МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики  
Кафедра информационных технологий

Отчет по индивидуальному заданию №4

**«Создание приложения “Жизнь”»**

Работу выполнил  
студент 42 группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Б.М.Ибрагимов

Направление подготовки 01.03.02 Прикладная математика и информатика

Проверил  
канд. техн. наук, доц.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А.Полупанов

Краснодар 2020

**Цель работы:** разработать игру «Жизнь» на C#.

**Задача 1.** Разработать игру «Жизнь»: В начале игры присутствуют:

1. Одно существо женского пола (красный круг радиуса 20)
2. Одно существо мужского пола (синий круг радиуса 20)
3. Произвольное количество еды (зеленые круги радиуса 20), распределенные по игровому полю

**ЛИСТИНГ ПРОГРАММЫ**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Life {

public partial class Form1 : Form {

public static Size size = new Size(20, 20);

List<Entity> allPerson = new List<Entity>();

List<Point> allFood = new List<Point>();

List<Entity> allDeadPerson = new List<Entity>();

Graphics g;

public int maxEntity = 500, countAteFood = 0;

public static List<Color> colors = new List<Color>() { Color.Blue, Color.Red, Color.Green };

readonly Random random = new Random();

long step = 0;

DateTime timeStart;

public Form1() {

InitializeComponent();

}

public void generateFood() {

Point p;

allFood.Clear();

int countFood = random.Next(10,100);

for (int i = 0; i < countFood; i++) {

p = new Point(random.Next(size.Width, board.Width - size.Width), random.Next(size.Height, board.Height - size.Height));

allFood.Add(p);

}

}

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

timeStart = DateTime.Now;

board.Size = new Size(800, 600);

timer1.Enabled = true;

allPerson.Add(new Entity(new Point(385, 285),colors[0]));

allPerson.Add(new Entity(new Point(400, 300),colors[1]));

}

private void board\_Paint(object sender, PaintEventArgs e) {

g = e.Graphics;

foreach (Entity entity in allPerson) {

g.FillEllipse(new SolidBrush(entity.gender), new Rectangle(entity.position, size));

}

if (step % 100 == 0) generateFood();

foreach (Point p in allFood)

g.FillEllipse(new SolidBrush(colors[2]), new Rectangle(p, size));

}

private void Timer1\_Tick(object sender, EventArgs e) {

step++;

List<Entity> children = new List<Entity>();

foreach (Entity entity in allPerson) {

entity.Run(board);

List<Point> ateFoot = new List<Point>();

foreach (Point posFood in allFood) {

if (entity.isPersonalSpaceFood(posFood)) ateFoot.Add(posFood);

}

countAteFood += ateFoot.Count;

foreach (Point p in ateFoot) {

allFood.Remove(p);

if (entity.countIteration < 80) entity.countIteration += 20;

}

if (entity.countIteration == 0 || entity.countIterationInLife == 0) allDeadPerson.Add(entity);

}

List<Entity> allPersonCopy = allPerson.GetRange(0,allPerson.Count);

int i = 0;

while(allPersonCopy.Count != 0) {

Entity entityIntersection = allPersonCopy[i].isPersonalSpaceEntity(allPersonCopy);

if (entityIntersection != null) {

Rectangle R = new Rectangle(Math.Min(entityIntersection.position.X, allPersonCopy[i].position.X), Math.Min(entityIntersection.position.Y, allPersonCopy[i].position.Y), 2 \* size.Width, 2 \* size.Height);

if (entityIntersection.countIteration <= 20 && allPersonCopy[i].countIteration <= 20) {

children.Add(new Entity(new Point(R.X, R.Y), colors[random.Next(2)]));

children.Add(new Entity(new Point(R.Right, R.Bottom), colors[random.Next(2)]));

}

if (entityIntersection.countIteration <= 10 || allPersonCopy[i].countIteration <= 10) {

children.Add(new Entity(new Point(R.X, R.Y), colors[random.Next(2)]));

}

if (entityIntersection.countIteration <= 40 || allPersonCopy[i].countIteration <= 40) {

children.Add(new Entity(new Point(R.X, R.Y), colors[random.Next(2)]));

children.Add(new Entity(new Point(R.Right, R.Bottom), colors[random.Next(2)]));

children.Add(new Entity(new Point(R.X, R.Bottom), colors[random.Next(2)]));

}

if (entityIntersection.countIteration <= 60 && allPersonCopy[i].countIteration <= 60) {

children.Add(new Entity(new Point(R.X, R.Y), colors[random.Next(2)]));

children.Add(new Entity(new Point(R.Right, R.Bottom), colors[random.Next(2)]));

children.Add(new Entity(new Point(R.X, R.Bottom), colors[random.Next(2)]));

children.Add(new Entity(new Point(R.Right, R.Y), colors[random.Next(2)]));

}

allPersonCopy.Remove(entityIntersection);

}

allPersonCopy.RemoveAt(i);

}

foreach (Entity entity1 in allDeadPerson) allPerson.Remove(entity1);

foreach (Entity entity1 in children) allPerson.Add(entity1);

allDeadPerson.Clear();

board.Refresh();

if (allPerson.Count >= maxEntity || allPerson.Count == 0) {

timer1.Enabled = false;

int countM = 0, countW = 0;

foreach (Entity entity1 in allPerson) {

if (entity1.gender == Color.Red) countW++;

else countM++;

}

MessageBox.Show("Потрачено времени = " + Math.Round((DateTime.Now - timeStart).TotalSeconds,3) + "\nКоличество женщин = " + countM + "\nКоличество мужчин = " + countW + "\nКоличество съеденной еды = " + countAteFood);

}

}

}

public class Entity {

readonly List<Point> orientations = new List<Point>() { new Point(0, -1), new Point(-1, 0), new Point(0, 1), new Point(1, 0) };

public Point position;

public int countIterationInLife = 500;

public int countIteration = 200;

private static readonly Random random = new Random();

public Color gender;

public int radius = Form1.size.Width;

public int countSteps = 5;

public Entity(Point position, Color gender) {

this.position = position;

this.gender = gender;

}

public void Run(PictureBox board) {

Point orientation = orientations[random.Next(orientations.Count)];

position.X += orientation.X \* countSteps;

position.Y += orientation.Y \* countSteps;

if (position.X + radius > board.Width) {

position.X = 2 \* radius + position.X - board.Width;

}

if (position.X - radius < 0) {

position.X = board.Width + position.X - 2 \* radius;

}

if (position.Y + radius > board.Height) {

position.Y = 2 \* radius + position.Y - board.Height;

}

if (position.Y - radius < 0) {

position.Y = board.Height + position.Y - 2 \* radius;

}

countIteration--;

countIterationInLife--;

}

public Entity isPersonalSpaceEntity(List<Entity> entities) {

for (int i = 0; i < entities.Count; i++) {

if (entities[i] != this && entities[i].gender != gender) {

Point distance = new Point(Math.Abs(entities[i].position.X - position.X), Math.Abs(entities[i].position.Y - position.Y));

if (distance.X < radius && distance.Y < radius) {

return entities[i];

}

}

}

return null;

}

public bool isPersonalSpaceFood(Point food) {

Point distance = new Point(Math.Abs(food.X - position.X), Math.Abs(food.Y - position.Y));

if (distance.X < radius && distance.Y < radius) return true;

else return false;

}

}

}

**РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

На рисунке 1 изображено игровое поле размером 800\*600 в начале:

Изображение выглядит как белый

Автоматически созданное описание

Рисунок 1 – игровое поле в начале игры

В начале игры, все существа сытые и могут сделать 200 итераций по 5 шагов в каждой в произвольном направлении. После того, как существо съело еду (пересеклось с едой), количество итераций увеличивается на 20. Если количество оставшихся итераций равно 0, то существо считается умершим и исчезает из поля. Если существа мужского и женского полов пересекаются, то в зависимости от количества оставшихся итераций, появляется разное количество новых существ произвольных полов. Этот процесс показан на рисунках 2 и 3:

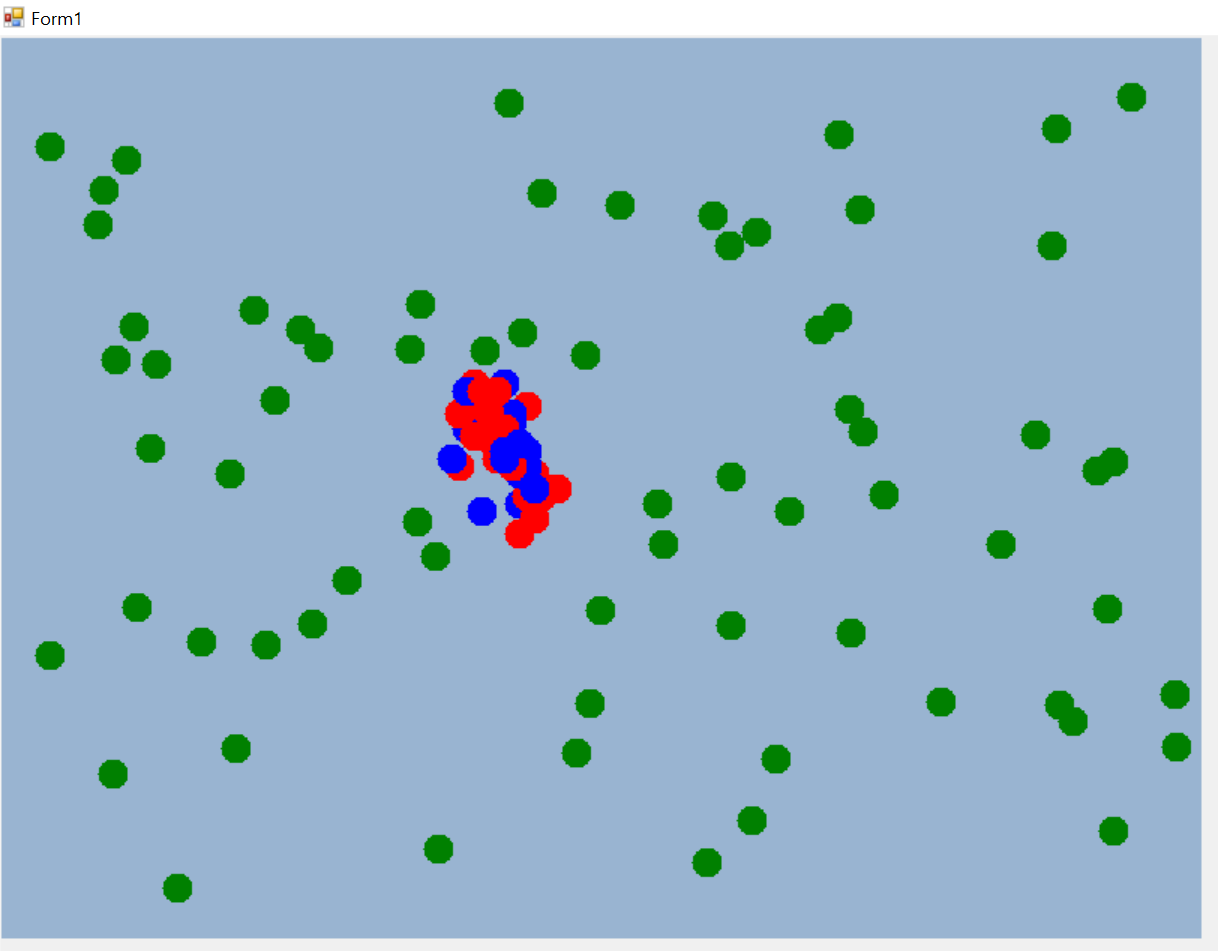


Рисунок 2 – игровое поле в процессе игры

Изображение выглядит как держит, еда, белый

Автоматически созданное описание

Рисунок 3 – игровое поле в процессе игры

Каждые 100 итераций на игровом поле заново расставляется еда. Игра заканчивается, когда количество существ достигло определенного количества. После этого выводится статистика прошедшей игры как на рисунке 4. В ней есть информация о длительности прошедшей игры, количестве женских и мужских особ и количестве съеденной еды:

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 4 – статистика прошедшей игры

**Вывод:** была написана игра «Жизнь», в которой было изначально две особи и случайное количество еды, распределенной по полю. При их пересечении появляются новые особи, в зависимости от количества оставшихся итераций. Игра заканчивается при достижении определенного количества особей, после чего выводится статистика прошедшей игры.