**РУКОВОДСТВО ПРОГРАММИСТА**

**Игра**

**«FlappyBird»**

**Листов 11**

2022

**АННОТАЦИЯ**

В данном программном документе приведено руководство программиста по работе с приложением «FlappyBird».

В разделе «Назначение и условия применения программы» указаны назначение и функции, выполняемые сайтом, условия, необходимые для его работы.

В разделе «Характеристика приложения» приведено описание основных

характеристик и особенностей.

1. Назначение и условия применения программы

В последнее время игры приобретают всё большую популярность, в связи со своей доступностью и возможностью отвлечься от повседневных задач и отдохнуть.

Игры различных жанров на данный момент имеют спрос, так как это позволяет развивать множество навыков в виртуальном мире, общаться с другими игроками, получая от этого удовольствие. Например, некоторые игры, имеют крайне проработанный сюжет, который многих увлекает, другие имеют сложные механики, к которым крайне сложно привыкнуть, что тренирует внимательность, точность и скорость реакции.

Представим следующую ситуацию: человек, хочет временно отвлечься от дел на перерыве между работой или учёбой и решает установить игру под систему с ОС Windows. Тяжеловесные игровые проекты, очевидно занимают много места и требуют много места на накопителе. В таких условиях удобно скачать простую, но в то же время увлекательную игру, в которой можно провести некоторое свободное время

Главной задачей, которую необходимо было выполнить, было создание игры «FlappyBird».

Создание лёгкой по системным требованиям аркадной игры позволило бы быстро установить её на ПК, и начать играть. Игра простая по физике поведения объектов, но достаточно сложные в самом управления пользователем, что тренирует скорость реакции и позволяет с интересом проводить в ней время.

В ходе производственной практики была создана игра «FlappyBird», предназначенная для возможности развлечься в свободное время, параллельно развивая скорость реакции.

1. Характеристики и работа программы

Требования к игре, которые были выполнены в ходе разработки:

1. Запуск возможен с любой «Windows» от 2007 года выпуска.
2. Лёгкое взаимодействие с игрой, понятное любому пользователю.
3. Псевдослучайная генерация препятствий.

Требования к логике программы, которые были выполнены в ходе разработки:

1. Вход в приложение легко доступен для любого пользователя
2. Авторизация или создание новой учётной записи происходит автоматически
3. Интуитивное главное меню со всеми необходимыми пунктами
4. Игра отображает количество набранных очков и сохраняет их в рекорды после конца игры
5. Возможность смотреть другие локальные результаты пользователей

Для того, чтобы собрать приложение требуется:

1. Скачать и установить Git, по следующей ссылке

https://git-scm.com/download/win/

1. После установки Git, нужно открыть командную строку, создать папку, где будет находиться проект, далее нужно написать git clone https://github.com/Vaiman1/FlappyBird/flappybird.git и дождаться завершения загрузки проекта.
2. Установить любую IDE, поддерживающую .Net, рекомендуется MS Visual Studio
3. Перейти в корень репозитория, открыть файл FlappyBird.csproj

На рисунке 1 представлена файловая архитектура приложения.

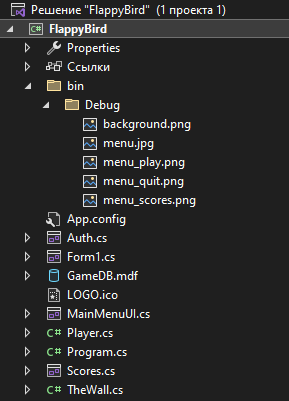


Рисунок 1 – Файловая архитектура приложения

Для работы необходимы все файлы, представленные на рисунке 1, их можно разделить на следующие группы:

1. Форма приложения
2. Ресурсы
3. Стили
4. Точка входа – Program
5. Классы отвечающие за поведение стен и игрока

При создании форума использовался язык C#.

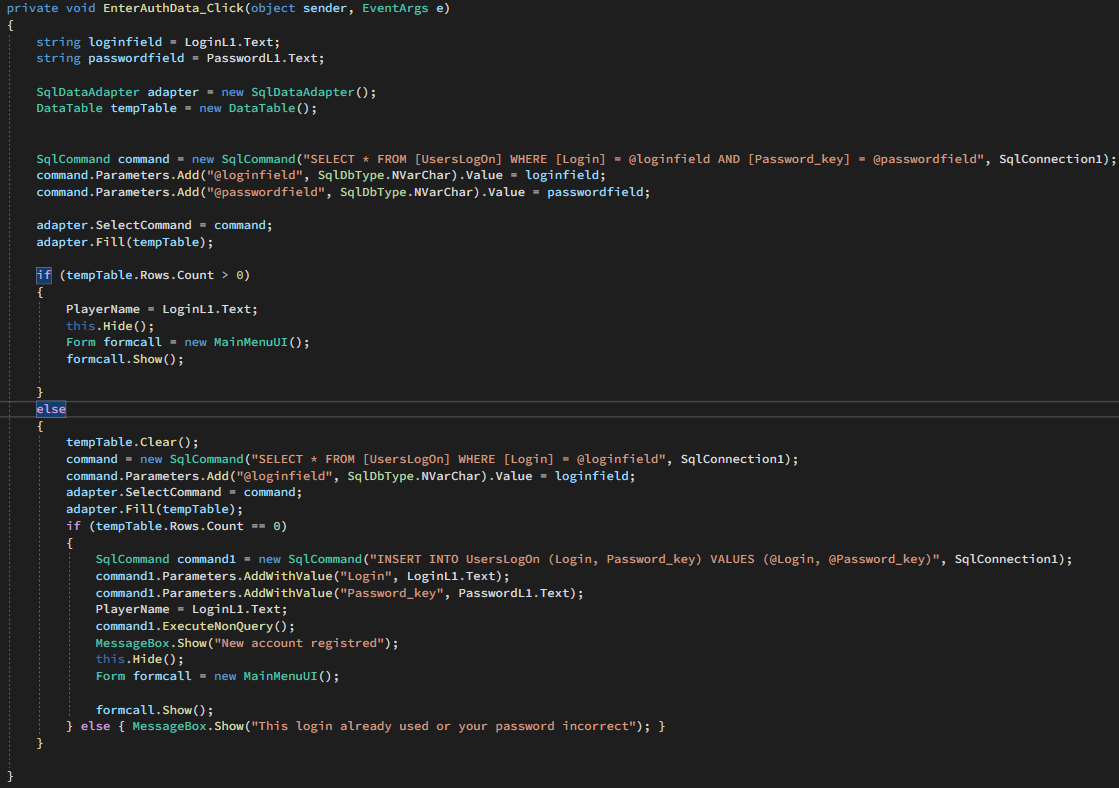
После авторизации становится доступен основной функционал приложения – меню со всем функционалом. На рисунке 2 представлен код проверки и авторизации пользователя.

Рисунок 2 – Код проверки и авторизации пользователя.

При создании использовался фреймворк Windows Forms. Основная форма где происходит отрисовка – Form1. В Form1.cs описаны методы генерации препядствий, их отрисовки и основные методы для управления объектами «Wall» и «Bird» В файле Form1.resx содержится элемент таймера. Также в форме определены тики таймера, которые предназначены для обновления события и вызова update, который в свою очередь вызывает остальные функции.

При создании приложения была реализована случайная генерация препядствий (объектов Wall). При соприкосновении птицы с трубой игра перезапускается, а набранные очки за пройденные препядствия сбрасываются:

На рисунке 2 представлена игра и препядствия.

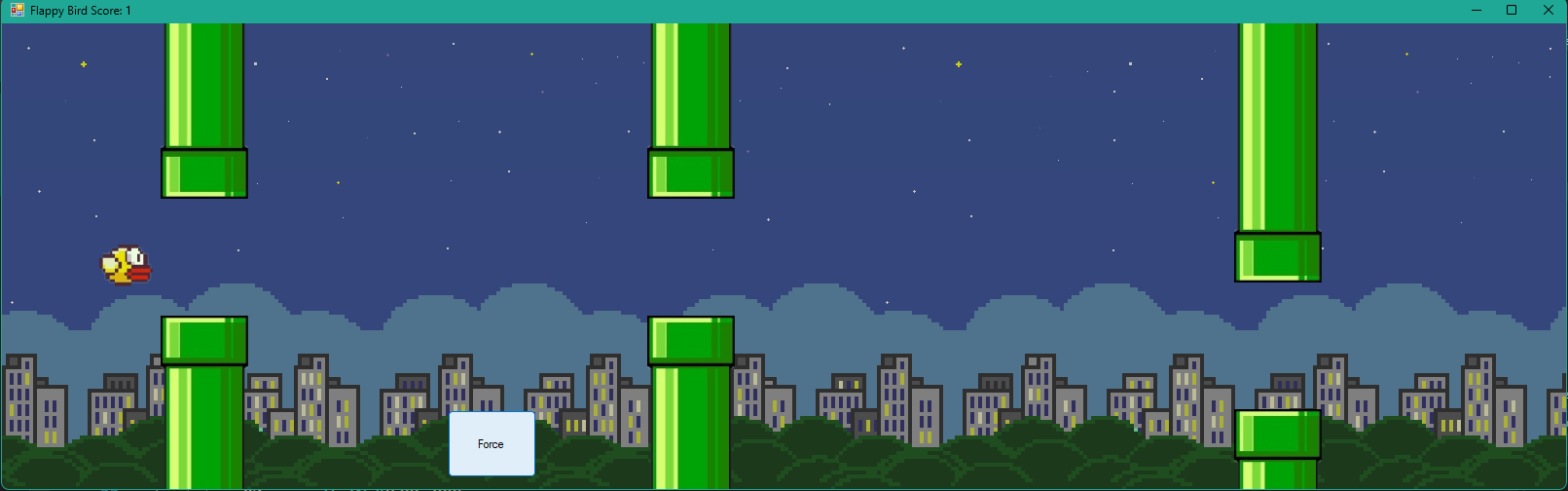


Рисунок 3 – Препядствия в игре

Проверка соприкосновения осуществляется методом Collide. Который просчитывет соприкасается ли пользователь с трубой.

На рисунке 4 предсталена реализация просчёта соприкосновения.

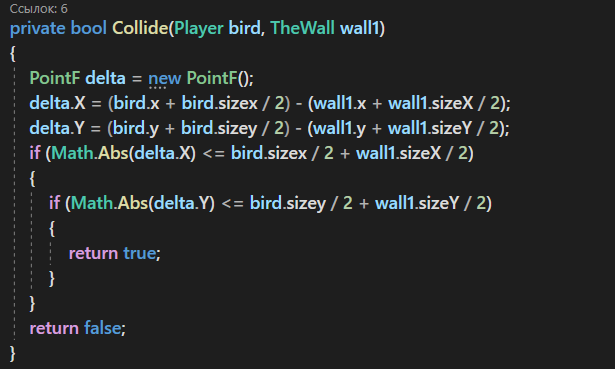


Рисунок 4 – Реализация просчёта соприкосновения

На рисунке видно, что метод просчитвает разницу координат и текстур двух объектов при помощи объекта delta класса PointF, определённого в System.Drawing. Генерация труб происходит рандомно, за счёт изменения высоты труб методом Random.

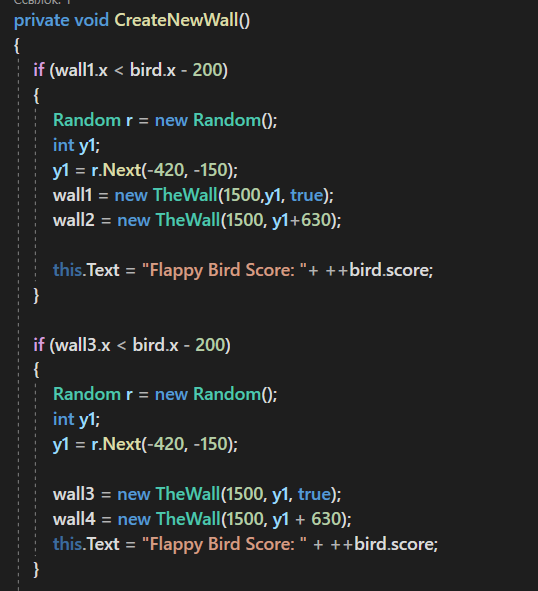


Рисунок 5 – Динамическое отобржаение кнопки на слайдере

На данном рисунке также видно, что при создании объекта Wall, передаётся дополнительная булевая переменная. Данный параметр переворачивает объект Bitmap на 180 градусов, что позволяет корректно отрисовывать верхние препядствия-трубы, реализация класса Wall, представлено на рисунке 6.

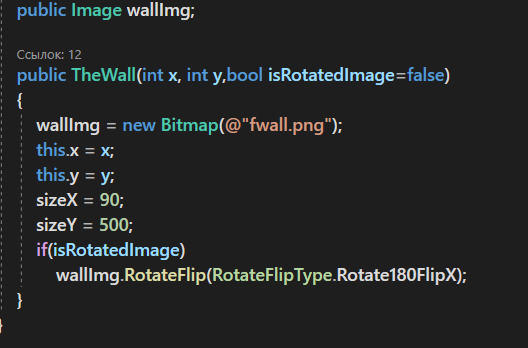


Рисунок 6 – Реализация класса Wall

На рисунке 6 представлен код отрисовки объектов и метод движения.

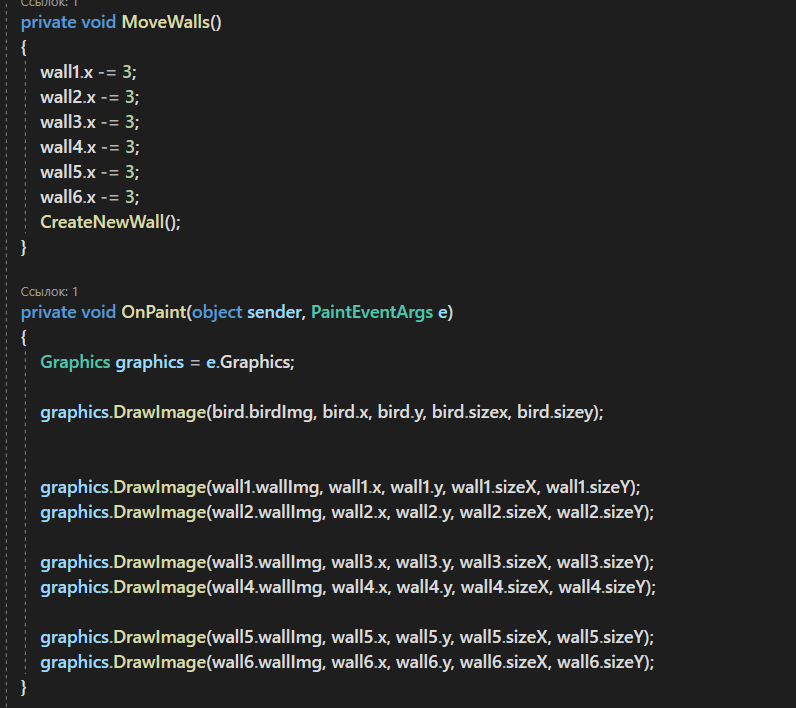


Рисунок 7 – Код обновления положения препядствий

Гравитация влияет на положение птицы. Ниже, на рисунке 7 представлен множитель гравитации и метод отвечающий за гравитацию в целом.

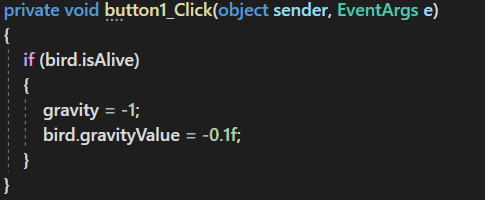


Рисунок 8 – Изменение гравитации по нажатию пользователя

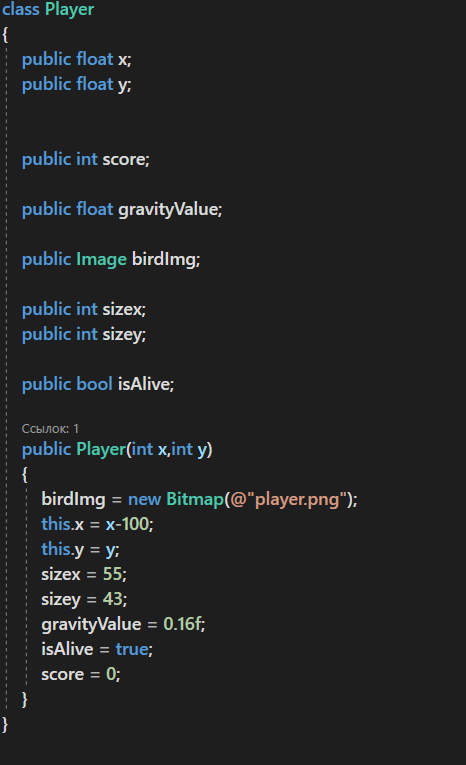
За местоположение, птицы отвечает её класс. Представлено на рисунке 8.

Рисунок 9 – Класс описывающий поведение птицы

Как можно заметить, положение обновляется каждый раз при обновлении метода Update.

ЛИСТИНГ ПРОГРАММЫ

**«Метод Update»**

private void update(object sender, EventArgs e)

{

if (bird.y > 600)

{

bird.isAlive = false;

timer1.Stop();

Init();

}

if (Collide(bird, wall1) || Collide(bird, wall2) || Collide(bird, wall3) || Collide(bird, wall4) || Collide(bird, wall5) || Collide(bird, wall6))

{

bird.isAlive = false;

timer1.Stop();

Init();

}

if (bird.gravityValue != 0.10f)

bird.gravityValue += 0.0046f;

gravity += bird.gravityValue;

bird.y += gravity;

if (bird.isAlive) {

MoveWalls();

}

Invalidate();

}