Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования  
«Новосибирский государственный технический университет»

Кафедра Автоматизированных систем управления



**Отчет**

**по лабораторной работе №3**

**Тема:**

**Дисциплина**: Программирование

Вариант **№9**

|  |  |
| --- | --- |
| Выполнили:  *Коваленко Е.А.*  *Арифуллин А.Д.*  Факультет *АВТФ*  Направление подготовки  *09.03.03. – Прикладная информатика*  Группа *АП-327* | Проверил:  *Cтарший преподаватель кафедры АСУ*  *Эстрайх И.В.* |

Новосибирск

2024

Оглавление

[Постановка задачи 3](#_Toc178323991)

[Тестовые данные и ожидаемые результаты 3](#_Toc178323992)

[Описание разработанных классов 3](#_Toc178323993)

[Тестирование программы 6](#_Toc178323994)

[Скриншоты результатов работы программы и выполнения тестов 6](#_Toc178323995)

[Выводы 8](#_Toc178323996)

[Приложение. Исходные коды всех разработанных файлов с комментариями. 8](#_Toc178323997)

# Постановка задачи

Разработать игру в жанре Арканоид с выпадающими из уничтоженных кирпичей предметами. Программа должна поддерживать сериализацию-десериализацию, код покрыт модульными тестами.

# Тестовые данные и ожидаемые результаты

|  |  |
| --- | --- |
| n | Запустится новая игра |
| y | Запустится сохраненная игра |

# Описание разработанных классов

**Класс Field**

|  |  |  |  |
| --- | --- | --- | --- |
| Поле/Свойство/Метод | Тип/Возвращаемый тип | Параметры | Описание |
| b\_hor | int | - | Горизонтальное направление движения мяча. |
| b\_ver | int | - | Вертикальное направление движения мяча. |
| speed | double | - | Скорость игры |
| pl\_pos | int | - | Позиция платформы на поле |
| pl\_size | int | - | Размер платформы |
| field | string[,] | - | Основное игровое поле. |
| buff\_field | string[,] | - | Поле, аналогичное игровому по размерам и содержащее бонусы |
| Field() | Field | int b\_h, int b\_v, int pl\_p, int pl\_s, double spd | Инициализирует игровое поле с указанными или стандартными параметрами. |
| FieldPrint() | void | - | Отображает текущее состояние игрового поля в консоли. |
| PlatformMove() | void | int dir | Перемещает платформу в указанном направлении, если это допустимо. |
| BallMove() | bool | - | Перемещает мяч в соответствии с направлением, возвращает false, если мяч достиг нижней границы. |
| CrushBlock() | void | int n, int m | Уничтожает блок на указанной позиции, возможно добавляя бонус в buff\_field. |
| WinCheck() | bool | - | Проверяет, есть ли на поле ещё разрушимые блоки. Возвращает true, если все разрушены. |
| BuffFalling() | void | - | Обрабатывает падение бонусов на одну позицию вниз. Применяет эффект бонусов, достигших платформы. |
| BallCollision() | bool | int n, int m | Проверяет наличие столкновения мяча с объектами. Обрабатывает смену направления мяча и уничтожение блоков, если требуется. |

**Класс Program**

|  |  |  |  |
| --- | --- | --- | --- |
| Поле/Свойство/Метод | Тип/Возвращаемый тип | Параметры | Описание |
| N | const int | - | Высота игрового поля. |
| M | const int | - | Ширина игрового поля. |
| Chance | const int | - | Шанс выпадения бонусов |
| Main | void | string[] args | Выводит управление, начинает игру, сохраняет/загружает игру, настраивает управление платформой, уведомляет игрока о победе или поражении |

# 

# Описание приложения

Изначально пользователя встречают инструкции управления и предложение загрузить игру или начать новую игру. После начала игры рисуется игровое поле. Пользователю необходимо, двигая платформой внизу игрового поля с помощью стрелочек влево-вправо не допускать падения мяча за нижнюю границу поля и разбить с помощью этого самого мяча все кирпичи. При нажатии на левую кнопку мыши игрок может поставить игру на паузу. При нажатии на ESC игроку может сохранить игру и выйти или выйти без сохранения. Если игрок сумел разбить все кирпичи мячом, то программа уведомляет его о победе. Если же мяч упал за нижнюю границу поля игра говорит игроку о поражении.

# Тестирование программы

Результаты тестирования программы оказались удовлетворительными. Полученные результаты совпали с теоретическими представлениями, модульные тесты успешно пройдены.

# Скриншоты результатов работы программы и выполнения тестов

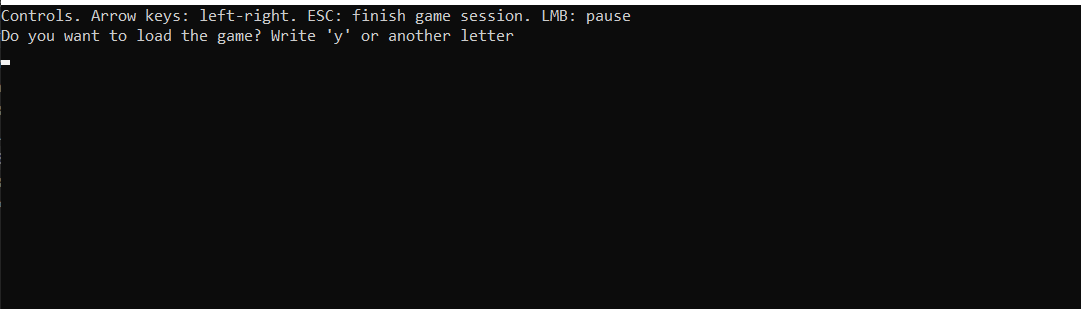


Рисунок . Начало игры

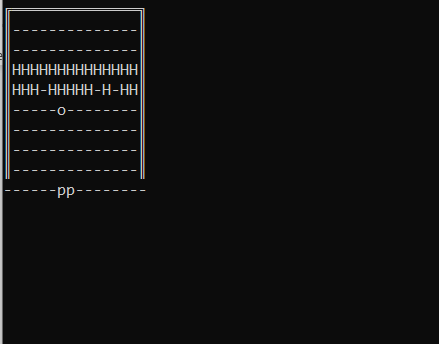


Рисунок Сама игра

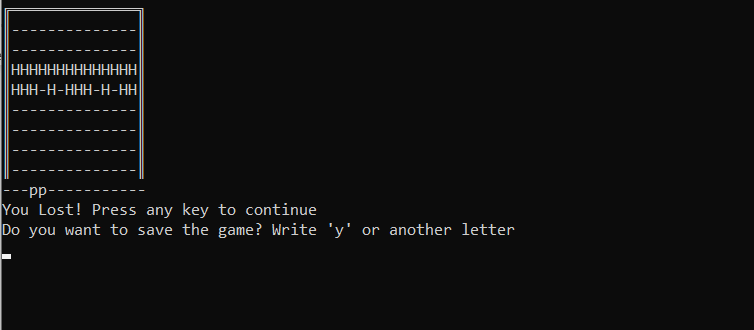


Рисунок 3 Уведомление о конце игры

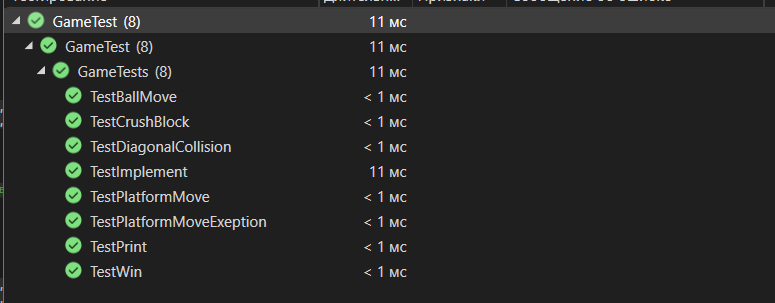


Рисунок Результат выполнения модульных тестов

# Выводы

При выполнении данной лабораторной работы мы разработали базовую игру в жанре Арканоид в виде консольного приложения. Мы освоили сериализацию/десериализацию, покрыли код модульными тестами и в итоге загрузили программу в репозиторий Git.

# Приложение. Исходные коды всех разработанных файлов с комментариями.

**Основная программа**

using System.Runtime.Serialization.Formatters.Binary;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace Game

{

public class Program

{

const int N = 10;

const int M = 16;

const int Chance = 2; // higher the number - lower the chance to obtain a bonus

// "-" - nothing "b" - ball "w" - wall "c" - crushable

[Serializable]

public class Field

{

public int b\_hor { get; set; } // horizontal ball direction

public int b\_ver { get; set; } // vertical ball direction

public double speed { get; set; } // higher the number - lower the speed to obtain a bonus

public int pl\_pos { get; set; } // platform position

public int pl\_size { get; set; } // platform size

public string[,] field { get; set; }

public string[,] buff\_field { get; set; }

public Field(int b\_h = 1, int b\_v = -1, int pl\_p = 0, int pl\_s = 3, double spd = 400)

{

b\_hor = b\_h;

b\_ver = b\_v;

pl\_pos = pl\_p;

pl\_size = pl\_s;

speed = spd;

field = new string[N, M];

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

{

field[0, j] = "w";

}

for (int i = 1; i < N - 1; i++)

{

field[i, 0] = "w";

for (int j = 1; j < M - 1; j++)

{

field[i, j] = "-";

}

field[i, M - 1] = "w";

}

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

{

field[N - 1, j] = "-";

}

for (int j = 1; j < M - 1; j++)

{

field[N / 2 - 2, j] = "c";

}

for (int j = 1; j < M - 1; j++)

{

field[N / 2 - 1, j] = "c";

}

field[N - 2, 1] = "b";

buff\_field = new string[N, M];

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

{

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

{

buff\_field[i, j] = "-";

}

}

}

public void FieldPrint()

{

string tmp = "";

Console.Write("╔");

for (int j = 1; j < M - 1; j++)

{

Console.Write("═");

}

Console.WriteLine("╗");

for (int i = 1; i < N - 1; i++)

{

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

{

switch (field[i, j])

{

case "w":

tmp = "║";

break;

case "b":

tmp = "o";

break;

case "c":

tmp = "H";

break;

default:

tmp = "-";

break;

}

if (buff\_field[i, j] != "-")

{

tmp = buff\_field[i, j];

}

Console.Write(tmp);

}

Console.WriteLine();

}

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

{

if ((j >= pl\_pos) && (j <= pl\_pos + pl\_size - 1))

{

Console.Write("p");

}

else

{

Console.Write("-");

}

}

Console.WriteLine();

}

public void PlatformMove(int dir)

{

if ((pl\_pos + dir >= 0) && (pl\_size + pl\_pos + dir <= M))

{

pl\_pos = pl\_pos + dir;

}

}

public bool BallMove()

{

int n = 0;

int m = 0;

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

{

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

{

if (field[i, j] == "b")

{

n = i;

m = j;

}

}

}

if (n == N - 1)

{

return false;

}

if (!BallCollision(n, m))

{

field[n, m] = "-";

field[n + b\_ver, m + b\_hor] = "b";

}

return true;

}

public void CrushBlock(int n, int m)

{

if (field[n, m] == "c")

{

field[n, m] = "-";

Random rnd = new Random();

int dice;

if (rnd.Next(Chance) == 0)

{

dice = rnd.Next(1, 5); // 1-platform\_size\_up 2-platform\_size\_down 3-game\_speed\_up 4-game\_speed\_down

buff\_field[n, m] = dice.ToString();

}

}

}

public bool WinCheck()

{

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

{

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

{

if (field[i, j] == "c")

{

return false;

}

}

}

return true;

}

public void BuffFalling()

{

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

{

switch (buff\_field[N - 1, j])

{

case "1":

pl\_size = pl\_size + 1;

break;

case "2":

if (pl\_size - 1 >= 2)

{

pl\_size = pl\_size - 1;

}

break;

case "3":

speed = speed / 1.4;

if (speed < 300)

{

speed = 300;

}

break;

case "4":

speed = speed \* 1.4;

if (speed > 600)

{

speed = 600;

}

break;

default:

break;

}

}

for (int i = N - 1; i > 0; i--)

{

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

{

buff\_field[i, j] = buff\_field[i - 1, j];

}

}

}

public bool BallCollision(int n, int m)

{

bool diagonal\_check = true;

if (field[n + b\_ver, m] != "-")

{

diagonal\_check = false;

CrushBlock(n + b\_ver, m);

b\_ver = b\_ver \* -1;

return true;

}

else

{

if ((n + b\_ver == N - 1) && (pl\_pos <= m) && (pl\_pos + pl\_size >= m))

{

diagonal\_check = false;

b\_ver = b\_ver \* -1;

return true;

}

}

if (field[n, m + b\_hor] != "-")

{

diagonal\_check = false;

CrushBlock(n, m + b\_hor);

b\_hor = b\_hor \* -1;

return true;

}

if (diagonal\_check && field[n + b\_ver, m + b\_hor] != "-")

{

CrushBlock(n + b\_ver, m + b\_hor);

b\_ver = b\_ver \* -1;

b\_hor = b\_hor \* -1;

return true;

}

else

{

if (diagonal\_check && (n + b\_ver == N - 1) && (pl\_pos <= m + b\_hor) && (pl\_pos + pl\_size >= m + b\_hor))

{

b\_ver = b\_ver \* -1;

b\_hor = b\_hor \* -1;

return true;

}

}

return false;

}

}

static void Main(string[] args)

{

Console.WriteLine("Controls. Arrow keys: left-right. ESC: finish game session. LMB: pause");

Console.WriteLine("Do you want to load the game? Write 'y' or another letter");

string loadflag = Console.ReadLine();

Console.CursorVisible = false;

Field game\_field = new Field();

BinaryFormatter formatter = new BinaryFormatter();

if (loadflag == "y")

{

// десериализация из файла gmamelevel.dat

using (FileStream fs = new FileStream("gmamelevel.dat", FileMode.OpenOrCreate))

{

game\_field = (Field)formatter.Deserialize(fs);

Console.WriteLine("Объект десериализован");

}

}

Console.Clear();

bool winflag = false;

bool lostflag = false;

System.ConsoleKey command\_key;

do

{

while (!Console.KeyAvailable)

{

Thread.Sleep((int)game\_field.speed);

game\_field.BuffFalling();

lostflag = !game\_field.BallMove();

winflag = game\_field.WinCheck();

Console.SetCursorPosition(0, 0);

game\_field.FieldPrint();

if (lostflag)

{

Console.CursorVisible = true;

Console.WriteLine("You Lost! Press any key to continue");

break;

}

if (winflag)

{

Console.CursorVisible = true;

Console.WriteLine("You Won! Press any key to continue");

break;

}

}

command\_key = Console.ReadKey(true).Key;

if (command\_key == ConsoleKey.LeftArrow)

{

game\_field.PlatformMove(-1);

}

if (command\_key == ConsoleKey.RightArrow)

{

game\_field.PlatformMove(1);

}

if (lostflag || winflag)

{

break;

}

} while (command\_key != ConsoleKey.Escape);

Console.CursorVisible = true;

Console.WriteLine("Do you want to save the game? Write 'y' or another letter");

string saveflag = Console.ReadLine();

if (saveflag == "y")

{

// получаем поток, куда будем записывать сериализованный объект

using (FileStream fs = new FileStream("gmamelevel.dat", FileMode.OpenOrCreate))

{

formatter.Serialize(fs, game\_field);

Console.WriteLine("Объект сериализован");

}

}

Console.WriteLine();

}

}

}

**Модульные тесты**

using System;

using Game;

using System.Security.Principal;

using System.IO;

namespace GameTest

{

[TestClass]

public class GameTests

{

const int N = 10;

const int M = 16;

[TestMethod]

public void TestPlatformMove()

{

// Arrange

Program.Field TestField1 = new Program.Field(1, -1, 0, 3, 300);

Program.Field TestField2 = new Program.Field(1, -1, 1, 3, 300);

// Act

TestField1.PlatformMove(1);

// Assert

Assert.AreEqual(TestField1.pl\_pos, TestField2.pl\_pos);

// попытка подвинуть находящуюся в левом углу платформу вправо должна привести к тому, что платформа сдвинется

}

[TestMethod]

public void TestPlatformMoveExeption()

{

// Arrange

Program.Field TestField1 = new Program.Field(1, -1, 0, 3, 300);

Program.Field TestField2 = new Program.Field(1, -1, 0, 3, 300);

// Act

TestField1.PlatformMove(-1);

// Assert

Assert.AreEqual(TestField1.pl\_pos, TestField2.pl\_pos);

// попытка подвинуть находящуюся в левом углу платформу влево должна привести к тому, что платформа останется на месте

}

[TestMethod]

public void TestBallMove()

{

// Arrange

Program.Field TestField1 = new Program.Field(1, 1, 0, 3, 300);

TestField1.field = new string[N, M];

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

{

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

{

TestField1.field[i, j] = "-";

}

}

TestField1.field[5, 5] = "b";

Program.Field TestField2 = new Program.Field(1, 1, 0, 3, 300);

TestField2.field = new string[N, M];

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

{

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

{

TestField2.field[i, j] = "-";

}

}

TestField2.field[6, 6] = "b";

// Act

TestField1.BallMove();

// Assert

Assert.AreEqual(TestField1.field[6, 6], TestField2.field[6, 6]);

// проверка нормального перемещения мяча

}

[TestMethod]

public void TestCrushBlock()

{

// Arrange

Program.Field TestField1 = new Program.Field(1, 1, 0, 3, 300);

TestField1.field = new string[N, M];

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

{

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

{

TestField1.field[i, j] = "-";

}

}

TestField1.field[5, 5] = "c";

// Act

TestField1.CrushBlock(5, 5);

// Assert

Assert.AreEqual(TestField1.field[5, 5], "-");

// проверка разрушаемости блока

}

[TestMethod]

public void TestWin()

{

// Arrange

Program.Field TestField1 = new Program.Field(1, 1, 0, 3, 300);

TestField1.field = new string[N, M];

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

{

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

{

TestField1.field[i, j] = "-";

}

}

// Assert

Assert.AreEqual(TestField1.WinCheck(), true);

// проверка на условие победы (полное отсутствие разрушаемых блоков)

}

[TestMethod]

public void TestImplement()

{

// Arrange

Program.Field TestField1 = new Program.Field(1, 1, 0, 3, 300);

TestField1.buff\_field[N - 1, 1] = "1";

Program.Field TestField2 = new Program.Field(1, 1, 0, 4, 300);

// Act

TestField1.BuffFalling();

// Assert

Assert.AreEqual(TestField1.pl\_size, TestField2.pl\_size);

// проверка эффекта от усиления

}

[TestMethod]

public void TestDiagonalCollision()

{

// Arrange

Program.Field TestField1 = new Program.Field(1, 1, 0, 3, 300);

TestField1.field = new string[N, M];

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

{

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

{

TestField1.field[i, j] = "-";

}

}

TestField1.field[5, 5] = "c";

TestField1.field[4, 4] = "b";

// Act

TestField1.BallCollision(4, 4);

// Assert

Assert.AreEqual(TestField1.field[5, 5], "-");

// проверка обработки столкновения с разрушаемым объектом по диагонали

}

[TestMethod]

public void TestPrint()

{

// Arrange

var stringWriter1 = new StringWriter();

Console.SetOut(stringWriter1);

Program.Field TestField = new Program.Field();

for (int i = 1; i < N; i++)

{

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

{

TestField.field[i, j] = "-";

}

}

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

{

TestField.field[0, j] = "w";

}

TestField.field[1, 0] = "w";

TestField.field[1, 1] = "c";

TestField.field[1, 2] = "b";

// Act

TestField.FieldPrint();

var TrueOut = stringWriter1.ToString();

var stringWriter2 = new StringWriter();

Console.SetOut(stringWriter2);

Console.Write("╔");

for (int j = 1; j < M - 1; j++)

{

Console.Write("═");

}

Console.WriteLine("╗");

Console.Write("║");

Console.Write("H");

Console.Write("o");

for (int j = 3; j < M; j++)

{

Console.Write("-");

}

Console.WriteLine();

for (int i = 2; i < N - 1; i++)

{

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

{

Console.Write("-");

}

Console.WriteLine();

}

Console.Write("p");

Console.Write("p");

Console.Write("p");

for (int j = 3; j < M; j++)

{

Console.Write("-");

}

Console.WriteLine();

var ExpectedOut = stringWriter2.ToString();

// Assert

Assert.AreEqual(TrueOut, ExpectedOut);

}

}

}