Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

|  |
| --- |
| Институт информационных технологий и анализа данных |

наименование института

|  |
| --- |
| Допускаю к защите  Руководитель:  З. А. Бахвалова |
| И.О. Фамилия |

Разработка системы управления частицами на языке C#

наименование темы

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине

|  |
| --- |
| Технологии программирования |

|  |
| --- |
| 1.003.00.00 ПЗ |
| обозначение документа |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | ИСТб 21-2 |  |  |  | Зимин Н. В. |
|  | шифр группы |  | подпись |  | Фамилия И.О |
| Нормоконтроль |  |  |  |  | Бахвалова З. А. |
|  |  |  | подпись |  | Фамилия И.О |

|  |  |
| --- | --- |
| Курсовая работа защищена с оценкой |  |

Иркутск 2023 г.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**ЗАДАНИЕ**

**НА КУРСОВУЮ РАБОТУ**

|  |  |  |
| --- | --- | --- |
| По курсу | Технологии программирования | |
| Студенту | Зимину Никиту Витальевичу | |
|  | (фамилия, инициалы) | |
| Тема работы: | Разработка системы управления частицами на языке C# | | |
| Исходные данные: | | Варианты 1, 2, 5, 6 | |
| Рекомендуемая литература: | | | |
| 1. Троелсен, Эндрю Язык программирования C# 5.0 и платформа .NET 4.5 / Эндрю Троелсен. - М.: Вильямс, 2015. - 486 c  2. Система частиц, часть 1. [Электронный ресурс] // aqua.tealeaf.su : чаинка, 2021. URL: http://aqua.tealeaf.su/particle-system.html (дата обращения: 01.12.2021). | | | |
|  | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Графическая часть на |  | | листах. | |
| Дата выдачи задания |  | 23 / 11 / 2022 г. | | |
| Задание получил студент | | | |  | |  | Зимин Н. В. |
|  | | | | подпись | |  | Фамилия И.О. |

|  |  |
| --- | --- |
| Дата представления работы руководителю | 21 / 12 / 2022 г. |
| Руководитель курсовой работы |  |  | Бахвалова З. А. |
|  | подпись | Фамилия И.О. |

# ****Содержание****

[Содержание 3](#_Toc124184373)

[Введение 4](#_Toc124184374)

[1 Индивидуальный вариант 5](#_Toc124184375)

[2 Внешний вид главного окна 6](#_Toc124184376)

[3 Код эмиттера 6](#_Toc124184377)

[4 Код зоны смерти частиц 10](#_Toc124184378)

[5 Код зоны перекраски частиц 11](#_Toc124184379)

[6 Код частиц 12](#_Toc124184380)

[7 Код формы 14](#_Toc124184381)

[8 Результаты работы приложения 18](#_Toc124184382)

[Заключение 21](#_Toc124184383)

[Список использованной литературы 22](#_Toc124184384)

# 

# ****Введение****

В данной работе необходимо разработать графический интерфейс для управления системой частиц.

Система частиц — используемый в компьютерной графике способ представления объектов, не имеющих чётких геометрических границ (различные облака, туманности, взрывы, струи пара, шлейфы от ракет, дым, снег, дождь и т. п.).

Система частиц состоит из определённого (фиксированного или произвольного) количества частиц. Математически каждая частица представляется как материальная точка с дополнительными атрибутами, такими как внешний вид реализуемого с помощью метода render, скоростью, запасом жизни и т. п.

В ходе работы программы каждая частица изменяет своё состояние по определённому, общему для всех частиц системы, закону. Например, частица может подвергаться воздействию гравитации, менять размер, цвет, скорость и так далее, и, после проведения всех расчётов, частица визуализируется.

Новые частицы испускаются так называемым «эмиттером». Эмиттером может быть точка, тогда новые частицы будут возникать в одном месте. Так можно смоделировать, например, взрыв: эмиттером будет его центр. Эмиттером может быть отрезок прямой или плоскость: например, частицы дождя или снега должны возникать на высоко расположенной горизонтальной плоскости. Эмиттером может быть и произвольный геометрический объект: в этом случае новые частицы будут возникать на всей его поверхности.

Для реализации используется язык C#. C# – это язык программирования, предназначенный для разработки самых разнообразных приложений. Язык C# прост, строго типизирован и объектно-ориентирован. Благодаря множеству нововведений C# обеспечивает возможность быстрой разработки приложений, но при этом сохраняет выразительность и элегантность, присущую си подобным языкам.

# 1 Индивидуальный вариант

Реализовать управление эмиттером, добавить trackbars для изменения направления кругового эмиттера, количества частиц в тик для эмиттера, максимальной длительности жизни частиц, таймер до конца игры, движение кругового эмиттера по оси X, .

Дополнительно

1. Реализовать зоны перекрашивания частиц, попадая в радиус действия которой частицы меняют свой цвет на цвет указанный у точки.
2. Реализовать точку-счетчик частиц, попадая в которую частица умирает, а на точке пишется сколько частиц она уже собрала.
3. Реализовать движение кругового эмиттера.
4. Реализовать перемещение зоны перекрашивания частиц и смерти частиц через trackbars.
5. Реализовать таймер до конца игры и при её окончании выводящий сообщение о окончании игры и количестве набранных очков.

# 2 Внешний вид главного окна

На рисунке 2.1 представлен внешний вид главного окна графического приложения.

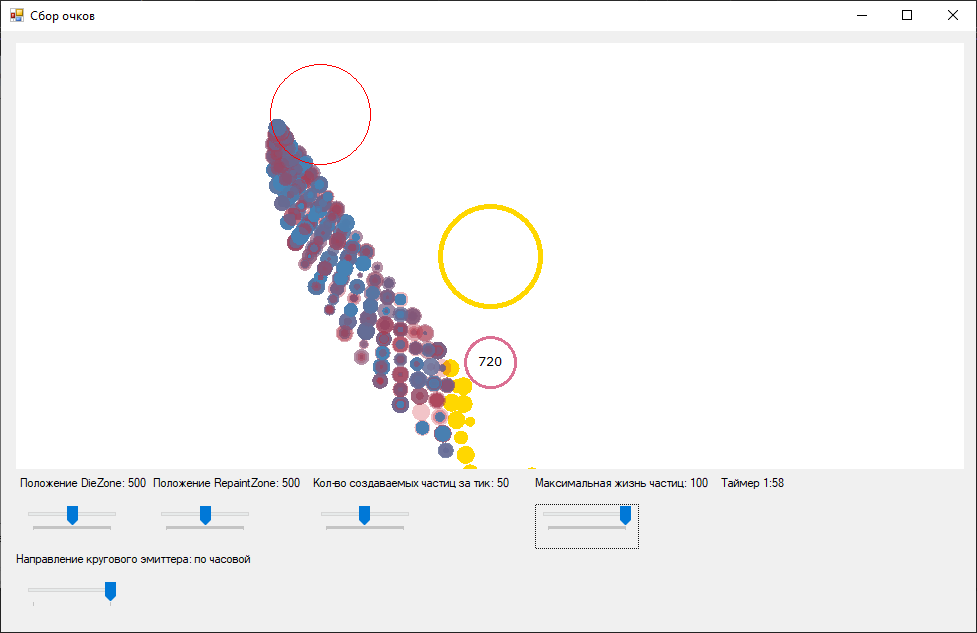


Рисунок 2.1 – внешний вид главного окна

На главном окне присутствуют такие элементы как:

* Игровая зона, самый большой объект;
* CircleEmitter создающий частицы на перемещающейся точке своей окружности и двигающийся от края игровой зоны к противоположному краю игровой зоны
* RepaintZone, перекрашивающий попадающие в неё частицы
* DieZone, поглощающая попавшие в неё частицы и подсчитывающие очки за каждую поглощённую частицы, обычная частица считается за 1 очко, перекрашенная частица считается за 5 очков
* trackBar «Положение DieZone: 500» отвечающий за положение зоны смерти частиц по оси X;
* trackBar «Положение RepaintZone: 500» отвечающий за положение зоны перекраски частиц по оси X;
* trackBar «Кол-во создаваемых частиц за тик: 50» отвечающий за количество частиц, создаваемых эмиттером за каждый тик таймера;
* trackBar «Максимальная жизнь частиц: 100» отвечающий за максимальное количество жизней с которым частицы могут создаться;
* trackBar «Направление кругового эмиттера: по часовой» отвечающий за направление движения точки генерации частиц на круговом эмиттере;
* label «Таймер» отвечающий за отображение времени до конца игры

# 3 Код эмиттера

internal class Emitter

{

public List<Particle> particles = new List<Particle>();

public List<IImpactPoint> impactPoints = new List<IImpactPoint>();

public int MousePositionX;

public int MousePositionY;

public float GravitationX = 0;

public float GravitationY = 1;

public int ParticlesCount = 500;

public int X; // координата X центра эмиттера

public int Y; // координата Y центра эмиттера

public int Direction = 0; // вектор направления в градусах куда эмиттер выпускает частицы

public int Spreading = 360; // разброс частиц относительно Direction

public int SpeedMin = 1; // начальная минимальная скорость движения частицы

public int SpeedMax = 10; // начальная максимальная скорость движения частицы

public int RadiusMin = 2; // минимальный радиус частицы

public int RadiusMax = 10; // максимальный радиус частицы

public int LifeMin = 20; // минимальное время жизни частицы

public int LifeMax = 100; // максимальное время жизни частицы

public int ParticlePerTick = 5; //создаваемые частицы за один тик таймера

public Color ColorFrom = Color.SteelBlue; // начальный цвет частицы

public Color ColorTo = Color.FromArgb(0, Color.Red); // конечный цвет частиц

public virtual void UpdateState() // метод обновления частиц эмиттера

{

int particleToCreate = ParticlePerTick;

foreach (var particle in particles)

{

if (particle.Life <= 0)

{

if (particleToCreate > 0)

{

particleToCreate -= 1;

particle.FromColor = ColorFrom;

particle.ToColor = ColorTo;

ResetParticle(particle);

}

}

else

{

particle.Life--;

particle.X += particle.SpeedX;

particle.Y += particle.SpeedY;

foreach (var point in impactPoints)

{

point.ImpactParticle(particle);

}

particle.SpeedX += GravitationX;

particle.SpeedY += GravitationY;

}

}

while(particleToCreate >= 1)

{

particleToCreate -= 1;

var particle = CreateParticle();

ResetParticle(particle);

particles.Add(particle);

}

}

public virtual void ResetParticle(Particle particle) // метод обновления одной частицы

{

particle.FromColor = ColorFrom;

particle.ToColor = ColorTo;

particle.Life = Particle.random.Next(LifeMin, LifeMax);

particle.X = X;

particle.Y = Y;

var direction = Direction + (double)Particle.random.Next(Spreading) - Spreading/2;

var speed = 1 + Particle.random.Next(SpeedMin, SpeedMax);

particle.SpeedX = (float)(Math.Cos(direction / 180 \* Math.PI) \* speed);

particle.SpeedY = -(float)(Math.Sin(direction / 180 \* Math.PI) \* speed);

particle.Radius = 2 + Particle.random.Next(RadiusMin, RadiusMax);

}

public virtual void Render(Graphics g) // метод рисовки графики частиц

{

foreach (var particle in particles)

{

particle.Draw(g);

}

foreach (var point in impactPoints)

{

point.Render(g);

}

}

public virtual Particle CreateParticle() // метод создания частиц

{

var particle = new ParticleColorful();

particle.FromColor = ColorFrom;

particle.ToColor = ColorTo;

return particle;

}

}

internal class CircleEmitter : Emitter

{

public int Radius = 100;

public bool MovingRight;

public int MovingAmount;

public int DirectionSpeed;

public override void ResetParticle(Particle particle) // метод обновления одной частицы

{

particle.Life = 20 + Particle.random.Next(LifeMin, LifeMax);

var direction = Direction + (double)Particle.random.Next(Spreading) - Spreading / 2;

particle.X = (float)(X + Radius \* Math.Cos(direction / 180 \* Math.PI) / 2);

particle.Y = (float)(Y + Radius \* Math.Sin(direction / 180 \* Math.PI) / 2);

var speed = Particle.random.Next(SpeedMin, SpeedMax);

particle.SpeedX = (float)(Math.Cos(direction / 180 \* Math.PI) \* speed);

particle.SpeedY = (float)(Math.Sin(direction / 180 \* Math.PI) \* speed);

particle.Radius = Particle.random.Next(RadiusMin, RadiusMax);

}

public override void UpdateState()// метод обновления частиц эмиттера и движения эмиттера по оси X

{

base.UpdateState();

if(MovingRight)

{

X += 5;

MovingAmount += 5;

}

else

{

X -= 5;

MovingAmount -= 5;

}

if (MovingAmount == 300)

{

MovingRight = false;

}

if (MovingAmount == -300)

{

MovingRight = true;

}

Direction += DirectionSpeed;

}

public override void Render(Graphics g)

{

base.Render(g);

g.DrawEllipse(

new Pen(Color.Red),

X - Radius / 2,

Y - Radius / 2,

Radius,

Radius

);

}

}

# 4 Код зоны смерти частиц

internal class DieZone : IImpactPoint //класс зоны при попадании в которую частицы умирают и увеличивается счётчик зоны

{

public Color Color = Color.DarkBlue;

public int amount = 0;

public override void ImpactParticle(Particle particle) //метод проверяющий пересечение с частицами и то, перекрашены ли они

{

float gX = X - particle.X;

float gY = Y - particle.Y;

double r = Math.Sqrt(gX \* gX + gY \* gY);

if (r < Radius / 2)

{

if (particle.FromColor == Color.Gold)

{

amount += 5;

}

else

{

amount++;

}

particle.FromColor = Color.White;

particle.ToColor = Color.White;

particle.Life = 0;

}

}

public override void Render(Graphics g)

{

g.DrawEllipse(

new Pen(color, 3),

X - Radius / 2,

Y - Radius / 2,

Radius,

Radius

);

var stringFormat = new StringFormat();

stringFormat.Alignment = StringAlignment.Center;

stringFormat.LineAlignment = StringAlignment.Center;

g.DrawString(

$"{amount}",

new Font("Verdana", 10),

new SolidBrush(Color.Black),

X,

Y,

stringFormat

);

}

}

# 5 Код зоны перекраски частиц

internal class RepaintZone : IImpactPoint //класс зоны, перекрашивающей частицы

{

public override void ImpactParticle(Particle particle) // метод проверяющий пересечение с частицами и перекрашивающий их

{

float gX = X - particle.X;

float gY = Y - particle.Y;

double r = Math.Sqrt(gX \* gX + gY \* gY);

if (r < Radius / 2)

{

particle.FromColor = color;

particle.ToColor = color;

}

}

public override void Render(Graphics g)

{

g.DrawEllipse(

new Pen(color, 5),

X - Radius / 2,

Y - Radius / 2,

Radius,

Radius

);

}

}

# 6 Код частиц

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ZiminN\_ISTb\_21\_2\_particl

{

internal class Particle

{

public Color FromColor;

public Color ToColor;

public float Life;

public int Radius;

public float X;

public float Y;

public float SpeedX;

public float SpeedY;

public static Random random = new Random();

public Particle() // конструктор

{

var direction = (double)random.Next(360);

var speed = 1 + random.Next(10);

SpeedX = (float)(Math.Cos(direction / 180 \* Math.PI) \* speed);

SpeedY = -(float)(Math.Sin(direction / 180 \* Math.PI) \* speed);

Radius = 2 + random.Next(10);

Life = 20 + random.Next(100);

}

public virtual void Draw(Graphics g) // метод рисовки графики частицы

{

float koef = Math.Min(1f, Life / 100);

int alpha = (int)(koef \* 255);

var color = Color.FromArgb(alpha, Color.Black);

var brush = new SolidBrush(color);

g.FillEllipse(brush, X - Radius, Y - Radius, Radius \* 2, Radius \* 2);

brush.Dispose();

}

public class ParticleColorful : Particle

{

public static Color MixColor(Color color1, Color color2, float koef) // метод смешивания цветов

{

return Color.FromArgb(

(int)(color2.A \* koef + color1.A \* (1 - koef)),

(int)(color2.R \* koef + color1.R \* (1 - koef)),

(int)(color2.G \* koef + color1.G \* (1 - koef)),

(int)(color2.B \* koef + color1.B \* (1 - koef))

);

}

public override void Draw(Graphics g) // метод рисовки цветной частицы

{

float koef = Math.Min(1f, Life / 100);

koef = Math.Max(0, koef);

var color = MixColor(ToColor, FromColor, koef);

var brush = new SolidBrush(color);

g.FillEllipse(brush, X - Radius, Y - Radius, Radius \* 2, Radius \* 2);

brush.Dispose();

}

}

}

}

# 7 Код формы

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 ZiminN\_ISTb\_21\_2\_particl.MyEmitter;

using ZiminN\_ISTb\_21\_2\_particl.MyParticle;

using static ZiminN\_ISTb\_21\_2\_particl.Particle;

namespace ZiminN\_ISTb\_21\_2\_particl

{

public partial class Form1 : Form

{

int miliseconds = 120000;

List<Emitter> emitters = new List<Emitter>();

List<RepaintZone> repaintZones = new List<RepaintZone>();

RepaintZone repaintZone;

DieZone dieZone;

CircleEmitter circleEmitter;

public Form1() // конструктор

{

InitializeComponent();

pictureBox1.Image = new Bitmap(pictureBox1.Width, pictureBox1.Height);

this.circleEmitter = new CircleEmitter

{

Direction = 0,

Spreading = 1,

SpeedMin = 2,

SpeedMax = 10,

ColorFrom = Color.SteelBlue,

ColorTo = Color.FromArgb(0, Color.Red),

ParticlePerTick = 50,

DirectionSpeed = 5,

X = pictureBox1.Width / 2,

Y = pictureBox1.Height / 2 - pictureBox1.Height / 3,

};

emitters.Add(this.circleEmitter);

this.repaintZone = new RepaintZone

{

color = Color.Gold,

Radius = 100,

X = pictureBox1.Width / 2,

Y = pictureBox1.Height / 2

};

this.dieZone = new DieZone

{

Color = Color.Red,

Radius = 50,

X = pictureBox1.Width / 2,

Y = pictureBox1.Height / 2 + pictureBox1.Height / 4

};

foreach (var emitter in emitters)

{

emitter.impactPoints.Add(repaintZone);

emitter.impactPoints.Add(dieZone);

}

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

private void timer1\_Tick(object sender, EventArgs e) // метод обновления игры, происходящий каждый тик тактового генератора

{

using (var g = Graphics.FromImage(pictureBox1.Image))

{

g.Clear(Color.White);

}

foreach (var emitter in emitters)

{

emitter.UpdateState();

using (var g = Graphics.FromImage(pictureBox1.Image))

{

emitter.Render(g);

}

}

miliseconds -= 40;

labelTimer.Text = $"Таймер {miliseconds / 1000 / 60}:{miliseconds / 1000 % 60}";

if (miliseconds <= 0)

{

timer1.Stop();

MessageBox.Show(

$"Собранные очки = {dieZone.amount}",

"Игра закончена");

}

pictureBox1.Invalidate();

}

private void pictureBox1\_MouseMove(object sender, MouseEventArgs e)

{

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void label2\_Click(object sender, EventArgs e)

{

}

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

{

}

private void trackBarLife\_Scroll(object sender, EventArgs e) //метод задающий максимальную жизнь частиц через trackBar

{

foreach (var emitter in emitters)

{

emitter.LifeMax = trackBarLife.Value;

labelParticleMaxLife.Text = $"Максимальная жизнь частиц: {emitter.LifeMax}";

}

}

private void trackBarTic\_Scroll(object sender, EventArgs e) //метод задающий количество создаваемых частиц за один тик через trackBar

{

foreach (var emitter in emitters)

{

emitter.ParticlePerTick = trackBarTic.Value;

labelParticlePerTic.Text = $"Кол-во создаваемых частиц за тик: {emitter.ParticlePerTick}";

}

}

private void pictureBox1\_MouseClick(object sender, MouseEventArgs e)

{

}

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

{

}

private void trackBarCircleEmitterDirection\_Scroll(object sender, EventArgs e) //метод задающий направление создания частиц кругового эмиттера через trackBar

{

foreach (var emitter in emitters)

{

if(emitter is CircleEmitter)

{

if (trackBarCircleEmitterDirection.Value == 0)

{

(emitter as CircleEmitter).DirectionSpeed = ((emitter as CircleEmitter).DirectionSpeed) \* (-1);

labelCircleEmitterDirection.Text = "Направление кругового эмиттера: против часовой";

}

else

{

(emitter as CircleEmitter).DirectionSpeed = Math.Abs(((emitter as CircleEmitter).DirectionSpeed));

labelCircleEmitterDirection.Text = "Направление кругового эмиттера: по часовой";

}

}

}

}

private void trackBarDieZonePosition\_Scroll(object sender, EventArgs e) //метод перемещающий DieZone по оси X через trackBar

{

foreach (var emitter in emitters)

{

dieZone.X = pictureBox1.Width / 100 \* trackBarDieZonePosition.Value;

}

labelDieZonePosition.Text = $"Положение DieZone: {dieZone.X}";

}

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

{

}

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

{

}

private void trackBarRepaintZonePosition\_Scroll(object sender, EventArgs e) //метод перемещающий RepaintZone по оси X через trackBar

{

foreach (var emitter in emitters)

{

repaintZone.X = pictureBox1.Width / 100 \* trackBarRepaintZonePosition.Value;

}

labelRepaintZonePosition.Text = $"Положение RepaintZone: {repaintZone.X}";

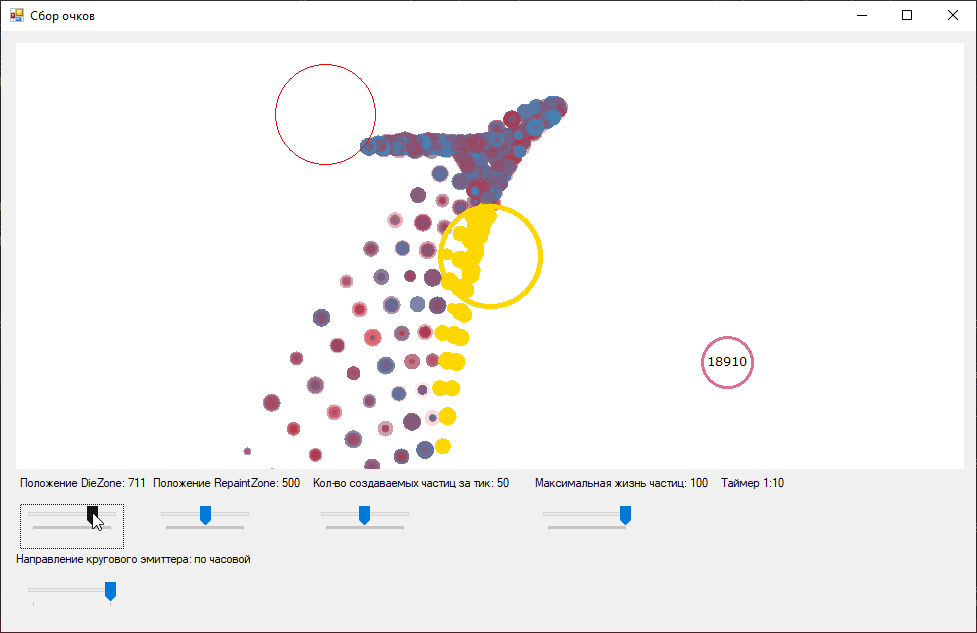
}

}

}

# 8 Результаты работы приложения

Изменение «**Положение DieZone»**

  
Рисунок 7.1 – Передвижение зоны смерти частиц

При изменении «Положение DieZone» зона смерти частиц перемещается по игровой зоне по оси X в соответствии с изменением trackBar.

Изменение «**Положение RepaintZone**»



Рисунок 7.2 – Передвижение зоны перекраски частиц

При изменении «Положение RepaintZone» зона перекраски частиц перемещается по игровой зоне по оси X в соответствии с изменением trackBar.

Изменение направления генерации частиц кругового эмиттера

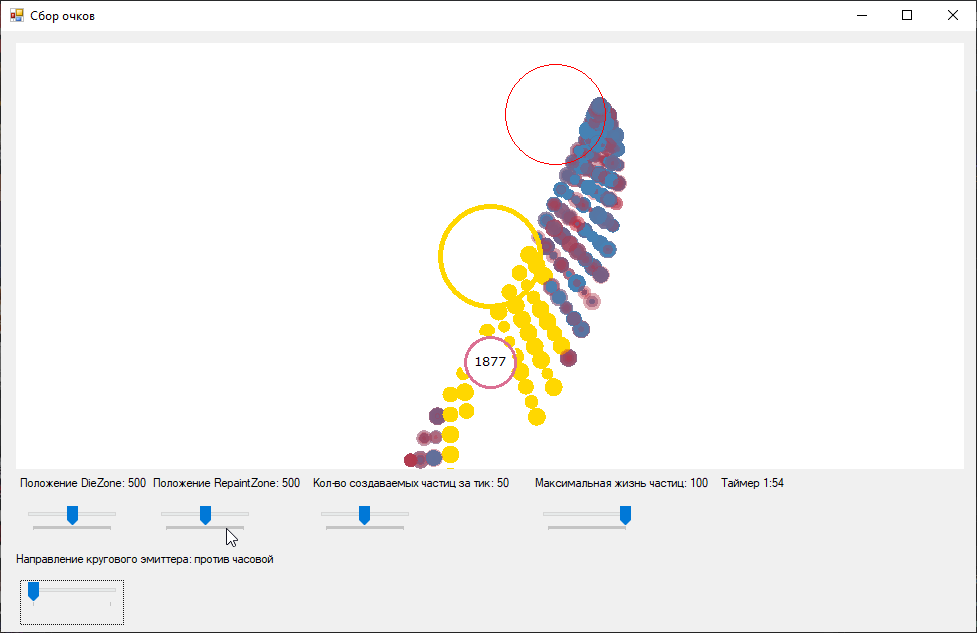


Рисунок 7.3 – Изменение генерации частиц кругового эмиттера на «против часовой»

При смене направления генерации частиц кругового эмиттера изменяется направление движение по круговому эмиттеру при генерации частиц и направление их вылета при генерации.

Изменение «**Кол-во создаваемых частиц за тик**»

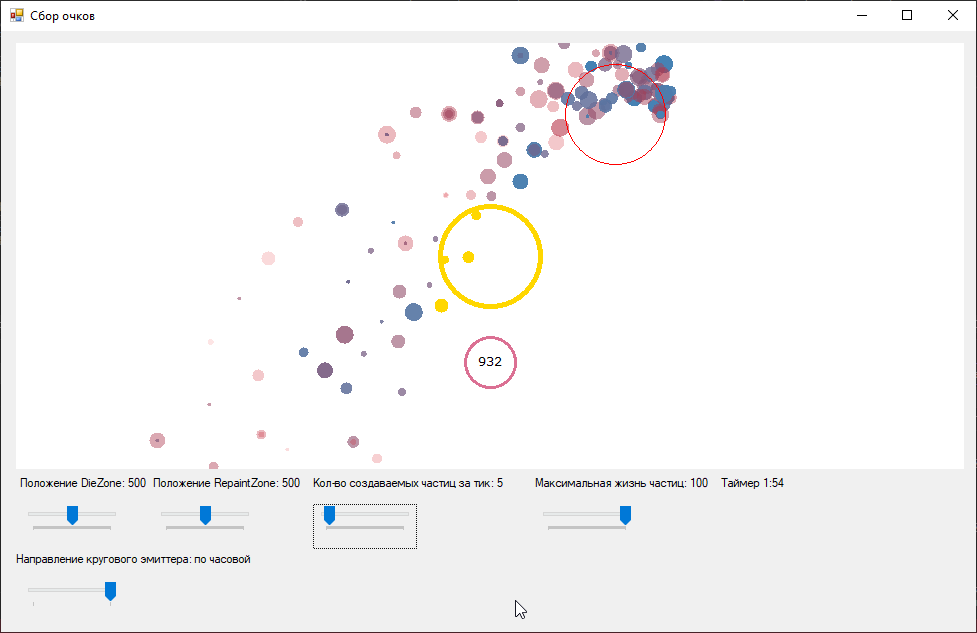


Рисунок 7.4 – Уменьшение кол-ва создаваемых частиц за тик

При изменении «Кол-во создаваемых частиц за тик» увеличивается/уменьшается количество частиц, генерируемых круговым эмиттером за тик таймера в соответствии с trackBar.

Изменение «**Максимальная жизнь частиц**»

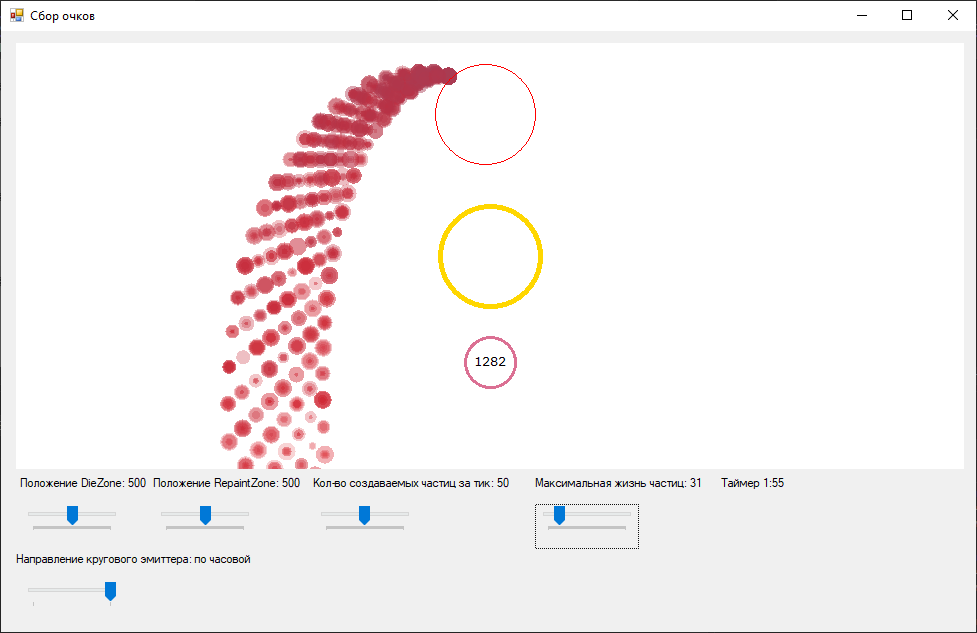


Рисунок 7.5 – Уменьшение максимальной возможной жизни частиц

При изменении «Максимальная жизнь частиц» увеличивается/уменьшается значение максимальной возможной жизни с которой могут генерироваться частицы в соответствии с trackBar.

# Заключение

В рамках курсовой работы было разработано приложение, позволяющее управлять системой частиц. В приложении реализованы следующие возможности:

* Специальный эмиттер «**CircleEmitter**» создающий частицы по окружности.
* Специальная точка «**RepaintZone**», которая перекрашивает попавшие в неё частицы.
* Специальная точка «**DieZone**», которая поглощает попавшие в неё частицы и подсчитывает их количество с учётом перекраски частиц.
* Изменение «**Положение DieZone**», для управления зоной смерти частиц для набора ей очков.
* Изменение «**Положение RepaintZone**», для управления зоной перекраски частиц для увеличения их ценности при подсчёте очков.
* Изменение «**Направление кругового эмиттера**» меняющий направление создания частиц кругового эмиттера.
* Изменение «**Кол-во создаваемых частиц**», для управления количеством частиц создаваемых за тик таймера.
* Изменение «**Максимальная жизнь частиц**», для управления максимальным возможным количеством жизней с которым создаются частицы.
* Таймер при окончании которого выводится сообщение о завершении игры и количество набранных очков.

Приложение было всесторонне протестировано и отлажено. Все поставленные в рамках курсовой работы задачи были выполнены.

В рамках выполнения данной работы был получен обширный спектр сведений об объектно-ориентированном языке программирования C#, разработки графических приложений и эмуляции сложных динамических систем.

Также результат работы можно посмотреть по ссылке - <https://github.com/CrusaderKleptoman/ZiminN_ISTb-21-2_particl.git>

# Список использованной литературы

1. Троелсен, Эндрю Язык программирования C# 5.0 и платформа .NET 4.5 / Эндрю Троелсен. - М.: Вильямс, 2015. - 486 c.
2. Вагнер, Билл С# Эффективное программирование / Билл Вагнер. - М.: ЛОРИ, 2013. - 320 c.
3. Ишкова, Э. А. Самоучитель С#. Начала программирования / Э.А. Ишкова. - М.: Наука и техника, 2013. - 496 c.
4. Биллинг В.А. Основы программирования на C#. - Т.: Интернет-университет информационных технологий, Бином, 2012. - 488 с.
5. Евдокимов П.В. C# на примерах. - М.: Наука и техника,2016. - 304 с.
6. Нейгел К., Ивьен Б., Глинн Д., Уотсон К., Скиннер М. C# 4.0 и платформа .NET 4, 2011.
7. Система частиц, часть 1. [Электронный ресурс] // aqua.tealeaf.su : чаинка, 2021. URL: http://aqua.tealeaf.su/particle-system.html (дата обращения: 19.12.2022).