Федеральное государственное образовательное бюджетное

учреждение высшего профессионального образования

«**Финансовый университет**

**при Правительстве Российской Федерации»**

**Кафедра «Информатика и программирование»**

**Дисциплина «Информатика и программирование»**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**На тему: «Разработка игровой программы с использованием управляемых объектов»**

**Выполнил:**

Студент группы ПИ 1-1

Остапенко Д. В.

**Научный руководитель:**

Доцент Горелов С. В.

Москва 2015

Оглавление

[**Введение** 3](#_Toc417941457)

[*Цель игры* 3](#_Toc417941458)

[**1.Разработка информационной модели предметной области** 4](#_Toc417941459)

[*1.1 Анализ предметной области* 4](#_Toc417941460)

[*1.2 Варианты использования* 4](#_Toc417941461)

[**2. Проектирование** 5](#_Toc417941462)

[*2.1 Проектирование структуры программы.* 5](#_Toc417941463)

[*2.2 Проектирование интерфейса пользователя.* 6](#_Toc417941464)

[*2.3 Описание классов, методов и переменных.* 7](#_Toc417941465)

[*2.4 Основные алгоритмы приложения* 10](#_Toc417941466)

[**3. Программная реализация проекта** 14](#_Toc417941467)

[**Заключение** 16](#_Toc417941468)

[**Приложение 1. Програмный код** 17](#_Toc417941469)

# **Введение**

## Цель данной работы заключается в освоении основ работы с Windows Forms в среде разработки Visual Studio на языке высокого уровня С#. А также в закреплении навыков работы с формами, классами, объектами и методами.

## Язык высокого уровня С# - один из самых актуальных языков программирования в наше время, а сфера развлечений расширяется с каждым днем, следовательно, работу по созданию игровой программы также можно считать актуальной.

## *Цель игры*

В качестве прототипа для создания игры была взята широко известная игра Pac-man.

Pac-Man — компьютерная игра в жанре аркады, впервые вышедшая в 1980 году в Японии. Суть игры заключается в том, чтобы съесть все шарики и не умереть от постоянно движущихся приведений. При столкновении с любым из приведений герой погибает, и игра заканчивается.

# **1.Разработка информационной модели предметной области**

## *1.1 Анализ предметной области*

В данной программе автоматически происходит проверка коллизий(стенок) для каждого из движушихся объектов, атоматическое движение объектов(приведений), так же автоматическая обработка столкновений.

## *1.2 Варианты использования*

Программа должна проверять находятся ли рядом с главным объектом(пакманом) стены, шарики и приведения. Так же должна проверять наличие главного героя и стен рядом с каждым из приведений.

Программа должна автоматически передвигать приведений и передвигать главного героя при нажатии опеределенных клавиш.

Кроме того, выдавать информацию о том пройдена игра, либо же главный герой погиб.

# **2. Проектирование**

## *2.1 Проектирование структуры программы.*

*Form2. Приветствие.*

* button1. Кнопка, при нажатии на которую открывается вторая форма с игрой.

*Form1. Форма с игрой.*

* PictureBox1. Элемент, где происходят все игровые действия
* timer1. Таймер, используемый для движения приведений.

## *2.2 Проектирование интерфейса пользователя.*

|  |  |
| --- | --- |
| C:\Users\DemOst\Desktop\Снимок5.PNG  *Рис.1. Начальная форма при запуске игры* | *C:\Users\DemOst\Desktop\Снимок3.PNG*  *Рис.2. Интерфейс игры* |
| *C:\Users\DemOst\Desktop\Снимок1.PNG*  *Рис.3. MessageBox, если игра пройдена.* | *C:\Users\DemOst\Desktop\Снимок4.PNG*  *Рис.4. MessageBox, если столкнуться с приведением* |

## *2.3 Описание классов, методов и переменных.*

*Класс Form1.cs*

Переменные:

* public GameAppearance Game. Создание объекта класса.
* public FileInfo[] Content. Создания массива с файлами.
* public Random rndRed. Создание рандома для красного приведения.
* public Random rndPink. Создание рандома для розового приведения.
* public Random rndGreen Создание рандома для зеленого приведения.
* public Random rndOrange. Создание рандома для оранжевого приведения.

Методы обработки событий:

* Form1\_Load. При запуске формы запускает метод StartGame()
* StartGame().Создание директории изображений, запуск игры и запуск таймера.
* Form1\_KeyDown(). Передвижения пакмана связанные с нажатием на клавиши.
* timer1\_Tick. Таймер, созданный для вычисления рандомных чисел, изпользуемых для движения приведений.
* GameOver(). Метод для оповещения о проигрыше в игре и возможностью выбрать начать заново или закрыть игру.

*Класс Form2.cs*

Переменные:

* Form1 form1 = new Form1(). Создание объекта класса 1 формы.

Методы обработки событий:

* button1\_Click. Запускает Form1.

*Класс GameApperance.cs*

Все методы и функции всех объектов игры я поместил в 1 класс, для того чтобы было проще использовать их в классе Form1.cs, с помощью 1 объекта класса GameApperance Game.

Публичные переменные:

* public int pacmanPositionX. Позиция Пакмана (X)
* public int pacmanPositionY. Позиция Пакмана (Y)
* public int RedPositionX. Позиция Красного Привидения (X)
* public int RedPositionY. Позиция Красного приведения (Y)
* аналогичные переменные для розового, зеленого и оранжевого приведения
* public int[,] GameObjects. Массив объектов (0-еда, 1-препятствие, 2-пакман, 3-Красное приведение, 4-Розовое, 5-Зеленое, 6-Оранжевое)

Приватные переменные:

* private PictureBox pictureBox1. Объект, на котором будет отображаться игра
* private FileInfo[] Content. Массив используемоего контента

Конструктор класса:

* public GameAppearance(PictureBox pictureBox1, FileInfo[] Content)

Метод для загрузки файлов и вывода их на экран.

* public void Apperance(). Создавалась область Bitmap, заполнение массива файлами

Функция для проверки есть ли еда (шарики) в массиве.

* public bool IsMeal()

Функции проверки для пакмана

* public bool IsMealNear(). Есть ли рядом еда(шарики).
* public bool IsWallBottom(). Обработка коллизий. Есть ли снизу стена.
* Аналогично для сверху, слева, справа
* public bool IsRedGhostBottom().Проверяет есть ли красное приведение снизу.
* Аналогично для сверху, слева, справа. Аналогично для каждого другого приведения

Методы движения пакмана

* public void MoveDown(). Метод для передвижения пакмана вниз
* Аналогично для вверх, влево, вправо
* public void EatDown(). На случай если рядом в пакманом еда, реализован этот метод
* Аналогично для вверх, влево, вправо

Функции проверки приведений

* public bool IsWallBottomRG(). Обработка коллизий, проверяет, есть ли снизу стена
* Аналогично для сверху, слева, справа
* public bool IsPackmanBottomRG(). Проверяет есть ли пакман снизу.
* Аналогично для сверху, слева, справа
* public bool IsPinkGhostBottomRG(). Проверяет есть ли снизу розовое приведение
* Аналогично для сверху, слева, справа. Аналогично для остальных приведений
* Те же функции проверки прописаны для каждого из приведений

Методы для движения приведений

* int ObjRed = 7. Создается переменная, запоминающая, что было впереди приведения, для каждого приведения своя переменная.
* public void MoveDownRG(). Метод для передвижения красного приведения вниз.
* Аналогично для вверх, влево, вправо
* Аналогично создается для каждого приведения переменная и методы предвижения.

## *2.4 Основные алгоритмы приложения*

*public bool IsWallBottom().*

Данная функция проверяет, если снизу пакмана(объект с номером 2), объекта стена, то есть объекта с номером 1. Возвращает, если данный объекта там имеется.

public bool IsWallBottom()

{

return (GameObjects[pacmanPositionX, pacmanPositionY + 1] == 1);

}

*public void MoveDownRG().*

Данный метод передвигает красное приведение (2) на место объекта, который находил снизу него. Для этого создается переменная, которая изначально пустая, но по мере движения запоминает что находится впереди, и, когда приведение уходит со своей текущей позиции прорисовывает то, что было на месте, которое занимало приведение.

int ObjRed = 7;

public void MoveDownRG()

{

GameObjects[RedPositionX, RedPositionY] = ObjRed;

ObjRed = GameObjects[RedPositionX, RedPositionY + 1];

GameObjects[RedPositionX, RedPositionY + 1]=3;

RedPositionY += 1;

}

*private void timer1\_Tick(object sender, EventArgs e)*

Данный метод используется на каждый тик таймера. Для каждого приведения создается рандом, на каждый тик задается переменной число, которое влияет на последующее передвижение приведения.

Часть кода:

//Создание объекта рандом для красного приведения

rndRed = new Random();

//Метод для рандомизации значений

int r = rndRed.Next(4);

//Ветвления, зависящие от значий а

switch (r)

{

//При значении 0

case 0:

{

//Если нет снизу стены или приведений

if (!Game.IsWallBottomRG() && !Game.IsPinkGhostBottomRG() &&

!Game.IsGreenGhostBottomRG() && !Game.IsOrangeGhostBottomRG())

{

//Если снизу пакман снизу-Game over

if (Game.IsPackmanBottomRG())

{

GameOver();

}

else

{

//Передвижение красного приведения вниз

Game.MoveDownRG();

Game.Apperance();

}

}

break;

}

//При заначении 1

case 1:

{

//Если нет сверху стены и приведений

if (!Game.IsWallAboveRG()&&!Game.IsPinkGhostAboveRG() &&

!Game.IsGreenGhostAboveRG() && !Game.IsOrangeGhostAboveRG())

{

//если пакман сверху - game over

if (Game.IsPackmanAboveRG())

GameOver();

else

{

//Передвижение красного приведения вверх

Game.MoveUpRG();

Game.Apperance();

}

}

break;

}

//При значении 2

case 2:

{

//Если нет слева стены и приведений

if (!Game.IsWallLeftRG() && !Game.IsPinkGhostLeftRG() &&

!Game.IsGreenGhostLeftRG() && !Game.IsOrangeGhostLeftRG())

{

//Если Пакман слева - gameover

if (Game.IsPackmanLeftRG())

GameOver();

else

{

//Передвижение красного приведения влево

Game.MoveLeftRG();

Game.Apperance();

}

}

break;

}

//При значении 3

case 3:

{

//Если нет справа стены и приведений

if (!Game.IsWallRightRG() && !Game.IsPinkGhostRightRG() &&

!Game.IsGreenGhostRightRG() && !Game.IsOrangeGhostRightRG())

{

//Если справа пакман - gameover

if (Game.IsPackmanRightRG())

GameOver();

else

{

//Передвижение красного приведения вниз

Game.MoveRightRG();

Game.Apperance();

}

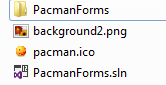
}

break;

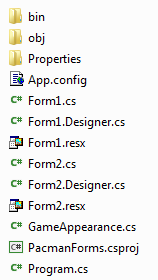
}

# **3. Программная реализация проекта**

Папка PacmanForms:

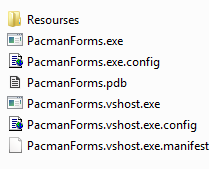


Папка PacmanForms:



Папка bin

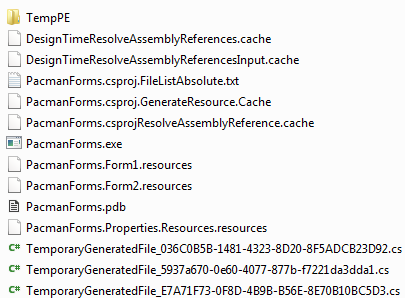
Папка Debug



Папка Realease пустая

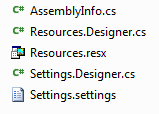
Папка obj

Папка Debug



Папка TempPE пустая

Папка Properties



# **Заключение**

В данном прототипе игры удалось добиться взаимодействия между объектами, взаимодействия между формами, выгрузкой файлами и прорисовкой файла на экране, так же рандомное движение объектов.

Т.к. данная игра является лишь прототип полной игры Pac-man хотелось бы добавить возможность съедать приведений, сделать счет в игре, дополнительные жизни, сделать появление объекта, который давал бы больше всего очков и перенести данную игру на XNA т.к. WindowsForms не может дать ту техническую составляющую, которая требуется для настоящей игры.

# **Приложение 1. Програмный код**

**Form2.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace PacmanForms

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

Form1 form1 = new Form1();

private void button1\_Click(object sender, EventArgs e)

{

this.Visible = false;

form1.ShowDialog();

this.Visible = true;

}

private void button2\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

**GameApperance.cs**

#region Using

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

#endregion

namespace PacmanForms

{

public class GameAppearance

{

#region Public Variables

#region Пакман

/// <summary>

/// Позиция Пакмана (X)

/// </summary>

public int pacmanPositionX;

/// <summary>

/// Позиция Пакмана (Y)

/// </summary>

public int pacmanPositionY;

#endregion

#region Привидения

/// <summary>

/// Позиция Красного Привидения (X)

/// </summary>

public int RedPositionX;

/// <summary>

/// Позиция Красного приведения (Y)

/// </summary>

public int RedPositionY;

/// <summary>

/// Позиция Розового Привидения (X)

/// </summary>

public int PinkPositionX;

/// <summary>

/// Позиция Розового приведения (Y)

/// </summary>

public int PinkPositionY;

/// <summary>

/// Позиция Зеленого Привидения (X)

/// </summary>

public int GreenPositionX;

/// <summary>

/// Позиция Зеленого приведения (Y)

/// </summary>

public int GreenPositionY;

/// <summary>

/// Позиция Оранжевого Привидения (X)

/// </summary>

public int OrangePositionX;

/// <summary>

/// Позиция Оранжевого приведения (Y)

/// </summary>

public int OrangePositionY;

#endregion

#region Объекты в игре

/// <summary>

/// Массив объектов (0-еда, 1-препятствие, 2-пакман,

/// 3-Красное приведение, 4-Розовое,

/// 5-Зеленое, 6-Оранжевое

/// </summary>

public int[,] GameObjects = { { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},

{ 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1},

{ 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1},

{ 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1},

{ 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1},

{ 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{ 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1},

{ 1, 1, 0, 1, 6, 5, 4, 3, 0, 1, 1},

{ 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1},

{ 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{ 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1},

{ 1, 0, 1, 1, 0, 2, 0, 1, 1, 0, 1},

{ 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1},

{ 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1},

{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},

};

#endregion

#endregion

#region Private Variables

/// <summary>

/// Объект, на котором будет отображаться игра

/// </summary>

private PictureBox pictureBox1;

/// <summary>

/// Массив используемоего контента

/// </summary>

private FileInfo[] Content;

#endregion

#region Constructor

/// <summary>

/// Конструктор Внешнего вида в игре

/// </summary>

/// <param name="pictureBox1">Управляющий компоент, в котором будет отображена игра</param>

/// <param name="Content">Используемые данные</param>

public GameAppearance(PictureBox pictureBox1, FileInfo[] Content)

{

this.pictureBox1 = pictureBox1;

this.Content = Content;

pacmanPositionX = 11;

pacmanPositionY = 5;

RedPositionX = 7;

RedPositionY = 7;

PinkPositionX = 7;

PinkPositionY = 6;

GreenPositionX = 7;

GreenPositionY = 5;

OrangePositionX = 7;

OrangePositionY = 4;

}

#endregion

#region Вывод объектов на экран

/// <summary>

/// Появление объектов на экране

/// </summary>

public void Apperance()

{

//Создание области в которую будем размещать объекты

Bitmap Images = new Bitmap(GameObjects.GetLength(0) \* 60, GameObjects.GetLength(1) \* 60);

Graphics e = Graphics.FromImage(Images);

e.Clear(SystemColors.AppWorkspace);

Image texture = null;

//Заполнение области объектами

if (Content != null)

{

FileInfo File = Content.First();

FileInfo Meal = Content.ElementAt(0);

FileInfo Block = Content.ElementAt(1);

FileInfo Pacman = Content.ElementAt(2);

FileInfo RedGhost = Content.ElementAt(3);

FileInfo PinkGhost = Content.ElementAt(4);

FileInfo GreenGhost = Content.ElementAt(5);

FileInfo OrangeGhost = Content.ElementAt(6);

FileInfo EmptySpace = Content.ElementAt(7);

//Проверка и выборка нужных объектов

if (Meal != null && Block != null && Pacman != null &&

RedGhost != null && PinkGhost != null && GreenGhost != null && OrangeGhost != null &&

EmptySpace!=null)

{

for (int x = 0; x < GameObjects.GetLength(0); x++)

{

for (int y = 0; y < GameObjects.GetLength(1); y++)

{

if (GameObjects[x, y] == 0)

texture = Image.FromFile(Meal.FullName);

if (GameObjects[x, y] == 1)

texture = Image.FromFile(Block.FullName);

if (GameObjects[x, y] == 2)

texture = Image.FromFile(Pacman.FullName);

if (GameObjects[x, y] == 3)

texture = Image.FromFile(RedGhost.FullName);

if (GameObjects[x, y] == 4)

texture = Image.FromFile(PinkGhost.FullName);

if (GameObjects[x, y] == 5)

texture = Image.FromFile(GreenGhost.FullName);

if (GameObjects[x, y] == 6)

texture = Image.FromFile(OrangeGhost.FullName);

if (GameObjects[x, y] == 7)

texture = Image.FromFile(EmptySpace.FullName);

e.DrawImage(texture, new Point(x \* 60, y \* 60));

}

}

pictureBox1.Image = Images;

}

}

}

#endregion

#region Проверка наличия еды

/// <summary>

/// Проверка наличия еды на экране

/// </summary>

/// <returns>Выводит, если еды нет</returns>

public bool IsMeal()

{

int z = 0;

for (int x = 0; x < GameObjects.GetLength(0); x++)

{

for (int y = 0; y < GameObjects.GetLength(1); y++)

{

//Если есть еда прибавляет значение

if (GameObjects[x, y] == 0)

z++;

}

}

return (z == 0);

}

#endregion

#region Проверяющие функции для Пакмана

#region Есть ли рядом еда (IsMealNear)

/// <summary>

/// Есть ли рядом еда

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsMealNear()

{

return (GameObjects[pacmanPositionX + 1, pacmanPositionY] == 0 ||

GameObjects[pacmanPositionX - 1, pacmanPositionY] == 0 ||

GameObjects[pacmanPositionX, pacmanPositionY + 1] == 0 ||

GameObjects[pacmanPositionX, pacmanPositionY - 1] == 0);

}

#endregion

#region Есть ли рядом стена Снизу(IsWallBottom)

/// <summary>

/// Проверяет есть ли стена снизу

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsWallBottom()

{

return (GameObjects[pacmanPositionX, pacmanPositionY + 1] == 1);

}

#endregion

#region Есть ли рядом стена Сверху (IsWallAbove)

/// <summary>

/// Проверяет есть ли стена сверху

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsWallAbove()

{

return (GameObjects[pacmanPositionX, pacmanPositionY - 1] == 1);

}

#endregion

#region Есть ли рядом стена Слева (IsWallLeft)

/// <summary>

/// Проверяет есть ли стена слева

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsWallLeft()

{

return (GameObjects[pacmanPositionX - 1, pacmanPositionY] == 1);

}

#endregion

#region Есть ли рядом стена Слева (IsWallRight)

/// <summary>

/// Проверяет есть ли стена справа

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsWallRight()

{

return (GameObjects[pacmanPositionX + 1, pacmanPositionY] == 1);

}

#endregion

#region Есть ли красное приведение Снизу(IsRedGhostBottom)

/// <summary>

/// Проверяет есть ли красное приведение снизу

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsRedGhostBottom()

{

return (GameObjects[pacmanPositionX, pacmanPositionY + 1] == 3);

}

#endregion

#region Есть ли красное приведение Сверху(IsRedGhostAbove)

/// <summary>

/// Проверяет есть ли красное приведение сверху

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsRedGhostAbove()

{

return (GameObjects[pacmanPositionX, pacmanPositionY - 1] == 3);

}

#endregion

#region Есть ли красное приведение Слева(IsRedGhostLeft)

/// <summary>

/// Проверяет есть ли красное приведение слева

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsRedGhostLeft()

{

return (GameObjects[pacmanPositionX-1, pacmanPositionY] == 3);

}

#endregion

#region Есть ли красное приведение Справа(IsRedGhostRight)

/// <summary>

/// Проверяет есть ли красное приведение справа

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsRedGhostRight()

{

return (GameObjects[pacmanPositionX + 1, pacmanPositionY] == 3);

}

#endregion

#region Есть ли розовое приведение Снизу(IsPinkGhostBottom)

/// <summary>

/// Проверяет есть ли розовое приведение снизу

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsPinkGhostBottom()

{

return (GameObjects[pacmanPositionX, pacmanPositionY + 1] == 4);

}

#endregion

#region Есть ли розовое приведение Сверху(IsPinkGhostAbove)

/// <summary>

/// Проверяет есть ли розовое приведение сверху

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsPinkGhostAbove()

{

return (GameObjects[pacmanPositionX, pacmanPositionY - 1] == 4);

}

#endregion

#region Есть ли розовое приведение Слева(IsPinkGhostLeft)

/// <summary>

/// Проверяет есть ли розовое приведение слева

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsPinkGhostLeft()

{

return (GameObjects[pacmanPositionX - 1, pacmanPositionY] == 4);

}

#endregion

#region Есть ли розовое приведение Справа(IsPinkGhostRight)

/// <summary>

/// Проверяет есть ли розовое приведение справа

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsPinkGhostRight()

{

return (GameObjects[pacmanPositionX + 1, pacmanPositionY] == 4);

}

#endregion

#region Есть ли зеленое приведение Снизу(IsGreenGhostBottom)

/// <summary>

/// Проверяет есть ли зеленое приведение снизу

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsGreenGhostBottom()

{

return (GameObjects[pacmanPositionX, pacmanPositionY + 1] == 5);

}

#endregion

#region Есть ли зеленое приведение Сверху(IsGreenGhostAbove)

/// <summary>

/// Проверяет есть ли зеленое приведение сверху

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsGreenGhostAbove()

{

return (GameObjects[pacmanPositionX, pacmanPositionY - 1] == 5);

}

#endregion

#region Есть ли зеленое приведение Слева(IsGreenGhostLeft)

/// <summary>

/// Проверяет есть ли зеленое приведение слева

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsGreenGhostLeft()

{

return (GameObjects[pacmanPositionX - 1, pacmanPositionY] == 5);

}

#endregion

#region Есть ли зеленое приведение Справа(IsGreenGhostRight)

/// <summary>

/// Проверяет есть ли зеленое приведение справа

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsGreenGhostRight()

{

return (GameObjects[pacmanPositionX + 1, pacmanPositionY] == 5);

}

#endregion

#region Есть ли оранжевое приведение Снизу(IsOrangeGhostBottom)

/// <summary>

/// Проверяет есть ли оранжевое приведение снизу

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsOrangeGhostBottom()

{

return (GameObjects[pacmanPositionX, pacmanPositionY + 1] == 6);

}

#endregion

#region Есть ли оранжевое приведение Сверху(IsOrangeGhostAbove)

/// <summary>

/// Проверяет есть ли оранжевое приведение сверху

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsOrangeGhostAbove()

{

return (GameObjects[pacmanPositionX, pacmanPositionY - 1] == 6);

}

#endregion

#region Есть ли оранжевое приведение Слева(IsOrangeGhostLeft)

/// <summary>

/// Проверяет есть ли оранжевое приведение слева

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsOrangeGhostLeft()

{

return (GameObjects[pacmanPositionX - 1, pacmanPositionY] == 6);

}

#endregion

#region Есть ли оранжевое приведение Справа(IsOrangeGhostRight)

/// <summary>

/// Проверяет есть ли оранжевое приведение справа

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsOrangeGhostRight()

{

return (GameObjects[pacmanPositionX + 1, pacmanPositionY] == 6);

}

#endregion

#endregion

#region Движение Пакмана

#region MoveDown

/// <summary>

/// Движение вниз

/// </summary>

public void MoveDown()

{

int NewPosition = GameObjects[pacmanPositionX, pacmanPositionY + 1];

GameObjects[pacmanPositionX, pacmanPositionY + 1] = GameObjects[pacmanPositionX, pacmanPositionY];

GameObjects[pacmanPositionX, pacmanPositionY] = NewPosition;

pacmanPositionY += 1;

}

#endregion

#region MoveUp

/// <summary>

/// Движение вверх

/// </summary>

public void MoveUp()

{

int NewPosition = GameObjects[pacmanPositionX, pacmanPositionY - 1];

GameObjects[pacmanPositionX, pacmanPositionY - 1] = GameObjects[pacmanPositionX, pacmanPositionY];

GameObjects[pacmanPositionX, pacmanPositionY] = NewPosition;

pacmanPositionY -= 1;

}

#endregion

#region MoveLeft

/// <summary>

/// Движение Влево

/// </summary>

public void MoveLeft()

{

int NewPosition = GameObjects[pacmanPositionX - 1, pacmanPositionY];

GameObjects[pacmanPositionX - 1, pacmanPositionY] = GameObjects[pacmanPositionX, pacmanPositionY];

GameObjects[pacmanPositionX, pacmanPositionY] = NewPosition;

pacmanPositionX -= 1;

}

#endregion

#region MoveRight

/// <summary>

/// Движение Вправо

/// </summary>

public void MoveRight()

{

int NewPosition = GameObjects[pacmanPositionX + 1, pacmanPositionY];

GameObjects[pacmanPositionX + 1, pacmanPositionY] = GameObjects[pacmanPositionX, pacmanPositionY];

GameObjects[pacmanPositionX, pacmanPositionY] = NewPosition;

pacmanPositionX += 1;

}

#endregion

#endregion

#region Eat

#region EatDown

/// <summary>

/// Съесть шарик

/// </summary>

public void EatDown()

{

int NewPosition = GameObjects[pacmanPositionX, pacmanPositionY+1];

GameObjects[pacmanPositionX, pacmanPositionY+1] = GameObjects[pacmanPositionX, pacmanPositionY];

NewPosition = 7;

GameObjects[pacmanPositionX, pacmanPositionY] = NewPosition;

pacmanPositionY += 1;

}

#endregion

#region EatUp

/// <summary>

/// Съесть шарик

/// </summary>

public void EatUp()

{

int NewPosition = GameObjects[pacmanPositionX, pacmanPositionY-1];

GameObjects[pacmanPositionX, pacmanPositionY-1] = GameObjects[pacmanPositionX, pacmanPositionY];

NewPosition = 7;

GameObjects[pacmanPositionX, pacmanPositionY] = NewPosition;

pacmanPositionY -= 1;

}

#endregion

#region EatLeft

/// <summary>

/// Съесть шарик

/// </summary>

public void EatLeft()

{

int NewPosition = GameObjects[pacmanPositionX - 1, pacmanPositionY];

GameObjects[pacmanPositionX - 1, pacmanPositionY] = GameObjects[pacmanPositionX, pacmanPositionY];

NewPosition = 7;

GameObjects[pacmanPositionX, pacmanPositionY] = NewPosition;

pacmanPositionX -= 1;

}

#endregion

#region EatRight

/// <summary>

/// Съесть шарик

/// </summary>

public void EatRight()

{

int NewPosition = GameObjects[pacmanPositionX + 1, pacmanPositionY];

GameObjects[pacmanPositionX + 1, pacmanPositionY] = GameObjects[pacmanPositionX, pacmanPositionY];

NewPosition = 7;

GameObjects[pacmanPositionX, pacmanPositionY] = NewPosition;

pacmanPositionX += 1;

}

#endregion

#endregion

#region Функции проверки для приведений

#region Красное

#region IsWallBottomRG

/// <summary>

/// Проверяет, есть ли снизу стена

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsWallBottomRG()

{

return (GameObjects[RedPositionX, RedPositionY + 1] == 1);

}

#endregion

#region IsWallAboveRG

/// <summary>

/// Проверяет, есть ли сверху стена

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsWallAboveRG()

{

return (GameObjects[RedPositionX, RedPositionY - 1] == 1);

}

#endregion

#region IsWallLeftRG

/// <summary>

/// Проверяет, есть ли снизу стена

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsWallLeftRG()

{

return (GameObjects[RedPositionX - 1, RedPositionY] == 1);

}

#endregion

#region IsWallRightRG

/// <summary>

/// Проверяет, есть ли справа стена

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsWallRightRG()

{

return (GameObjects[RedPositionX + 1, RedPositionY] == 1);

}

#endregion

#region IsPacmanBottom

/// <summary>

/// Проверка есть ли снизу пакман

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsPackmanBottomRG()

{

return (GameObjects[RedPositionX, RedPositionY + 1] == 2);

}

#endregion

#region IsPacmanAbove

/// <summary>

/// Проверка есть ли сверху пакман

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsPackmanAboveRG()

{

return (GameObjects[RedPositionX, RedPositionY - 1] == 2);

}

#endregion

#region IsPacmanLeft

/// <summary>

/// Проверка есть ли слева пакман

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsPackmanLeftRG()

{

return (GameObjects[RedPositionX-1, RedPositionY] == 2);

}

#endregion

#region IsPacmanRight

/// <summary>

/// Проверка есть ли справа пакман

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsPackmanRightRG()

{

return (GameObjects[RedPositionX+1, RedPositionY] == 2);

}

#endregion

#region Есть ли розовое приведение Снизу(IsPinkGhostBottomRG)

/// <summary>

/// Проверяет есть ли розовое приведение снизу

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsPinkGhostBottomRG()

{

return (GameObjects[RedPositionX, RedPositionY + 1] == 4);

}

#endregion

#region Есть ли розовое приведение Сверху(IsPinkGhostAboveRG)

/// <summary>

/// Проверяет есть ли розовое приведение сверху

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsPinkGhostAboveRG()

{

return (GameObjects[RedPositionX, RedPositionY - 1] == 4);

}

#endregion

#region Есть ли розовое приведение Слева(IsPinkGhostLeftRG)

/// <summary>

/// Проверяет есть ли розовое приведение слева

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsPinkGhostLeftRG()

{

return (GameObjects[RedPositionX - 1, RedPositionY] == 4);

}

#endregion

#region Есть ли розовое приведение Справа(IsPinkGhostRightRG)

/// <summary>

/// Проверяет есть ли розовое приведение справа

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsPinkGhostRightRG()

{

return (GameObjects[RedPositionX + 1, RedPositionY] == 4);

}

#endregion

#region Есть ли зеленое приведение Снизу(IsGreenGhostBottomRG)

/// <summary>

/// Проверяет есть ли зеленое приведение снизу

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsGreenGhostBottomRG()

{

return (GameObjects[RedPositionX, RedPositionY + 1] == 5);

}

#endregion

#region Есть ли зеленое приведение Сверху(IsGreenGhostAboveRG)

/// <summary>

/// Проверяет есть ли зеленое приведение сверху

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsGreenGhostAboveRG()

{

return (GameObjects[RedPositionX, RedPositionY - 1] == 5);

}

#endregion

#region Есть ли зеленое приведение Слева(IsGreenGhostLeftRG)

/// <summary>

/// Проверяет есть ли зеленое приведение слева

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsGreenGhostLeftRG()

{

return (GameObjects[RedPositionX - 1, RedPositionY] == 5);

}

#endregion

#region Есть ли зеленое приведение Справа(IsGreenGhostRightRG)

/// <summary>

/// Проверяет есть ли зеленое приведение справа

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsGreenGhostRightRG()

{

return (GameObjects[RedPositionX + 1, RedPositionY] == 5);

}

#endregion

#region Есть ли оранжевое приведение Снизу(IsOrangeGhostBottomRG)

/// <summary>

/// Проверяет есть ли оранжевое приведение снизу

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsOrangeGhostBottomRG()

{

return (GameObjects[RedPositionX, RedPositionY + 1] == 6);

}

#endregion

#region Есть ли оранжевое приведение Сверху(IsOrangeGhostAboveRG)

/// <summary>

/// Проверяет есть ли оранжевое приведение сверху

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsOrangeGhostAboveRG()

{

return (GameObjects[RedPositionX, RedPositionY - 1] == 6);

}

#endregion

#region Есть ли оранжевое приведение Слева(IsOrangeGhostLeftRG)

/// <summary>

/// Проверяет есть ли оранжевое приведение слева

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsOrangeGhostLeftRG()

{

return (GameObjects[RedPositionX - 1, RedPositionY] == 6);

}

#endregion

#region Есть ли оранжевое приведение Справа(IsOrangeGhostRightRG)

/// <summary>

/// Проверяет есть ли оранжевое приведение справа

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsOrangeGhostRightRG()

{

return (GameObjects[RedPositionX + 1, RedPositionY] == 6);

}

#endregion

#region Есть ли рядом еда (IsMealNearRG)

/// <summary>

/// Есть ли рядом еда

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsMealNearRG()

{

return (GameObjects[RedPositionX + 1, RedPositionY] == 0 ||

GameObjects[RedPositionX - 1, RedPositionY] == 0 ||

GameObjects[RedPositionX, RedPositionY + 1] == 0 ||

GameObjects[RedPositionX, RedPositionY - 1] == 0);

}

#endregion

#endregion

#region Розовое

#region IsWallBottomPG

/// <summary>

/// Проверяет, есть ли снизу стена

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsWallBottomPG()

{

return (GameObjects[PinkPositionX, PinkPositionY + 1] == 1);

}

#endregion

#region IsWallAbovePG

/// <summary>

/// Проверяет, есть ли сверху стена

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsWallAbovePG()

{

return (GameObjects[PinkPositionX, PinkPositionY - 1] == 1);

}

#endregion

#region IsWallLeftPG

/// <summary>

/// Проверяет, есть ли снизу стена

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsWallLeftPG()

{

return (GameObjects[PinkPositionX - 1, PinkPositionY] == 1);

}

#endregion

#region IsWallRightPG

/// <summary>

/// Проверяет, есть ли справа стена

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsWallRightPG()

{

return (GameObjects[PinkPositionX + 1, PinkPositionY] == 1);

}

#endregion

#region IsPacmanBottom

/// <summary>

/// Проверка есть ли снизу пакман

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsPackmanBottomPG()

{

return (GameObjects[PinkPositionX, PinkPositionY + 1] == 2);

}

#endregion

#region IsPacmanAbove

/// <summary>

/// Проверка есть ли сверху пакман

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsPackmanAbovePG()

{

return (GameObjects[PinkPositionX, PinkPositionY - 1] == 2);

}

#endregion

#region IsPacmanLeft

/// <summary>

/// Проверка есть ли слева пакман

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsPackmanLeftPG()

{

return (GameObjects[PinkPositionX - 1, PinkPositionY] == 2);

}

#endregion

#region IsPacmanRight

/// <summary>

/// Проверка есть ли справа пакман

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsPackmanRightPG()

{

return (GameObjects[PinkPositionX + 1, PinkPositionY] == 2);

}

#endregion

#region Есть ли красное приведение Снизу(IsRedGhostBottomPG)

/// <summary>

/// Проверяет есть ли красное приведение снизу

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsRedGhostBottomPG()

{

return (GameObjects[PinkPositionX, PinkPositionY + 1] == 3);

}

#endregion

#region Есть ли красное приведение Сверху(IsRedGhostAbovePG)

/// <summary>

/// Проверяет есть ли красное приведение сверху

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsRedGhostAbovePG()

{

return (GameObjects[PinkPositionX, PinkPositionY - 1] == 3);

}

#endregion

#region Есть ли красное приведение Слева(IsRedGhostLeftPG)

/// <summary>

/// Проверяет есть ли красное приведение слева

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsRedGhostLeftPG()

{

return (GameObjects[PinkPositionX - 1, PinkPositionY] == 3);

}

#endregion

#region Есть ли красное приведение Справа(IsRedGhostRightPG)

/// <summary>

/// Проверяет есть ли красное приведение справа

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsRedGhostRightPG()

{

return (GameObjects[PinkPositionX + 1, PinkPositionY] == 3);

}

#endregion

#region Есть ли зеленое приведение Снизу(IsGreenGhostBottomPG)

/// <summary>

/// Проверяет есть ли зеленое приведение снизу

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsGreenGhostBottomPG()

{

return (GameObjects[PinkPositionX, PinkPositionY + 1] == 5);

}

#endregion

#region Есть ли зеленое приведение Сверху(IsGreenGhostAbovePG)

/// <summary>

/// Проверяет есть ли зеленое приведение сверху

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsGreenGhostAbovePG()

{

return (GameObjects[PinkPositionX, PinkPositionY - 1] == 5);

}

#endregion

#region Есть ли зеленое приведение Слева(IsGreenGhostLeftPG)

/// <summary>

/// Проверяет есть ли зеленое приведение слева

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsGreenGhostLeftPG()

{

return (GameObjects[PinkPositionX - 1, PinkPositionY] == 5);

}

#endregion

#region Есть ли зеленое приведение Справа(IsGreenGhostRightPG)

/// <summary>

/// Проверяет есть ли зеленое приведение справа

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsGreenGhostRightPG()

{

return (GameObjects[PinkPositionX + 1, PinkPositionY] == 5);

}

#endregion

#region Есть ли оранжевое приведение Снизу(IsOrangeGhostBottomPG)

/// <summary>

/// Проверяет есть ли оранжевое приведение снизу

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsOrangeGhostBottomPG()

{

return (GameObjects[PinkPositionX, PinkPositionY + 1] == 6);

}

#endregion

#region Есть ли оранжевое приведение Сверху(IsOrangeGhostAbovePG)

/// <summary>

/// Проверяет есть ли оранжевое приведение сверху

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsOrangeGhostAbovePG()

{

return (GameObjects[PinkPositionX, PinkPositionY - 1] == 6);

}

#endregion

#region Есть ли оранжевое приведение Слева(IsOrangeGhostLeftPG)

/// <summary>

/// Проверяет есть ли оранжевое приведение слева

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsOrangeGhostLeftPG()

{

return (GameObjects[PinkPositionX - 1, PinkPositionY] == 6);

}

#endregion

#region Есть ли оранжевое приведение Справа(IsOrangeGhostRightPG)

/// <summary>

/// Проверяет есть ли оранжевое приведение справа

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsOrangeGhostRightPG()

{

return (GameObjects[PinkPositionX + 1, PinkPositionY] == 6);

}

#endregion

#region Есть ли рядом еда (IsMealNearPG)

/// <summary>

/// Есть ли рядом еда

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsMealNearPG()

{

return (GameObjects[PinkPositionX + 1, PinkPositionY] == 0 ||

GameObjects[PinkPositionX - 1, PinkPositionY] == 0 ||

GameObjects[PinkPositionX, PinkPositionY + 1] == 0 ||

GameObjects[PinkPositionX, PinkPositionY - 1] == 0);

}

#endregion

#endregion

#region Зеленое

#region IsWallBottomGG

/// <summary>

/// Проверяет, есть ли снизу стена

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsWallBottomGG()

{

return (GameObjects[GreenPositionX, GreenPositionY + 1] == 1);

}

#endregion

#region IsWallAboveGG

/// <summary>

/// Проверяет, есть ли сверху стена

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsWallAboveGG()

{

return (GameObjects[GreenPositionX, GreenPositionY - 1] == 1);

}

#endregion

#region IsWallLeftGG

/// <summary>

/// Проверяет, есть ли снизу стена

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsWallLeftGG()

{

return (GameObjects[GreenPositionX - 1, GreenPositionY] == 1);

}

#endregion

#region IsWallRightGG

/// <summary>

/// Проверяет, есть ли справа стена

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsWallRightGG()

{

return (GameObjects[GreenPositionX + 1, GreenPositionY] == 1);

}

#endregion

#region IsPacmanBottom

/// <summary>

/// Проверка есть ли снизу пакман

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsPackmanBottomGG()

{

return (GameObjects[GreenPositionX, GreenPositionY + 1] == 2);

}

#endregion

#region IsPacmanAbove

/// <summary>

/// Проверка есть ли сверху пакман

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsPackmanAboveGG()

{

return (GameObjects[GreenPositionX, GreenPositionY - 1] == 2);

}

#endregion

#region IsPacmanLeft

/// <summary>

/// Проверка есть ли слева пакман

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsPackmanLeftGG()

{

return (GameObjects[GreenPositionX - 1, GreenPositionY] == 2);

}

#endregion

#region IsPacmanRight

/// <summary>

/// Проверка есть ли справа пакман

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsPackmanRightGG()

{

return (GameObjects[GreenPositionX + 1, GreenPositionY] == 2);

}

#endregion

#region Есть ли красное приведение Снизу(IsRedGhostBottomGG)

/// <summary>

/// Проверяет есть ли красное приведение снизу

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsRedGhostBottomGG()

{

return (GameObjects[GreenPositionX, GreenPositionY + 1] == 3);

}

#endregion

#region Есть ли красное приведение Сверху(IsRedGhostAboveGG)

/// <summary>

/// Проверяет есть ли красное приведение сверху

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsRedGhostAboveGG()

{

return (GameObjects[GreenPositionX, GreenPositionY - 1] == 3);

}

#endregion

#region Есть ли красное приведение Слева(IsRedGhostLeftGG)

/// <summary>

/// Проверяет есть ли красное приведение слева

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsRedGhostLeftGG()

{

return (GameObjects[GreenPositionX - 1, GreenPositionY] == 3);

}

#endregion

#region Есть ли красное приведение Справа(IsRedGhostRightGG)

/// <summary>

/// Проверяет есть ли красное приведение справа

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsRedGhostRightGG()

{

return (GameObjects[GreenPositionX + 1, GreenPositionY] == 3);

}

#endregion

#region Есть ли розовое приведение Снизу(IsPinkGhostBottomGG)

/// <summary>

/// Проверяет есть ли розовое приведение снизу

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsPinkGhostBottomGG()

{

return (GameObjects[GreenPositionX, GreenPositionY + 1] == 4);

}

#endregion

#region Есть ли розовое приведение Сверху(IsPinkGhostAboveGG)

/// <summary>

/// Проверяет есть ли розовое приведение сверху

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsPinkGhostAboveGG()

{

return (GameObjects[GreenPositionX, GreenPositionY - 1] == 4);

}

#endregion

#region Есть ли розовое приведение Слева(IsPinkGhostLeftGG)

/// <summary>

/// Проверяет есть ли розовое приведение слева

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsPinkGhostLeftGG()

{

return (GameObjects[GreenPositionX - 1, GreenPositionY] == 4);

}

#endregion

#region Есть ли розовое приведение Справа(IsPinkGhostRightGG)

/// <summary>

/// Проверяет есть ли розовое приведение справа

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsPinkGhostRightGG()

{

return (GameObjects[GreenPositionX + 1, GreenPositionY] == 4);

}

#endregion

#region Есть ли оранжевое приведение Снизу(IsOrangeGhostBottomGG)

/// <summary>

/// Проверяет есть ли оранжевое приведение снизу

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsOrangeGhostBottomGG()

{

return (GameObjects[GreenPositionX, GreenPositionY + 1] == 6);

}

#endregion

#region Есть ли оранжевое приведение Сверху(IsOrangeGhostAboveGG)

/// <summary>

/// Проверяет есть ли оранжевое приведение сверху

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsOrangeGhostAboveGG()

{

return (GameObjects[GreenPositionX, GreenPositionY - 1] == 6);

}

#endregion

#region Есть ли оранжевое приведение Слева(IsOrangeGhostLeftGG)

/// <summary>

/// Проверяет есть ли оранжевое приведение слева

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsOrangeGhostLeftGG()

{

return (GameObjects[GreenPositionX - 1, GreenPositionY] == 6);

}

#endregion

#region Есть ли оранжевое приведение Справа(IsOrangeGhostRightGG)

/// <summary>

/// Проверяет есть ли оранжевое приведение справа

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsOrangeGhostRightGG()

{

return (GameObjects[GreenPositionX + 1, GreenPositionY] == 6);

}

#endregion

#region Есть ли рядом еда (IsMealNearGG)

/// <summary>

/// Есть ли рядом еда

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsMealNearGG()

{

return (GameObjects[GreenPositionX + 1, GreenPositionY] == 0 ||

GameObjects[GreenPositionX - 1, GreenPositionY] == 0 ||

GameObjects[GreenPositionX, GreenPositionY + 1] == 0 ||

GameObjects[GreenPositionX, GreenPositionY - 1] == 0);

}

#endregion

#endregion

#region Оранжевое

#region IsWallBottomOG

/// <summary>

/// Проверяет, есть ли снизу стена

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsWallBottomOG()

{

return (GameObjects[OrangePositionX, OrangePositionY + 1] == 1);

}

#endregion

#region IsWallAboveOG

/// <summary>

/// Проверяет, есть ли сверху стена

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsWallAboveOG()

{

return (GameObjects[OrangePositionX, OrangePositionY - 1] == 1);

}

#endregion

#region IsWallLeftOG

/// <summary>

/// Проверяет, есть ли снизу стена

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsWallLeftOG()

{

return (GameObjects[OrangePositionX - 1, OrangePositionY] == 1);

}

#endregion

#region IsWallRightOG

/// <summary>

/// Проверяет, есть ли справа стена

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsWallRightOG()

{

return (GameObjects[OrangePositionX + 1, OrangePositionY] == 1);

}

#endregion

#region IsPacmanBottom

/// <summary>

/// Проверка есть ли снизу пакман

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsPackmanBottomOG()

{

return (GameObjects[OrangePositionX, OrangePositionY + 1] == 2);

}

#endregion

#region IsPacmanAbove

/// <summary>

/// Проверка есть ли сверху пакман

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsPackmanAboveOG()

{

return (GameObjects[OrangePositionX, OrangePositionY - 1] == 2);

}

#endregion

#region IsPacmanLeft

/// <summary>

/// Проверка есть ли слева пакман

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsPackmanLeftOG()

{

return (GameObjects[OrangePositionX - 1, OrangePositionY] == 2);

}

#endregion

#region IsPacmanRight

/// <summary>

/// Проверка есть ли справа пакман

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsPackmanRightOG()

{

return (GameObjects[OrangePositionX + 1, OrangePositionY] == 2);

}

#endregion

#region Есть ли красное приведение Снизу(IsRedGhostBottomOG)

/// <summary>

/// Проверяет есть ли красное приведение снизу

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsRedGhostBottomOG()

{

return (GameObjects[OrangePositionX, OrangePositionY + 1] == 3);

}

#endregion

#region Есть ли красное приведение Сверху(IsRedGhostAboveOG)

/// <summary>

/// Проверяет есть ли красное приведение сверху

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsRedGhostAboveOG()

{

return (GameObjects[OrangePositionX, OrangePositionY - 1] == 3);

}

#endregion

#region Есть ли красное приведение Слева(IsRedGhostLeftOG)

/// <summary>

/// Проверяет есть ли красное приведение слева

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsRedGhostLeftOG()

{

return (GameObjects[OrangePositionX - 1, OrangePositionY] == 3);

}

#endregion

#region Есть ли красное приведение Справа(IsRedGhostRightOG)

/// <summary>

/// Проверяет есть ли красное приведение справа

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsRedGhostRightOG()

{

return (GameObjects[OrangePositionX + 1, OrangePositionY] == 3);

}

#endregion

#region Есть ли розовое приведение Снизу(IsPinkGhostBottomOG)

/// <summary>

/// Проверяет есть ли розовое приведение снизу

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsPinkGhostBottomOG()

{

return (GameObjects[OrangePositionX, OrangePositionY + 1] == 4);

}

#endregion

#region Есть ли розовое приведение Сверху(IsPinkGhostAboveOG)

/// <summary>

/// Проверяет есть ли розовое приведение сверху

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsPinkGhostAboveOG()

{

return (GameObjects[OrangePositionX, OrangePositionY - 1] == 4);

}

#endregion

#region Есть ли розовое приведение Слева(IsPinkGhostLeftOG)

/// <summary>

/// Проверяет есть ли розовое приведение слева

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsPinkGhostLeftOG()

{

return (GameObjects[OrangePositionX - 1, OrangePositionY] == 4);

}

#endregion

#region Есть ли розовое приведение Справа(IsPinkGhostRightOG)

/// <summary>

/// Проверяет есть ли розовое приведение справа

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsPinkGhostRightOG()

{

return (GameObjects[OrangePositionX + 1, OrangePositionY] == 4);

}

#endregion

#region Есть ли зеленое приведение Снизу(IsGreenGhostBottomOG)

/// <summary>

/// Проверяет есть ли зеленое приведение снизу

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsGreenGhostBottomOG()

{

return (GameObjects[OrangePositionX, OrangePositionY + 1] == 5);

}

#endregion

#region Есть ли зеленое приведение Сверху(IsGreenGhostAboveOG)

/// <summary>

/// Проверяет есть ли зеленое приведение сверху

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsGreenGhostAboveOG()

{

return (GameObjects[OrangePositionX, OrangePositionY - 1] == 5);

}

#endregion

#region Есть ли зеленое приведение Слева(IsGreenGhostLeftOG)

/// <summary>

/// Проверяет есть ли зеленое приведение слева

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsGreenGhostLeftOG()

{

return (GameObjects[OrangePositionX - 1, OrangePositionY] == 5);

}

#endregion

#region Есть ли зеленое приведение Справа(IsGreenGhostRightOG)

/// <summary>

/// Проверяет есть ли зеленое приведение справа

/// </summary>

/// <returns>Возвращает,если есть</returns>

public bool IsGreenGhostRightOG()

{

return (GameObjects[OrangePositionX + 1, OrangePositionY] == 5);

}

#endregion

#region Есть ли рядом еда (IsMealNearOG)

/// <summary>

/// Есть ли рядом еда

/// </summary>

/// <returns>Возвращает, если есть</returns>

public bool IsMealNearOG()

{

return (GameObjects[OrangePositionX + 1, OrangePositionY] == 0 ||

GameObjects[OrangePositionX - 1, OrangePositionY] == 0 ||

GameObjects[OrangePositionX, OrangePositionY + 1] == 0 ||

GameObjects[OrangePositionX, OrangePositionY - 1] == 0);

}

#endregion

#endregion

#endregion

#region Движение приведений

#region Красное

//Переменная для запоминания того, что было впереди

int ObjRed = 7;

#region MoveDownRG

/// <summary>

/// Движение вниз

/// </summary>

public void MoveDownRG()

{

GameObjects[RedPositionX, RedPositionY] = ObjRed;

ObjRed = GameObjects[RedPositionX, RedPositionY + 1];

GameObjects[RedPositionX, RedPositionY + 1]=3;

RedPositionY += 1;

}

#endregion

#region MoveUpRG

/// <summary>

/// Движение вверх

/// </summary>

public void MoveUpRG()

{

GameObjects[RedPositionX, RedPositionY] = ObjRed;

ObjRed = GameObjects[RedPositionX, RedPositionY - 1];

GameObjects[RedPositionX, RedPositionY - 1] = 3;

RedPositionY -= 1;

}

#endregion

#region MoveLeftRG

/// <summary>

/// Движение влево

/// </summary>

public void MoveLeftRG()

{

GameObjects[RedPositionX, RedPositionY] = ObjRed;

ObjRed = GameObjects[RedPositionX-1, RedPositionY];

GameObjects[RedPositionX-1, RedPositionY] = 3;

RedPositionX -= 1;

}

#endregion

#region MoveRightRG

/// <summary>

/// Движение вправо

/// </summary>

public void MoveRightRG()

{

GameObjects[RedPositionX, RedPositionY] = ObjRed;

ObjRed = GameObjects[RedPositionX+1, RedPositionY];

GameObjects[RedPositionX+1, RedPositionY] = 3;

RedPositionX += 1;

}

#endregion

#endregion

#region Розовое

//Переменная для запоминания того, что было впереди

int ObjPink = 7;

#region MoveDownPG

/// <summary>

/// Движение вниз

/// </summary>

public void MoveDownPG()

{

GameObjects[PinkPositionX, PinkPositionY] = ObjPink;

ObjPink = GameObjects[PinkPositionX, PinkPositionY + 1];

GameObjects[PinkPositionX, PinkPositionY + 1] = 4;

PinkPositionY += 1;

}

#endregion

#region MoveUpPG

/// <summary>

/// Движение вверх

/// </summary>

public void MoveUpPG()

{

GameObjects[PinkPositionX, PinkPositionY] = ObjPink;

ObjPink = GameObjects[PinkPositionX, PinkPositionY - 1];

GameObjects[PinkPositionX, PinkPositionY - 1] = 4;

PinkPositionY -= 1;

}

#endregion

#region MoveLeftPG

/// <summary>

/// Движение влево

/// </summary>

public void MoveLeftPG()

{

GameObjects[PinkPositionX, PinkPositionY] = ObjPink;

ObjPink = GameObjects[PinkPositionX - 1, PinkPositionY];

GameObjects[PinkPositionX - 1, PinkPositionY] = 4;

PinkPositionX -= 1;

}

#endregion

#region MoveRightPG

/// <summary>

/// Движение вправо

/// </summary>

public void MoveRightPG()

{

GameObjects[PinkPositionX, PinkPositionY] = ObjPink;

ObjPink = GameObjects[PinkPositionX + 1, PinkPositionY];

GameObjects[PinkPositionX+1, PinkPositionY] = 4;

PinkPositionX += 1;

}

#endregion

#endregion

#region Зеленое

//Переменная для запоминания того, что было впереди

int ObjGreen = 7;

#region MoveDownGG

/// <summary>

/// Движение вниз

/// </summary>

public void MoveDownGG()

{

GameObjects[GreenPositionX, GreenPositionY] = ObjGreen;

ObjGreen = GameObjects[GreenPositionX, GreenPositionY + 1];

GameObjects[GreenPositionX, GreenPositionY + 1] = 5;

GreenPositionY += 1;

}

#endregion

#region MoveUpGG

/// <summary>

/// Движение вверх

/// </summary>

public void MoveUpGG()

{

GameObjects[GreenPositionX, GreenPositionY] = ObjGreen;

ObjGreen = GameObjects[GreenPositionX, GreenPositionY - 1];

GameObjects[GreenPositionX, GreenPositionY - 1] = 5;

GreenPositionY -= 1;

}

#endregion

#region MoveLeftGG

/// <summary>

/// Движение влево

/// </summary>

public void MoveLeftGG()

{

GameObjects[GreenPositionX, GreenPositionY] = ObjGreen;

ObjGreen = GameObjects[GreenPositionX - 1, GreenPositionY];

GameObjects[GreenPositionX - 1, GreenPositionY] = 5;

GreenPositionX -= 1;

}

#endregion

#region MoveRightGG

/// <summary>

/// Движение вправо

/// </summary>

public void MoveRightGG()

{

GameObjects[GreenPositionX, GreenPositionY] = ObjGreen;

ObjGreen = GameObjects[GreenPositionX + 1, GreenPositionY];

GameObjects[GreenPositionX + 1, GreenPositionY] = 5;

GreenPositionX += 1;

}

#endregion

#endregion

#region Оранжевое

//Переменная для запоминания того, что было впереди

int ObjOrange = 7;

#region MoveDownOG

/// <summary>

/// Движение вниз

/// </summary>

public void MoveDownOG()

{

GameObjects[OrangePositionX, OrangePositionY] = ObjOrange;

ObjOrange = GameObjects[OrangePositionX, OrangePositionY + 1];

GameObjects[OrangePositionX, OrangePositionY + 1] = 6;

OrangePositionY += 1;

}

#endregion

#region MoveUpOG

/// <summary>

/// Движение вверх

/// </summary>

public void MoveUpOG()

{

GameObjects[OrangePositionX, OrangePositionY] = ObjOrange;

ObjOrange = GameObjects[OrangePositionX, OrangePositionY - 1];

GameObjects[OrangePositionX, OrangePositionY - 1] = 6;

OrangePositionY -= 1;

}

#endregion

#region MoveLeftOG

/// <summary>

/// Движение влево

/// </summary>

public void MoveLeftOG()

{

GameObjects[OrangePositionX, OrangePositionY] = ObjOrange;

ObjOrange = GameObjects[OrangePositionX - 1, OrangePositionY];

GameObjects[OrangePositionX - 1, OrangePositionY] = 6;

OrangePositionX -= 1;

}

#endregion

#region MoveRightOG

/// <summary>

/// Движение вправо

/// </summary>

public void MoveRightOG()

{

GameObjects[OrangePositionX, OrangePositionY] = ObjOrange;

ObjOrange = GameObjects[OrangePositionX + 1, OrangePositionY];

GameObjects[OrangePositionX + 1, OrangePositionY] = 6;

OrangePositionX += 1;

}

#endregion

#endregion

#endregion

}

}

**Form1.cs**

#region Using

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

#endregion

namespace PacmanForms

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

#region Public Variables

//Создание объекта класса

public GameAppearance Game;

//Создания массива с файлами

public FileInfo[] Content;

//Создание рандома для красного приведения

public Random rndRed;

//Создание рандома для розового приведения

public Random rndPink;

//Создание рандома для зеленого приведения

public Random rndGreen;

//Создание рандома для оранжевого приведения

public Random rndOrange;

#endregion

#region Form1\_Load

/// <summary>

/// Загрузка данных в pictureBox

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Form1\_Load(object sender, EventArgs e)

{

StartGame();

}

#endregion

#region Form1\_KeyDown

/// <summary>

/// Обработчик нажатых клавиш

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

#region Keys.Down

//При нажатии на клавишу вниз

if (e.KeyCode == Keys.Down)

{

//Если нет снизу стены

if (!Game.IsWallBottom())

{

//Если снизу привидение - GameOver

if (Game.IsRedGhostBottom() || Game.IsPinkGhostBottom() || Game.IsGreenGhostBottom() || Game.IsOrangeGhostBottom())

{

GameOver();

}

else

{

//Если есть еда, то съесть и прибавить к очкам

if (Game.IsMealNear())

{

Game.EatDown();

Game.Apperance();

}

//Иначе просто передвинуться

else

{

Game.MoveDown();

Game.Apperance();

}

}

}

}

#endregion

#region Keys.Up

//При нажатии на клавишу Вверх

if (e.KeyCode == Keys.Up)

{

//Если сверху нет стены

if (!Game.IsWallAbove())

{

//Если сверху приведение - GameOver

if (Game.IsRedGhostAbove() || Game.IsPinkGhostAbove() || Game.IsGreenGhostAbove() || Game.IsOrangeGhostAbove())

{

GameOver();

}

else

{

//Если сверху еда, съесть и прибавить к очкам

if (Game.IsMealNear())

{

Game.EatUp();

Game.Apperance();

}

//Иначе просто продвинутся

else

{

Game.MoveUp();

Game.Apperance();

}

}

}

}

#endregion

#region Keys.Left

//Если нажата клавиша влево

if (e.KeyCode == Keys.Left)

{

//Если нет стены

if (!Game.IsWallLeft())

{

//Если слева приведение - GameOver

if (Game.IsRedGhostLeft() || Game.IsPinkGhostLeft() || Game.IsGreenGhostLeft() || Game.IsOrangeGhostLeft())

{

GameOver();

}

else

{

//Если слева еда, съесть и привать к счету

if (Game.IsMealNear())

{

Game.EatLeft();

Game.Apperance();

}

//Иначе просто передвинуться

else

{

Game.MoveLeft();

Game.Apperance();

}

}

}

}

#endregion

#region Keys.Right

//Если нажата клавиша вправо

if (e.KeyCode == Keys.Right)

{

//Если нет стены

if (!Game.IsWallRight())

{

//Если справа приведение - GameOver

if (Game.IsRedGhostRight() || Game.IsPinkGhostRight() || Game.IsGreenGhostRight() || Game.IsOrangeGhostRight())

{

GameOver();

}

else

{

//Если справо еда, съесть и привать к счету

if (Game.IsMealNear())

{

Game.EatRight();

Game.Apperance();

}

//Иначе просто передвинуться

else

{

Game.MoveRight();

Game.Apperance();

}

}

}

}

#endregion

#region Игра пройдена

//Если съели всю еду - игра пройдена

if (Game.IsMeal())

{

timer1.Stop();

DialogResult result1 = MessageBox.Show("Вы прошли игру! Поздравляю! \nХотите попробовать еще раз?", "Победа!", MessageBoxButtons.YesNo);

if (result1 == DialogResult.Yes)

{

StartGame();

}

else

{

Form1.ActiveForm.Close();

}

}

#endregion

}

#endregion

#region Timer

/// <summary>

/// Таймер для управления приведениями

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void timer1\_Tick(object sender, EventArgs e)

{

#region Передвижения красного приведения

//Создание объекта рандом для красного приведения

rndRed = new Random();

//Метод для рандомизации значений

int r = rndRed.Next(4);

//Ветвления, зависящие от значий а

switch (r)

{

//При значении 0

case 0:

{

//Если нет снизу стены или приведений

if (!Game.IsWallBottomRG() && !Game.IsPinkGhostBottomRG() &&

!Game.IsGreenGhostBottomRG() && !Game.IsOrangeGhostBottomRG())

{

//Если снизу пакман снизу-Game over

if (Game.IsPackmanBottomRG())

{

GameOver();

}

else

{

//Передвижение красного приведения вниз

Game.MoveDownRG();

Game.Apperance();

}

}

break;

}

//При заначении 1

case 1:

{

//Если нет сверху стены и приведений

if (!Game.IsWallAboveRG()&&!Game.IsPinkGhostAboveRG() &&

!Game.IsGreenGhostAboveRG() && !Game.IsOrangeGhostAboveRG())

{

//если пакман сверху - game over

if (Game.IsPackmanAboveRG())

GameOver();

else

{

//Передвижение красного приведения вверх

Game.MoveUpRG();

Game.Apperance();

}

}

break;

}

//При значении 2

case 2:

{

//Если нет слева стены и приведений

if (!Game.IsWallLeftRG() && !Game.IsPinkGhostLeftRG() &&

!Game.IsGreenGhostLeftRG() && !Game.IsOrangeGhostLeftRG())

{

//Если Пакман слева - gameover

if (Game.IsPackmanLeftRG())

GameOver();

else

{

//Передвижение красного приведения влево

Game.MoveLeftRG();

Game.Apperance();

}

}

break;

}

//При значении 3

case 3:

{

//Если нет справа стены и приведений

if (!Game.IsWallRightRG() && !Game.IsPinkGhostRightRG() &&

!Game.IsGreenGhostRightRG() && !Game.IsOrangeGhostRightRG())

{

//Если справа пакман - gameover

if (Game.IsPackmanRightRG())

GameOver();

else

{

//Передвижение красного приведения вниз

Game.MoveRightRG();

Game.Apperance();

}

}

break;

}

}

#endregion

#region Передвижения розового приведения

//Создание объекта рандом для розового приведения

rndPink = new Random();

//Метод для рандомизации значений

int p = rndPink.Next(4);

//Ветвления, зависящие от значий а

switch (p)

{

//При значении 0

case 0:

{

//Если нет снизу стены и приведений

if (!Game.IsWallBottomPG() && !Game.IsRedGhostBottomPG() &&

!Game.IsGreenGhostBottomPG() && !Game.IsOrangeGhostBottomPG())

{

//Если снизу пакман снизу-Game over

if (Game.IsPackmanBottomPG())

{

GameOver();

}

else

{

//Передвижение розового приведения вниз

Game.MoveDownPG();

Game.Apperance();

}

}

break;

}

//При заначении 1

case 1:

{

//Если нет сверху стены и приведений

if (!Game.IsWallAbovePG() && !Game.IsRedGhostAbovePG() &&

!Game.IsGreenGhostAbovePG() && !Game.IsOrangeGhostAbovePG())

{

//если пакман сверху - game over

if (Game.IsPackmanAbovePG())

{

GameOver();

}

else

{

//Передвижение розового приведения вверх

Game.MoveUpPG();

Game.Apperance();

}

}

break;

}

//При значении 2

case 2:

{

//Если нет слева стены и приведений

if (!Game.IsWallLeftPG() && !Game.IsRedGhostLeftPG() &&

!Game.IsGreenGhostLeftPG() && !Game.IsOrangeGhostLeftPG())

{

//Если Пакман слева - gameover

if (Game.IsPackmanLeftPG())

GameOver();

else

{

//Передвижение розового приведения влево

Game.MoveLeftPG();

Game.Apperance();

}

}

break;

}

//При значении 3

case 3:

{

//Если нет справа стены и приведений

if (!Game.IsWallRightPG() && !Game.IsRedGhostRightPG() &&

!Game.IsGreenGhostRightPG() && !Game.IsOrangeGhostRightPG())

{

//Если справа пакман - gameover

if (Game.IsPackmanRightPG())

GameOver();

else

{

//Передвижение розового приведения вправо

Game.MoveRightPG();

Game.Apperance();

}

}

break;

}

}

#endregion

#region Передвижения зеленого приведения

//Создание объекта рандом для зеленого приведения

rndGreen = new Random();

//Метод для рандомизации значений

int g = rndGreen.Next(4);

//Ветвления, зависящие от значий а

switch (g)

{

//При значении 0

case 0:

{

//Если нет снизу стены и приведений

if (!Game.IsWallBottomGG() && !Game.IsRedGhostBottomGG() &&

!Game.IsPinkGhostBottomGG() && !Game.IsOrangeGhostBottomGG())

{

//Если снизу пакман снизу-Game over

if (Game.IsPackmanBottomGG())

{

GameOver();

}

else

{

//Передвижение зеленого приведения вниз

Game.MoveDownGG();

Game.Apperance();

}

}

break;

}

//При заначении 1

case 1:

{

//Если нет сверху стены и приведений

if (!Game.IsWallAboveGG() && !Game.IsRedGhostAboveGG() &&

!Game.IsPinkGhostAboveGG() && !Game.IsOrangeGhostAboveGG())

{

//если пакман сверху - game over

if (Game.IsPackmanAboveGG())

{

GameOver();

}

else

{

//Передвижение зеленого приведения вверх

Game.MoveUpGG();

Game.Apperance();

}

}

break;

}

//При значении 2

case 2:

{

//Если нет слева стены и приведений

if (!Game.IsWallLeftGG() && !Game.IsRedGhostLeftGG() &&

!Game.IsPinkGhostLeftGG() && !Game.IsOrangeGhostLeftGG())

{

//Если Пакман слева - gameover

if (Game.IsPackmanLeftGG())

GameOver();

else

{

//Передвижение зеленого приведения влево

Game.MoveLeftGG();

Game.Apperance();

}

}

break;

}

//При значении 3

case 3:

{

//Если нет справа стены и приведений

if (!Game.IsWallRightGG() && !Game.IsRedGhostRightGG() &&

!Game.IsPinkGhostRightGG() && !Game.IsOrangeGhostRightGG())

{

//Если справа пакман - gameover

if (Game.IsPackmanRightGG())

GameOver();

else

{

//Передвижение зеленого приведения вправо

Game.MoveRightGG();

Game.Apperance();

}

}

break;

}

}

#endregion

#region Передвижения оранжевого приведения

//Создание объекта рандом для оранжевого приведения

rndOrange = new Random();

//Метод для рандомизации значений

int o = rndOrange.Next(4);

//Ветвления, зависящие от значий а

switch (o)

{

//При значении 0

case 0:

{

//Если нет снизу стены и приведений

if (!Game.IsWallBottomOG() && !Game.IsRedGhostBottomOG() &&

!Game.IsPinkGhostBottomOG() && !Game.IsGreenGhostBottomOG())

{

//Если снизу пакман снизу-Game over

if (Game.IsPackmanBottomOG())

{

GameOver();

}

else

{

//Передвижение оранжевого приведения вниз

Game.MoveDownOG();

Game.Apperance();

}

}

break;

}

//При заначении 1

case 1:

{

//Если нет сверху стены и приведений

if (!Game.IsWallAboveOG() && !Game.IsRedGhostAboveOG() &&

!Game.IsPinkGhostAboveOG() && !Game.IsGreenGhostAboveOG())

{

//если пакман сверху - game over

if (Game.IsPackmanAboveOG())

{

GameOver();

}

else

{

//Передвижение оранжевого приведения вверх

Game.MoveUpOG();

Game.Apperance();

}

}

break;

}

//При значении 2

case 2:

{

//Если нет слева стены и приведений

if (!Game.IsWallLeftOG() && !Game.IsRedGhostLeftOG() &&

!Game.IsPinkGhostLeftOG() && !Game.IsGreenGhostLeftOG())

{

//Если Пакман слева - gameover

if (Game.IsPackmanLeftOG())

GameOver();

else

{

//Передвижение оранжевого приведения влево

Game.MoveLeftOG();

Game.Apperance();

}

}

break;

}

//При значении 3

case 3:

{

//Если нет справа стены и приведений

if (!Game.IsWallRightOG() && !Game.IsRedGhostRightOG() &&

!Game.IsPinkGhostRightOG() && !Game.IsGreenGhostRightOG())

{

//Если справа пакман - gameover

if (Game.IsPackmanRightOG())

GameOver();

else

{

//Передвижение оранжевого приведения вправо

Game.MoveRightOG();

Game.Apperance();

}

}

break;

}

}

#endregion

}

#endregion

#region StartGame

public void StartGame()

{

//Создание новой директории игры

DirectoryInfo directory = new DirectoryInfo("Resourses");

//Проверка: если директория не путая, берем файлы

if (directory != null)

{

Content = directory.GetFiles();

}

//Новый объект игры

Game = new GameAppearance(pictureBox1, Content);

//Функция вызова содержимого игры

Game.Apperance();

//Запуск таймера

timer1.Start();

}

#endregion

#region GameOver

//Завершение игры

private void GameOver()

{

timer1.Stop();

DialogResult result1 = MessageBox.Show("Вы проиграли. \nХотите начать заново?", "Game Over", MessageBoxButtons.YesNo);

if (result1 == DialogResult.Yes)

{

StartGame();

}

else

{

Form1.ActiveForm.Close();

}

}

#endregion

}

}