкуренции с существующими аналогами, на основе анализа популярных приложений, необходимо наличие следующих возможностей:

– динамическое генерирование карты;

– различные модели труб, птиц, а также заднего фона;

– генерирование монет;

– магазин, для покупки моделей;

– отображение набранных очков в конце игры;

– масштабируемость всех объектов;

– приостановление игры;

– перезапуск игры.

В качестве языка программирования, на котором будет написано данное приложение, является C++ с использованием WinAPI.

2 РАЗРАБОТКА ПРОГРММНОГО СРЕДСТВА

2.1 Интерфейс программного средства

Был проведен анализ существующих аналогов и был выдвинут ряд пожеланий к созданному пользовательскому интерфейсу:

– Доходчивость интерфейса, обеспечивающая понимание структуры приложения пользователем.

– Лаконичность интерфейса, обеспечивающая отсутствие перегрузки пользователя информацией.

– Узнаваемость интерфейса. Узнаваемость делает его похожим на аналоги, чтобы пользователь понимал, что можно ожидать от приложения и как оно работает.

– Восприимчивость интерфейса, обеспечивающая короткое время отклика приложения на действия пользователя.

– Эффективность интерфейса, обеспечивающая быстродействия приложения.

В результате рассмотрения пожеланий к пользовательскому интерфейсу было разработано приложение, соответствующее пунктам. Внешний вид программного средства представлен на рисунках 2.1.

Рисунок 2.1 – Интерфейс приложения «Flappy Bird»

2.2 Динамическая генерация карты

Важным условием данного приложения является динамическое генеририрвоание карты, чтобы пользователь не потерял интерес к игре.

Был придуман алгоритм, который будет генерировать часть карты, и заменять ту часть, которая находится вне зоны видимости пользователя. Блок-схема алгоритма представлена на рисунке 2.2.

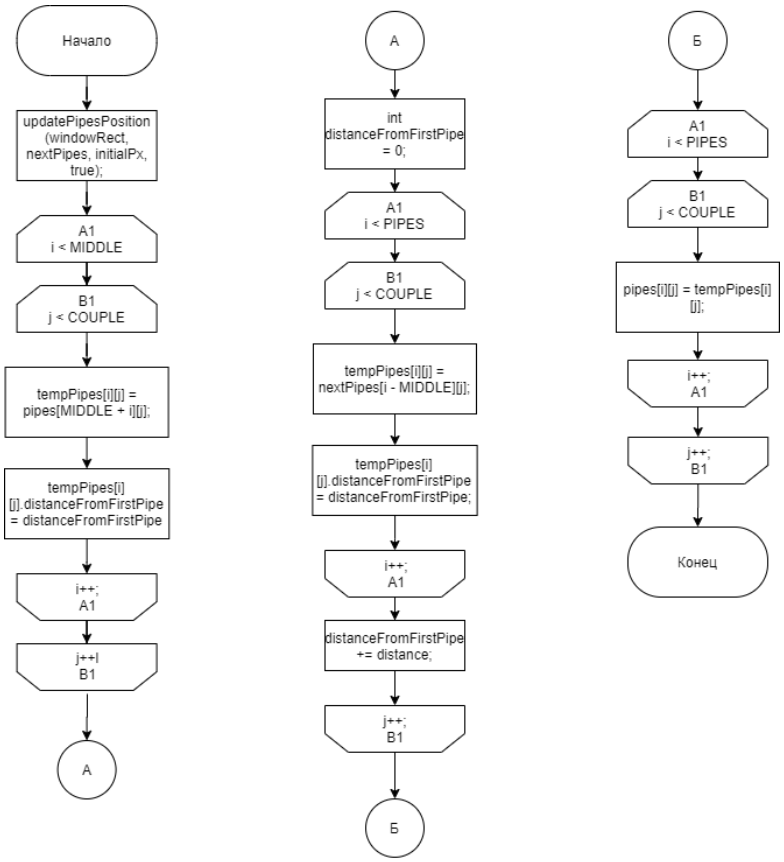

Рисунок 2.2 – Блок-схема алгоритма генерирования карты

Данный алгоритм позволяет генерировать карту, не напрягая систему пользователя, и обеспечивает комфортную игру.

2.3 Магазин моделей объктов

Для того, чтобы разнообразить геймплей пользователя, в игру были добавлены модели объктов, которые пользователь может купить за определённую сумму. Магазин моделей представлен на рисунке 2.3.

Рисунок 2.3 – Интерфейс магазина

Магазин имеет интуитивно-понятный интерфейс. Нажимая клавиши 1-9, пользователь может купить любую модель птицы или труб, расположенных на карте. Если же пользователю не хватает монет, то он не сможет купить в магазине необходимую ему модель объекта, но он может накопить монеты, играя с моделями объктов, которые уже есть у него в наличии.

2.4 Генерирование монет

Для покупки моделей объктов пользователю необходимы монеты, которые он может собрать на карте. Шанс появления монеты на карте состовляет 25%, что является оптимальным для сбора недостающих монет. Пользователь может собрать монеты, пройдя место появления данной монеты. Все монеты зачисляются на счёт после окончания игры. Пример места появления монеты представлен на рисунке 2.4.

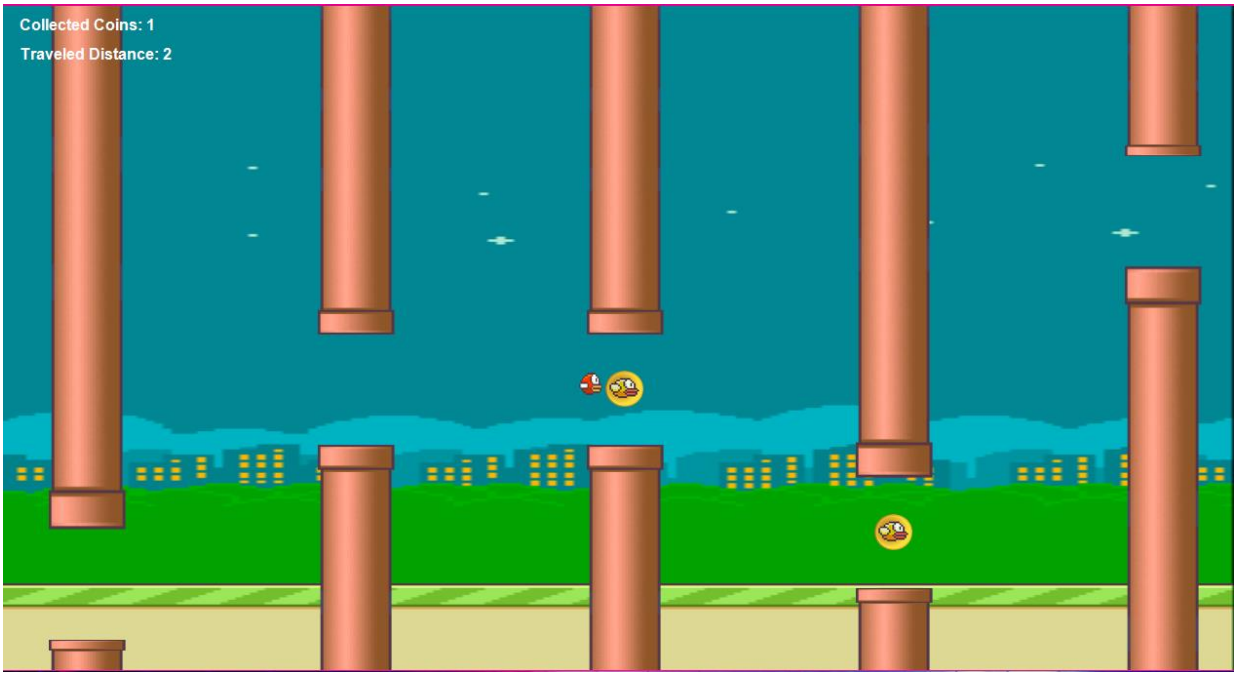

Рисунок 2.4 – Место появления монеты

2.5 Отображение набранных очков в конце игры

Прове анализ прототипов, было выявлено, что не каждое приложение отображает количество набранных очков в конце игры. Я решил создать эту возможность, чтобы у пользователя было время отдохнуть после продолжительной игры. Также будет отображаться не только количесвто набранных очков, но и количество монет, которое есть у пользователя. Пример отображения количества набранных очков представлен на рисунке 2.5.

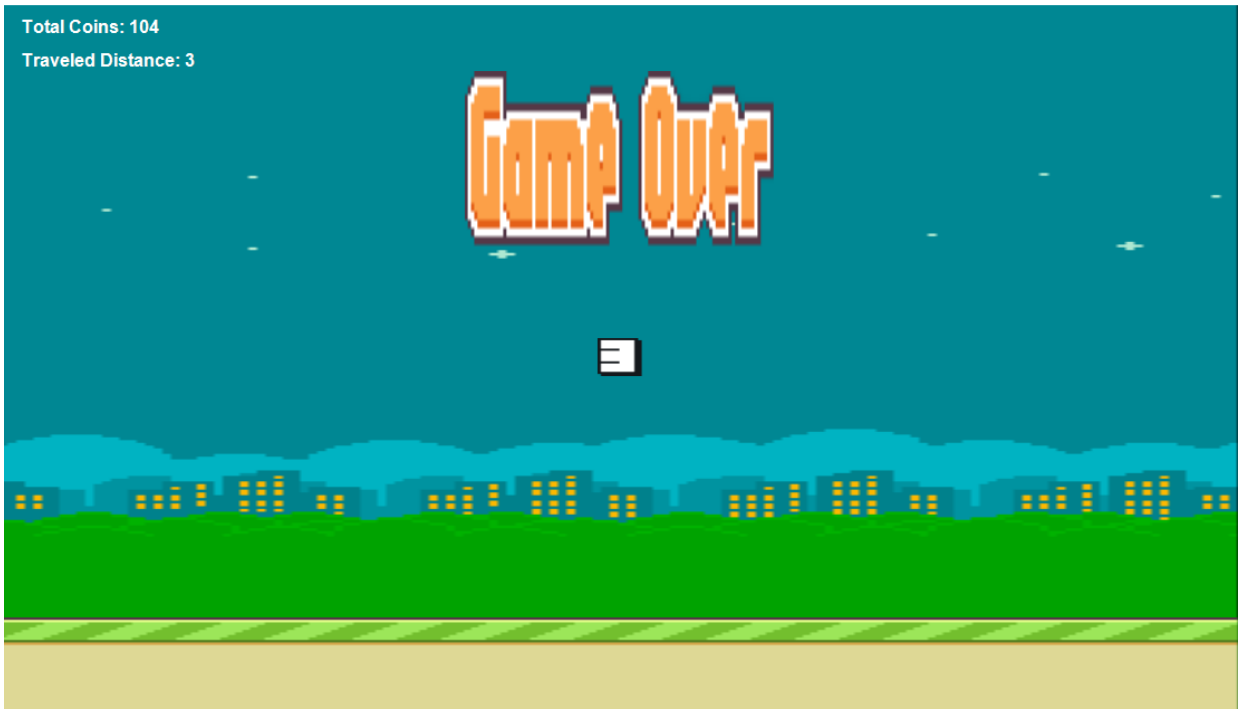

Рисунок 2.5 – Отображение набранных очков

2.6 Масштабируемость объектов

Многие аналоги не имеют возможности масштабирования картинки приложения, что я считаю неправильным, так как у пользователей могу быть различные размеры экрана. Масштабируемость выполняется умножением коэффициента оси абсцисс или ординат на координаты объекта. Коэффициенты находятся путём деления разрешения в данный момент времени на разрешение, которое было выставлено при создании окна. Блок-схема алгоритма нахождения коэффициентов представлена на рисункке 2.6.

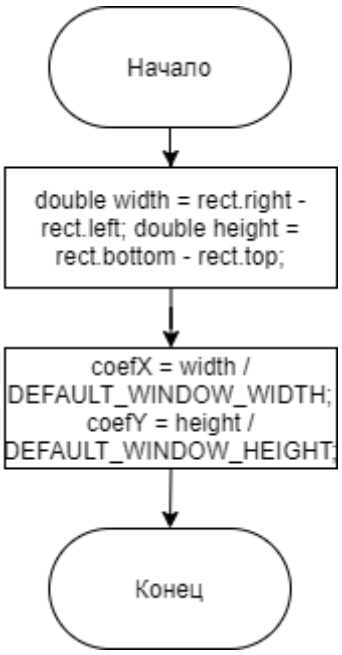

Рисунок 2.6 – Блок-схема алгоритма нахождения коэффициентов

2.7 Перезапуск и пауза игры

Перезапуск игры производится путём выхода в главное меню и началом игры, при этом стоит учесть, что то количество монет, которое было собрано в процессе игры будет утеряно, а карта сгенерировано заново и будет отличаться от прошлой. Пауза также является такой уникальной возможностью и может спасти пользователя в трудный момент, например, нужно помочь кому-нибудь. Благодаря паузе пользователь сможет приостановитиь игру, сохранив набранные очки и собранные монеты, и прододжить с места, на котором остановился.

3 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

В ходе тестирования приложения не было выявлено недостатков программного средства. Была составлена таблица 3.1, показывающая ожидаемые и реальные результаты, полученные при заданных условиях, она представлена ниже.

Таблица 3.1 – Ожидаемые и реальные результаты тестирования

№	Тестовые случаи	Ожидаемый результат	Полученный результат
1.	Запуск программы	Успешный запуск и инициализация начальных значений параметров программы	Тест пройден
2.	Открытие магазина	Успешное отображение ассортимента магазина по нажатии клавиши F1	Тест пройден
3.	Смена заднего фона	Смена заднего фона по нажатии клавиши F5/F6	Тест пройден
4.	Выключение программы	Успешное выключение программы без ошибок	Тест пройден
5.	Запуск игры	Появление движущийся птицы и сгенерированной карты	Тест пройден
6.	Пауза	Приостановление игры по нажатию клавиши SHIFT	Тест пройден
7.	Перезауск игры	Сброс набранных очков и моет, генерация новой карты	Тест пройден
8.	Покупка модели птицы	Выбор птицы в магазине, списывание денег со счёта и отображение выбранной модели в игре	Тест пройден
9.	Покупка модели трубы	Выбор цвета труб в магазине, списывание денег со счёта и отображение выбранной модели в игре	Тест пройден
10.	Покупка при нехватки денег	Отображение старой модели в игре	Тест пройден

Продолжение таблицы 3.1.

11.	Гененрирование карты	Бесконечное генерирование карты	Тест пройден
12.	Генерирование монет	Случайное генерирование монет	Тест пройден
13.	Масштабируемость приложения	Растянутая картинка приложения	Тест пройден
14.	Конец игры при столкновении с объетом карты	Завершение игры	Тест пройден
15.	Отображение набранных очков	Отображения сообщения о конце игры с количеством набранных очков	Тест пройден

Разработка приложения велась с использованием системы контроля версий GitHub, позволившая сохранять состояние программы на каждом отдельном этапе по ходу добавления нового функционала или изменения уже существующего. Появление новых точек возврата происходит посредством группировки изменённых файлов, затем они объединяются под общим именем «коммита», в котором кратко изложена суть изменений. Также можно добавлять к каждому этапу новые файлы, или удалять устаревшие варианты. После накопления определённого количества групп изменений, их следует отправить на удалённый репозиторий, где видна вся история приложения и разница между каждым новым «коммитом».

Путем тщательной проверки тестами, было выявлено, что ошибок в работе программы нет и она является конкурентоспособной.

4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

4.1 Основные требования для запуска

Требования:

– ОС: Версия Microsoft Windows от 8 и выше;

– процессор: не менее 1 ГГц;

– RAM: От 512 МБ;

– место на диске: от 3 Мб свободного места.

Исходя из данных требований следует, что данная программа может запускаться практически на любом компьютере.

4.2 Руководство по установке

Установка программы не требуется, т.к она является переносимым приложением.

Переносимое приложение – программное обеспечение, которое для своего запуска не требует процедуры установки и может полностью храниться на съёмных носителях информации, что позволяет использовать данное ПО на многих компьютерах.

4.3 Руководство по использованию

Для запуска приложения можно воспользоваться exe-файлом, который находится в папке. После появляется окно, изображенное на рисунке 4.1.

Далее у пользователя есть выбор: запустить игру, либо зайти в магазин, либо сменить задний фон. Задний фон меянется нажатием клавиш F5/F6. F5 – светлая тема, F6 – тёмная. Пример тёмной темы приведен на рисунке 4.1, а светлой – на рисунке 4.2.

Игру можно запустить нажатием клавиши SHIFT, после чего пропадет главное меню и начнётся игровой процесс. Игровой процесс изображен на рисунке 4.3.

Рисунок 4.1 – Скриншот запущенной программы

Рисунок 4.2 – Скриншот светлой темы

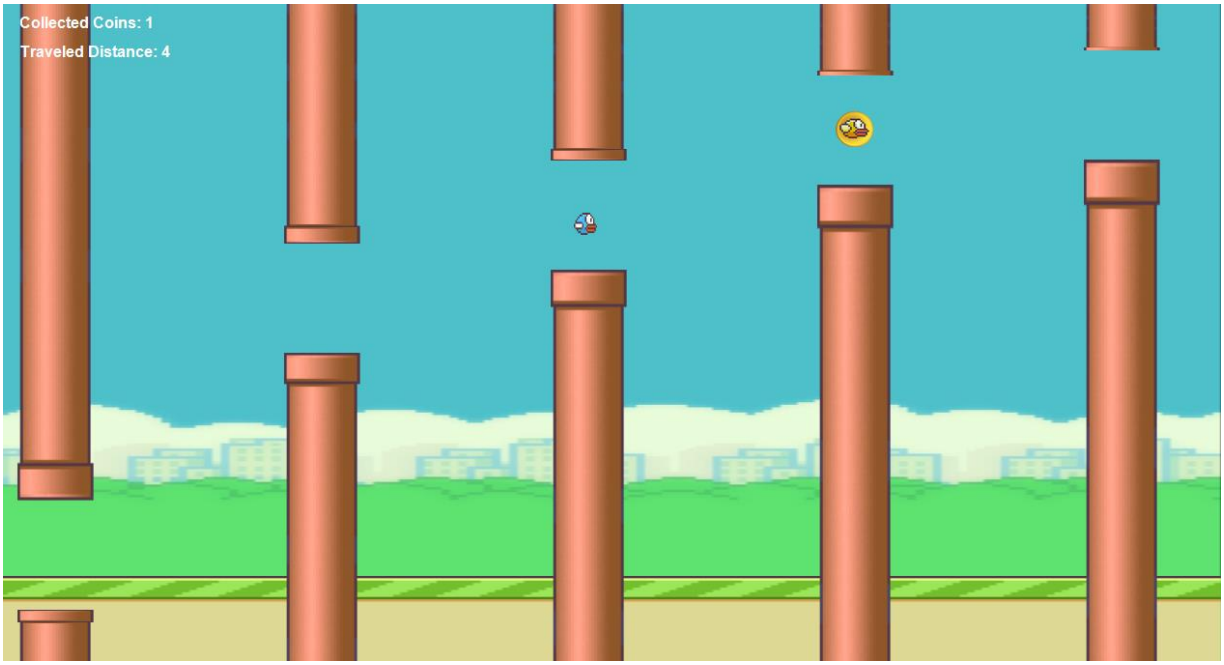

Рисунок 4.3 – Скриншот игрового процесса

Клавишей SHIFT можно приостановить игру, а используя клавишу ESC можно вернуться в главное меню.

Столкнувшись с трубой появится окно с набранными очками, в котором можно нажать SHIFT, чтобы начать игру заново, либо ESC, чтобы вернуться в главное меню. На рисунке 4.4 представлено окно с набранными очками.

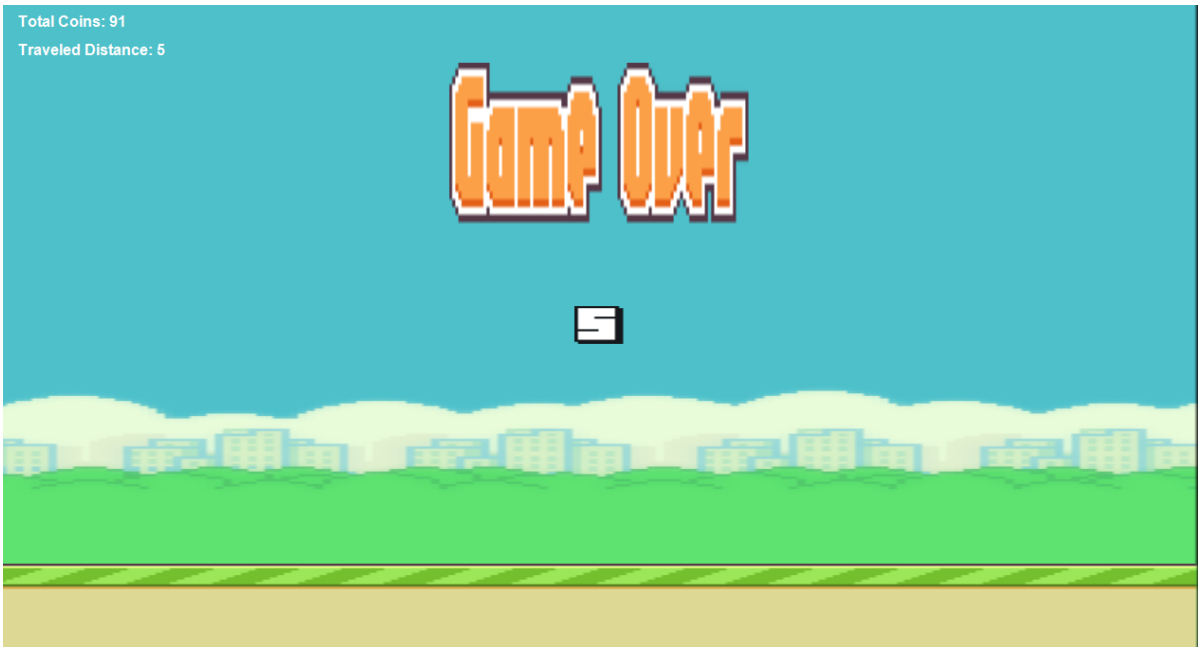

Рисунок 4.4 – Скриншот конца игры

Из главного меню, нажатием клавиши F1, можно зайти в магазин, где надписью “(Current)” обозначена модель, которая установлена в данный момент времени. Магазин представлен на рисунке 4.5.

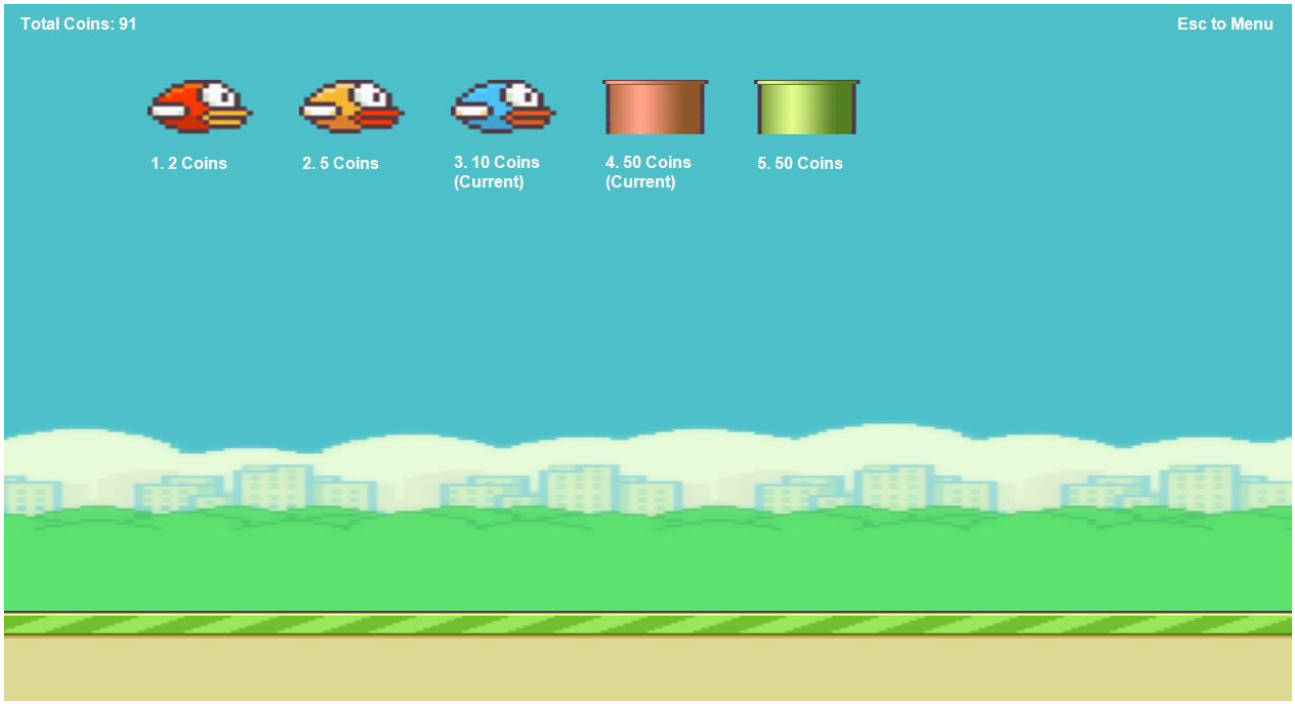

Рисунок 4.5 – Скриншот магазина

Закрытие программы производится нажатием крестика в правом верхнем углу приложения.

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта было разработано игровое приложение Flappy Bird. Стоит отметить, что также были улучшены навыки владения таким языком программирования как C++ и детально изучено взаимодействие с WinAPI.

Разработанное программное средство имеет способность осуществлять следующий функционал:

– динамическое генерирование карты;

– различные модели труб, птиц, а также заднего фона;

– генерирование монет;

– магазин, для покупки моделей;

– отображение набранных очков в конце игры;

– масштабируемость всех объектов;

– приостановление игры;

– перезапуск игры.

Написанный код возможно модифицировать для добавления нового функционала. В дальнейшем планируются улучшения и доработки, вносящие в программу такие возможности как игра по интернету, испытания.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] www.microsoft.com [Электронный портал]. – Электронные данные. – Режим доступа: https://www.microsoft.com/

[2] www.habr.com [Электронный портал]. – Электронные данные. – Режим доступа: https://www.habr.com/

[3] www.apple.com [Электронный портал]. – Электронные данные. – Режим доступа: https://www.apple.com/

[4] Ч. Петзолд «Программирование для Windows 95» (1997) – 737 с.

[5] Ю.Щупак «Win32 API. Разработка приложений для Windows» (2008) – 593 с.

[6]]Richter J. «Programming applications for Microsoft Windows» (2008) – 720с.

[7] emanual.ru [Электронный портал]. – Электронные данные. – Режим доступа: https://www.emanual.ru/download/www.eManual.ru_358.html

ПРИЛОЖЕНИЕ 1
Исходный код программы

Файл main.cpp

```
#include "windows.h"
#include "gdiplus.h"
#include "Game/Game.h"

#define FPS_TIMER 1

const SIZE MIN_WINDOW_SIZE = {640,480};
const double FPS = 1000 / 60;
Game game = Game();

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
   RECT windowRect;
   GetClientRect(hWnd, &windowRect);
   switch (message) {
      case WM_CREATE: {
         SetTimer(hWnd, FPS_TIMER, FPS, nullptr);
         break;
      }
      case WM_SIZE: {
         double width = (windowRect.right - windowRect.left);
         double height = (windowRect.bottom - windowRect.top);
         game.bird.UpdateMoveDistance(width, height);
         game.scene.UpdateObjectPositions(windowRect);
         game.scene.Render(hWnd);
         break;
      }
      case WM_TIMER: {
         game.scene.MovePipe(windowRect);
         game.MoveBird(windowRect, 0.1);
         game.scene.Render(hWnd);
         break;
      }
      case WM_GETMINMAXINFO: {
         LPMINMAXINFO lpMMI = (LPMINMAXINFO) lParam;
         lpMMI->ptMinTrackSize.x = MIN_WINDOW_SIZE.cx;
```

```
            lpMMI->ptMinTrackSize.y = MIN_WINDOW_SIZE.cy;
            double width = (windowRect.right - windowRect.left);
            double height = (windowRect.bottom - windowRect.top);
            game.bird.UpdateMoveDistance(height, width);
            game.scene.UpdateObjectPositions(windowRect);
            game.scene.Render(hWnd);
            break;
        }
        case WM_KEYDOWN: {
            game.KeyAnalyse(hWnd, wParam, windowRect);
            break;
        }
        case WM_PAINT: {
          // game.Run(hWnd, wParam);
            break;
        }
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}

int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPTSTR
lpCmdLine, int nCmdShow)
{
    Gdiplus::GdiplusStartupInput gdiplusStartupInput;
    ULONG_PTR gdiplusToken;
    GdiplusStartup(&gdiplusToken, &gdiplusStartupInput, nullptr);
    WNDCLASSEX wcex; HWND hWnd; MSG msg;
    wcex.cbSize = sizeof(WNDCLASSEX);
    wcex.style = CS_HREDRAW | CS_VREDRAW;
    wcex.lpfnWndProc = WndProc;
    wcex.cbClsExtra = 0;
    wcex.cbWndExtra = 0;
    wcex.hInstance = hInstance;
    wcex.hIcon = LoadIcon(nullptr, IDI_APPLICATION);
    wcex.hCursor = LoadCursor(nullptr, IDC_ARROW);
    wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
```

```
    wcex.lpszMenuName = nullptr;
    wcex.lpszClassName = "Main";
    wcex.hIconSm = wcex.hIcon;
    RegisterClassEx(&wcex);
    hWnd = CreateWindow("Main", "Flappy Bird;",
                        WS_OVERLAPPEDWINDOW, CW_USEDEFAULT,
                        CW_USEDEFAULT, 1280, 720, nullptr, nullptr, hInstance, nullptr);
    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);
    game.Run(hWnd);
    while (GetMessage(&msg, nullptr, 0, 0))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    Gdiplus::GdiplusShutdown(gdiplusToken);
    return (int)msg.wParam;
}
```

Файл Game.h

```
//
// Created by shine on 11/3/2020.
//

#ifndef FLAPPYBIRD_WINAPI_GAME_H
#define FLAPPYBIRD_WINAPI_GAME_H

#include "algorithm"
#include "Scene/Scene.h"
#include "GameState.h"

class Game {
    private:
        const int downModifier = 1;
        const int upModifier = -1;
        WCHAR backgrounds[2][255] = {L"background-day.png",
                                     L"background-night.png"};
        void ResetView(RECT windowRect);
    public:
        Game() = default;
```

```
    GameState gameState = GameState(INTRO);
    Bird bird = Bird(L"redbird-midflap.png");
    Scene scene = Scene(&gameState, &bird);
    void MoveBird(RECT windowRect, double multiplierDirection);
    void KeyAnalyse(HWND hWnd, WPARAM wParam, RECT windowRect);
    int GetCoins();
    void Run(HWND hWnd);
    void Quit();
};


#endif //FLAPPYBIRD_WINAPI_GAME_H
```

Файл Game.cpp

```
//
// Created by shine on 11/3/2020.
//

#include "Game.h"

void Game::KeyAnalyse(HWND hWnd, WPARAM wParam, RECT windowRect) {
  States state = gameState.GetState();
  switch(wParam) {
    case VK_SHIFT: {
      if (state == SCORE || state == SHOP)
        ResetView(windowRect);
      scene.isActive = !scene.isActive;
      if (state != GAMELEVEL) {
        gameState.ChangeToGameLevel();
      }
      break;
    }
    case VK_UP: {
      MoveBird(windowRect, upModifier);
      break;
    }
    case VK_DOWN: {
      MoveBird(windowRect, downModifier);
      break;
    }
```

```
        case VK_F1: {
            if (state == GAMELEVEL) return;
            gameState.ChangeToShop();
            break;
        }
        case VK_ESCAPE: {
            scene.isActive = false;
            ResetView(windowRect);
            gameState.ChangeToIntro();
            break;
        }
        case VK_F5: {
            scene.SetNewBackground(backgrounds[0]);
            break;
        }
        case VK_F6: {
            scene.SetNewBackground(backgrounds[1]);
            break;
        }
    }
    if (state == SHOP)
        scene.KeyAnalyse(hWnd, wParam, windowRect);
}

void Game::ResetView(RECT windowRect) {
    scene.ResetCounter();
    bird.UpdateXY(windowRect);
    Pipe::InitializePipes(scene.pipe.pipes, true);
    scene.pipe.updatePipesPosition(windowRect, scene.pipe.pipes,
DEFAULT_WINDOW_WIDTH, true);
}

void Game::MoveBird(RECT windowRect, double multiplierDirection) {
    if (!scene.isActive) return;
    bird.MoveVertical(windowRect, multiplierDirection);
}


void Game::Run(HWND hWnd) {
    RECT windowRect;
    GetClientRect(hWnd, &windowRect);
```

```
    bird.UpdateXY(windowRect);
    scene.Render(hWnd);
}

int Game::GetCoins() {
    return scene.coins;
}

void Game::Quit() {

}
```

Файл Pipe.h

```
//
// Created by shine on 11/7/2020.
//

#ifndef FLAPPYBIRD_WINAPI_PIPE_H
#define FLAPPYBIRD_WINAPI_PIPE_H

#include "windows.h"
#include <iostream>
#include <gdiplus.h>
#include <ctime>
#include <cwchar>
#include <cstring>
#include <cstdlib>
#include "../Helper/Helper.h"

#define DEFAULT_WINDOW_WIDTH 1280
#define DEFAULT_WINDOW_HEIGHT 720
#define PIPES 16
#define COUPLE 2
#define MIDDLE 8

struct PipeItem {
    int x;
    int y;
    int width;
    int height;
```

```
    int offsetX;
    int distanceFromFirstPipe;
    int offsetY;
    bool hasCoin;
    bool passed;
};

class Pipe {
    private:
        const int pipeWidth = 80;
        PipeItem nextPipes[PIPES][COUPLE];
        int _coins = 0;
        int traveledDistance = 0;
        WCHAR pipeType[255];
        int pipeID = 3;
        double coefX = 1;
        double coefY = 1;
        void GenerateNextPipes(RECT windowRect);
        void GetCoefs(RECT rect);
        static int * GenerateHeightForCouplePipes(int windowHeight, int *pipesHeight);
        static bool randCoin();
    public:
        PipeItem pipes[PIPES][COUPLE];
        explicit Pipe(const WCHAR* _pipeType);
        void CollectCoin(POINTL birdPoint, int i);
        bool CollisionCheck(POINTL birdPoint, int i);
        bool DrawPipes(HDC &memDC, POINTL birdPoint);
        void DrawCoin(Gdiplus::Graphics &graphics, int i);
        static void DrawCollectedCoins(Gdiplus::Graphics &graphics, std::string string, int
coins);
        void DrawTraveledDistance(Gdiplus::Graphics &graphics) const;
        int GetPipeID() const ;
        int GetTraveledDistance() const;
        void IncTraveledDistance(POINTL birdPoint, int i);
        static void InitializePipes(PipeItem (&pipeItem)[PIPES][COUPLE], bool genCoin);
        void Movement();
        void PrintPipes();
        int ResetCounter();
        void SetCoins(int coins);
        void SetNewPipe(WCHAR* name, int id);
        void StopCounting();
```

```
        void ValidateMap(RECT windowRect);
        void updatePipesPosition(RECT windowRect, PipeItem (&pipeItem)[PIPES][COUPLE],
int initialPx, BOOL generate);
};


#endif //FLAPPYBIRD_WINAPI_PIPE_H
```

Файл Pipe.cpp

```
//
// Created by shine on 11/7/2020.
//

#include "Pipe.h"

Pipe::Pipe(const WCHAR* _pipeType) {
    srand((unsigned int)time(nullptr));
    WCHAR * wcharPipe = Helper::GetWCHAR(Helper::GetAssetsDir());
    wcscat(wcharPipe, _pipeType);
    wcscpy(pipeType, wcharPipe);
    InitializePipes(pipes, true);
    InitializePipes(nextPipes, false);
    RECT rect = {0, 0, DEFAULT_WINDOW_WIDTH, DEFAULT_WINDOW_HEIGHT};
    updatePipesPosition(rect, pipes, DEFAULT_WINDOW_WIDTH, true);
}

void Pipe::InitializePipes(PipeItem (&pipeItem)[PIPES][COUPLE], bool genCoin) {
    for (int i = 0; i < PIPES; ++i) {
        for (int j = 0; j < COUPLE; ++j) {
            pipeItem[i][j].x = 0;
            pipeItem[i][j].y = 0 ;
            pipeItem[i][j].width = 0;
            pipeItem[i][j].height = 0;
            pipeItem[i][j].offsetX = 0;
            pipeItem[i][j].distanceFromFirstPipe = 0;
            pipeItem[i][j].offsetY = 0;
            pipeItem[i][j].hasCoin = (genCoin && j == 0) && randCoin();
            pipeItem[i][j].passed = false;
        }
    }
```

```
}

int * Pipe::GenerateHeightForCouplePipes(int windowHeight, int *pipesHeight) {
   pipesHeight[0] = rand() % (windowHeight - 100);
   pipesHeight[1] = windowHeight - pipesHeight[0];
}

void Pipe::updatePipesPosition(RECT windowRect, PipeItem (&pipeItem)[PIPES][COUPLE],
int initialPx, BOOL generate) {
   GetCoefs(windowRect);
   const int distanceBetweenPipesX = int(280 * coefX);
   const int distanceBetweenPipesY = int(120 * coefY);

   int distanceFromFirstPipe = 0;
   int windowHeight = windowRect.bottom - windowRect.top;
   for (int i = 0; i < PIPES; ++i) {
      int offsetY = 0;

      int pipesHeight[2] = {0, 0};
      if (generate)
         GenerateHeightForCouplePipes(windowHeight, pipesHeight);

      for (int j = 0; j < COUPLE; ++j) {
         pipeItem[i][j].x -= pipeItem[i][j].offsetX + pipeItem[i][j].distanceFromFirstPipe ;
         int offsetX = int(pipeItem[i][j].x * coefX) - pipeItem[i][j].x;
         pipeItem[i][j].x = pipeItem[i][j].x + offsetX + distanceFromFirstPipe + initialPx;
         pipeItem[i][j].offsetX = offsetX;
         pipeItem[i][j].distanceFromFirstPipe = distanceFromFirstPipe;
         pipeItem[i][j].y = 0 + offsetY;
         pipeItem[i][j].width = int(pipeWidth * coefX);
         pipeItem[i][j].height -= pipeItem[i][j].offsetY;
         int additionalY = int(pipeItem[i][j].height * coefY) - pipeItem[i][j].height;
         pipeItem[i][j].height = generate ? pipesHeight[j] : pipeItem[i][j].height + additionalY;
         pipeItem[i][j].offsetY = additionalY;

         offsetY += pipeItem[i][j].height + distanceBetweenPipesY;
      }
      distanceFromFirstPipe += distanceBetweenPipesX;
   }
}
```

```
bool Pipe::DrawPipes(HDC &memDC, POINTL birdPoint) {
   Gdiplus::Graphics graphics(memDC);
   bool collision = false;
   for (int i = 0; i < PIPES; ++i) {
      if (!pipes[i][0].passed && (!pipes[0][0].passed || pipes[i-1][0].passed) &&
CollisionCheck(birdPoint, i))
         collision = true;
      CollectCoin(birdPoint, i);
      DrawCoin(graphics, i);
      IncTraveledDistance(birdPoint, i);
      for (int j = 0; j < COUPLE; ++j) {
         Gdiplus::Rect destRect(pipes[i][j].x, pipes[i][j].y, pipes[i][j].width, pipes[i][j].height);
         Gdiplus::Image image(pipeType);
         if (j == 0) {
            image.RotateFlip(Gdiplus::Rotate180FlipX);
         }
         graphics.DrawImage(&image, destRect);
      }
   }
   DrawCollectedCoins(graphics, "Collected Coins: ", _coins);
   DrawTraveledDistance(graphics);
   return collision;
}

void Pipe::DrawCollectedCoins(Gdiplus::Graphics &graphics, std::string string, int coins) {
   std::string text(std::move(string));
   Gdiplus::RectF  rectF(15.0f, 10.0f, 200.0f, 100.0f);
   text += std::to_string(coins);
   Helper::DrawTextZ(graphics, text, rectF);
}

void Pipe::DrawTraveledDistance(Gdiplus::Graphics &graphics) const {
   std::string text("Traveled Distance: ");
   Gdiplus::RectF  rectF(15.0f, 40.0f, 300.0f, 100.0f);
   text += std::to_string(traveledDistance);
   Helper::DrawTextZ(graphics, text, rectF);
}

bool Pipe::CollisionCheck(POINTL birdPoint, int i) {
   int pipeX = pipes[i][0].x;
   int upperHeight = pipes[i][0].height;
```

```
    const int distanceBetweenPipesY = int(120 * coefY);
    int lowerHeight = upperHeight + distanceBetweenPipesY;
    if(birdPoint.x >= pipeX && (birdPoint.y <= upperHeight || birdPoint.y >= lowerHeight - 25 *
coefY))
        return true;
    return false;
}

void Pipe::IncTraveledDistance(POINTL birdPoint, int i) {
    int pipePos = pipes[i][0].x + pipes[i][0].width * coefX;
    if(!pipes[i][0].passed && pipePos <= birdPoint.x)
    {
        pipes[i][0].passed = true;
        pipes[i][1].passed = true;
        traveledDistance++;
    }
}


void Pipe::CollectCoin(POINTL birdPoint, int i) {
    int pipePos = pipes[i][0].x + 20 * coefX;
    if(pipes[i][0].hasCoin && pipePos <= birdPoint.x)
    {
        pipes[i][0].hasCoin = false;
        _coins++;
    }
}

void Pipe::DrawCoin(Gdiplus::Graphics &graphics, int i) {
    if(pipes[i][0].hasCoin) {
        int x = pipes[i][0].x + int(20 * coefX);
        int y = pipes[i][0].height + int(40 * coefY);
        int width = int(40 * coefX);
        int height = int(40 * coefY);
        Gdiplus::Rect rectForCoin(x, y, width, height);
        Gdiplus::Image
coinImg(L"C:\\Users\\shine\\Desktop\\Dev\\FlappyBird_WinAPI\\Assets\\coin.png");
        graphics.DrawImage(&coinImg, rectForCoin);
    }
}
```

```
void Pipe::GetCoefs(RECT rect) {
   double width = rect.right - rect.left;
   double height = rect.bottom - rect.top;
   coefX = width / DEFAULT_WINDOW_WIDTH;
   coefY = height / DEFAULT_WINDOW_HEIGHT;
   if (coefY == 0) {
      coefX = 0.935;
      coefY = 0.935;
   }
}

void Pipe::Movement() {
   for (int i = 0; i < PIPES; ++i) {
      for (int j = 0; j < COUPLE; ++j) {
         pipes[i][j].x -= 4;
      }
   }
}

void Pipe::PrintPipes() {
   for (int i = 0; i < PIPES; ++i) {
      for (int j = 0; j < COUPLE; ++j) {
         std::cout << "pipe = ";
         std::cout << i;
         std::cout << " ";
         std::cout << j << std::endl;

         std::cout << "x = ";
         std::cout << pipes[i][j].x << std::endl;
         std::cout << "y = ";
         std::cout << pipes[i][j].y << std::endl;
         std::cout << "width = ";
         std::cout << pipes[i][j].width << std::endl;
         std::cout << "height = ";
         std::cout << pipes[i][j].height << std::endl;
         std::cout << "offsetX = ";
         std::cout << pipes[i][j].offsetX << std::endl;
         std::cout << "offsetY = ";
         std::cout << pipes[i][j].offsetY << std::endl;
         std::cout << "distanceFromFirstPipe = ";
         std::cout << pipes[i][j].distanceFromFirstPipe << std::endl;
```

```
        }
        std::cout << "_____" << std::endl;
    }
}

void Pipe::ValidateMap(RECT windowRect) {
    if (pipes[MIDDLE][1].x <= 0) {
        GenerateNextPipes(windowRect);
    }
}

void Pipe::GenerateNextPipes(RECT windowRect) {

    int initialPx = pipes[PIPES - 1][0].x + pipes[PIPES - 1][0].width + 200 * coefX;
    InitializePipes(nextPipes, true);
    updatePipesPosition(windowRect, nextPipes, initialPx, true);

    const int distance = nextPipes[2][0].distanceFromFirstPipe -
nextPipes[1][0].distanceFromFirstPipe;
    PipeItem tempPipes[PIPES][COUPLE];

    int distanceFromFirstPipe = 0;
    for (int i = 0; i < MIDDLE; ++i) {
        for (int j = 0; j < COUPLE; ++j) {
            tempPipes[i][j] = pipes[MIDDLE + i][j];    // 0...4
            tempPipes[i][j].distanceFromFirstPipe = distanceFromFirstPipe;
        }
        distanceFromFirstPipe += distance;
    }

    for (int i = MIDDLE; i < PIPES; ++i) {
        for (int j = 0; j < COUPLE; ++j) {
            tempPipes[i][j] = nextPipes[i - MIDDLE][j];  // 5...9
            tempPipes[i][j].distanceFromFirstPipe = distanceFromFirstPipe;
        }
        distanceFromFirstPipe += distance;
    }

    for (int i = 0; i < PIPES; ++i) {
        for (int j = 0; j < COUPLE; ++j) {
            pipes[i][j] = tempPipes[i][j];
```

```
        }
    }
}

bool Pipe::randCoin() {
    int value = rand() % 100;
    return value % 4 == 0;
}

int Pipe::ResetCounter() {
    int coins = _coins;
    _coins = 0;
    return coins;
}

void Pipe::StopCounting() {
    traveledDistance = 0;
}

int Pipe::GetTraveledDistance() const {
    return traveledDistance;
}

void Pipe::SetNewPipe(WCHAR* name, int id) {
    wcscpy(pipeType, name);
    pipeID = id;
}

void Pipe::SetCoins(int coins) {
    _coins = coins;
}

int Pipe::GetPipeID() const {
    return pipeID;
}
```

Файл Bird.h

```
//
// Created by shine on 11/3/2020.
//
```

```
#ifndef FLAPPYBIRD_WINAPI_BIRD_H
#define FLAPPYBIRD_WINAPI_BIRD_H

#define DEFAULT_WINDOW_WIDTH 1280
#define DEFAULT_WINDOW_HEIGHT 720

#include "windows.h"
#include <iostream>
#include <gdiplus.h>
#include <cwchar>

class Bird {
   private:
      const int birdWidth = 25;
      const int birdHeight = 25;
      WCHAR bird[255];
      int birdID = 0;
      double coefX = 1;
      double coefY = 1;
      int x = 0;
      int y = 0;
      int offsetX = 0;
      int offsetY = 0;
      int moveDistance = 10;
      void ValidatePosition(RECT windowRect, double multiplierDirection);
   public:
      explicit Bird(const WCHAR* _bird);
      void DrawBird(HDC &memDC);
      void MoveVertical(RECT windowRect, double multiplierDirection);
      void UpdateMoveDistance(double windowWidth, double windowHeight);
      int GetBirdID() const;
      void GetCoefs(RECT rect);
      void UpdateBirdPosition(RECT windowRect);
      void UpdateXY(RECT windowRect);
      void SetNewBird(const WCHAR* name, int id);
      POINTL GetPos();
};


#endif //FLAPPYBIRD_WINAPI_BIRD_H
```

Файл Bird.cpp

```
//
// Created by shine on 11/3/2020.
//

#include "Bird.h"
#include "../Helper/Helper.h"

Bird::Bird(const WCHAR* _bird) {
   WCHAR * wchar = Helper::GetWCHAR(Helper::GetAssetsDir());
   wcscat(wchar, _bird);
   wcscpy(bird, wchar);
   x = 0;
   y = 0;
}

void Bird::DrawBird(HDC &memDC) {
   Gdiplus::Graphics graphics(memDC);
   Gdiplus::Rect destRect(x, y, birdWidth * coefX, birdHeight * coefY);
   Gdiplus::Image image(bird);
   graphics.DrawImage(&image, destRect);
}

POINTL Bird::GetPos() {
   POINTL point = {x, y};
   return point;
}

void Bird::ValidatePosition(RECT windowRect, double multiplierDirection) {
   int moveDist = y + moveDistance * multiplierDirection;
   if (moveDist <= windowRect.top && multiplierDirection < 0 ||
         moveDist >= windowRect.bottom && multiplierDirection > 0) {
      return;
   }
   y = moveDist;
}

void Bird::UpdateXY(RECT windowRect) {
   int width = (windowRect.right - windowRect.left);
   int height = (windowRect.bottom - windowRect.top);
```

```
    x =  width / 2 - 40;
    y = height / 2;
}
void Bird::MoveVertical(RECT windowRect, double multiplierDirection) {
    windowRect.bottom -= birdHeight;
    ValidatePosition(windowRect, multiplierDirection);
}

void Bird::UpdateMoveDistance(double windowWidth, double windowHeight) {
    int distance = 10;
    double coefX = windowHeight / DEFAULT_WINDOW_HEIGHT;
    distance *= coefX;
    moveDistance = distance;
}
void Bird::GetCoefs(RECT rect) {
    double width = rect.right - rect.left;
    double height = rect.bottom - rect.top;
    coefX = width / DEFAULT_WINDOW_WIDTH;
    coefY = height / DEFAULT_WINDOW_HEIGHT;
    if (coefY == 0) {
        coefX = 0.935;
        coefY = 0.935;
    }
}
void Bird::UpdateBirdPosition(RECT windowRect) {
    GetCoefs(windowRect);
    x -= offsetX;          // Return initial value
    y -= offsetY;
    offsetX = x * coefX - x;
    offsetY = y * coefY - y;
    x += offsetX;          // New value
    y += offsetY;
}

void Bird::SetNewBird(const WCHAR* name, int id) {
    wcscpy(bird, name);
    birdID = id;
}
int Bird::GetBirdID() const {
    return birdID;
```

Обозначение	Наименование	Дополнительные сведения
	Текстовые документы	
БГУИР КР 1–40 01 01 601 ПЗ	Пояснительная записка	40 с.
	Графические документы	
ГУИР 8510006 601 СП	Схема программы на А1	Формат А1

					БГУИР КР 1-40 01 01 601 Д1		
Изм.	Л.	№ докум.	Подп.	Дата		Лист	Листов
Разраб.		Барыко И.Ю.			Игровое приложение Flappy Bird	Т 40	40
Пров.		Жиденко А.Л.				Кафедра ПОИТ гр. 851006	