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:  П.П. Преподаватель |
| И.О. Фамилия |

Разработка системы управления частицами на языке C#

наименование темы

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине

|  |
| --- |
| Технологии программирования |

|  |
| --- |
| 1.001.00.00 ПЗ |
| обозначение документа |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | ИСТб 20-\* |  |  |  | Иванов И.И. |
|  | шифр группы |  | подпись |  | Фамилия И.О |
| Нормоконтроль |  |  |  |  | Препод П.П. |
|  |  |  | подпись |  | Фамилия И.О |

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

Иркутск 2021 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 | Иванову Ивану Ивановичу | |
|  | (фамилия, инициалы) | |
| Тема работы: | Разработка системы управления частицами на языке C# | | |
| Исходные данные: | | Вариант 1 | |
| 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 / 2021 г. | | |
| Задание получил студент | | | |  | |  | Иванов И.И. |
|  | | | | подпись | |  | Фамилия И.О. |

|  |  |
| --- | --- |
| Дата представления работы руководителю | 21 / 12 / 2021 г. |
| Руководитель курсовой работы |  |  | Преподаватель П.П. |
|  | подпись | Фамилия И.О. |

Содержание

[Введение 4](#_Toc58847158)

[1 Индивидуальный вариант 5](#_Toc58847159)

[2 Внешний вид главного окна 6](#_Toc58847160)

[3 Код эмиттера 7](#_Toc58847161)

[4 Код специальных точек 9](#_Toc58847162)

[5 Код частиц 10](#_Toc58847163)

[6 Код формы 11](#_Toc58847164)

[7 Описание работы интерфейса 13](#_Toc58847165)

[Заключение 15](#_Toc58847166)

[Список использованной литературы 16](#_Toc58847167)

# 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 направленным эмиттером:

* добавить trackbar для изменения направления эмиттера
* и для градуса распределения частиц
* и для изменения скорости вылета частиц
* а также для изменения количества частиц за тик
* ну и для продолжительности жизни
* дополнительно выводить количество активных частиц на данный тик

# 2 Внешний вид главного окна

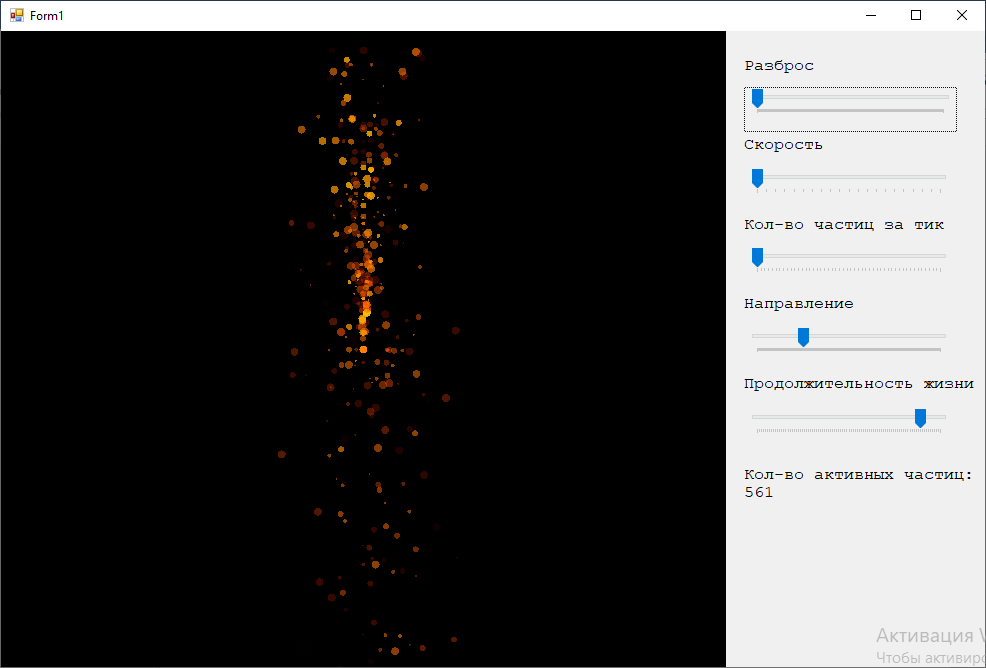


Рисунок 2.1

# 3 Код эмиттера

using System;

using System.Collections.Generic;

using System.Drawing;

namespace CourseWork\_V.Dodonova

{

public class Emitter

{

public List<Particle> particles = new List<Particle>(); // Список частиц

public List<IImpactPoint> impactPoints = new List<IImpactPoint>(); // Список специальных точек

public float GravitationX = 0; // Гравитация по X

public float GravitationY = 1; // Гравитация по Y

public int X; // координата X центра эмиттера, будем ее использовать вместо MousePositionX

public int Y; // соответствующая координата Y

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

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 = 10; // минимальное время жизни частицы

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

public int ParticlesPerTick = 1; // Кол-во частиц за тик

public int ParticlesCount = 0; // Kол-во частиц

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

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

public void UpdateState() // Метод обновления состояния системы

{

int particlesToCreate = ParticlesPerTick; // Добавляю генерацию частиц не больше частиц за тик

ParticlesCount = particles.Count; // Кол-во активных частиц равно кол-во частиц

foreach (var particle in particles)

{

if (particle.life <= 0) // если здоровье кончилось

{

ParticlesCount--; // Уменьшаю кол-во активных частиц

if (particlesToCreate > 0)

{

/\* у нас как сброс частицы равносилен созданию частицы \*/

particlesToCreate -= 1; // поэтому уменьшаем счётчик созданных частиц на 1

ResetParticle(particle); // Вызываю метод респавна частиц (сброс частицы)

}

}

else

{

particle.X += particle.speedX; // Передвигаю частицу по X

particle.Y += particle.speedY; // Передвигаю частицу по Y

particle.life -= 1; // Уменьшаю кол-во здоровья на 1

foreach (var point in impactPoints)

{

point.ImpactParticle(particle);

}

particle.speedX += GravitationX; // Гравитация по X

particle.speedY += GravitationY; // Гравитация по Y

}

}

while (particlesToCreate >= 1) // Пока кол-во

{

particlesToCreate -= 1; // Уменьшаю кол-во частиц

var particle = CreateParticle(); // Создаю частицу

ResetParticle(particle); // Сбрасываю частицу

particles.Add(particle); // Добавляю частицу в список

}

}

public virtual void ResetParticle(Particle particle) // Метод сброса частицы

{

particle.life = Particle.random.Next(LifeMin, LifeMax); // Задаю кол-во здоровья

particle.X = X; // Устанавливаю место генерации частицы по X

particle.Y = Y; // Устанавливаю место генерации частицы по Y

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

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

particle.speedX = (float)(Math.Cos(direction / 180 \* Math.PI) \* speed); // Скорость частиц по X

particle.speedY = -(float)(Math.Sin(direction / 180 \* Math.PI) \* speed); // Скорость частиц по Y

particle.radius = Particle.random.Next(RadiusMin, RadiusMax); // Задаю радиус

}

public virtual Particle CreateParticle() // Метод создания частицы

{

var particle = new ParticleColorful();

particle.FromColor = ColorFrom;

particle.ToColor = ColorTo;

return particle;

}

public void Render(Graphics g) // Отрисовка или рендеринг

{

foreach (var particle in particles) // Отрисовка частиц

{

particle.Draw(g);

}

foreach (var point in impactPoints)

{

point.Render(g); // Рендер частиц

}

}

}

}

# 4 Код специальных точек

// По заданию специальные точки не используются

using System.Drawing;

namespace CourseWork\_V.Dodonova

{

public abstract class IImpactPoint // Этот класс не нужен, но пусть будет

{

public float X; // ну точка же, вот и две координаты

public float Y;

// абстрактный метод с помощью которого будем изменять состояние частиц

// например притягивать или отталкивать

public abstract void ImpactParticle(Particle particle);

// базовый класс для отрисовки точечки

public virtual void Render(Graphics g)

{

g.FillEllipse(

new SolidBrush(Color.Red),

X - 5,

Y - 5,

10,

10

);

}

}

}

# 5 Код частиц

using System;

using System.Drawing;

namespace CourseWork\_V.Dodonova

{

public class Particle

{

public int radius; // Обозначаем радиус частиц

public float X; // X координата положения частицы в пространстве

public float Y; // Y координата положения частицы в пространстве

public float speedX; // Скорость по X

public float speedY; // Скорость по Y

public float life; // Кол-во здоровья частиц

public static Random random = new Random(); // Генератор случайных чисел

public Particle() // конструктор по умолчанию будет создавать кастомную частицу

{

var direction = (double)random.Next(360); // Направление движения

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

speedX = (float)(Math.Cos(direction / 180 \* Math.PI) \* speed); // Скорость по X

speedY = -(float)(Math.Sin(direction / 180 \* Math.PI) \* speed); // Скорость по Y

radius = 2 + random.Next(10); // Радиус частицы

life = 20 + random.Next(100); // Кол-во 'здоровья' частицы

}

public virtual void Draw(Graphics g)

{

float k = Math.Min(1f, life / 100); // Коэффициент прозрачности от 0 до 1.0

int alpha = (int)(k \* 255); // рассчитываем значение альфа канала в шкале от 0 до 255

Color color = Color.FromArgb(alpha, Color.Violet); //Добавили цвет для кисти

var b = new SolidBrush(color); // создали кисть для рисования

// нарисовали залитый кружок радиусом Radius с центром в X, Y

g.FillEllipse(b, X - radius, Y - radius, radius \* 2, radius \* 2);

// удалили кисть из памяти, вообще сборщик мусора рано или поздно это сам сделает

// но документация рекомендует делать это самому

b.Dispose();

}

}

public class ParticleColorful : Particle

{

public Color FromColor; // Начальный цвет

public Color ToColor; // Конечный цвет

public static Color MixColor(Color color1, Color color2, float k) // Задаем цвет от начального к конечному

{

return Color.FromArgb(

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

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

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

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

);

}

public override void Draw(Graphics g) // Отрисовка частиц

{

float k = Math.Min(1f, life / 100); // Задаем коэффициент прозрачности в зависимости от кол-ва здоровья частицы

var color = MixColor(ToColor, FromColor, k); // Добавляем цвет для кисти

var b = new SolidBrush(color); // создали кисть для рисования

// нарисовали залитый кружок радиусом Radius с центром в X, Y

g.FillEllipse(b, X - radius, Y - radius, radius \* 2, radius \* 2);

// удалили кисть из памяти, вообще сборщик мусора рано или поздно это сам сделает

// но документация рекомендует делать это самому

b.Dispose();

}

}

}

# 6

# Код формы

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

namespace CourseWork\_V.Dodonova

{

public partial class Form1 : Form

{

List<Emitter> emitters = new List<Emitter>(); // Список эмиттеров

Emitter emitter; // Сам эмиттер

public Form1()

{

InitializeComponent();

picDisplay.Image = new Bitmap(picDisplay.Width, picDisplay.Height); // Привязка изображения

this.emitter = new Emitter // создаю эмиттер и привязываю его к полю emitter

{

RadiusMin = 1, // Задаю минимальный радиус

RadiusMax = 5, // Задаю максимальный радиус

Direction = 90, // Направление движения

Spreading = 10, // Разброс частицы

SpeedMin = 10, // Минимальная скорость частицы

SpeedMax = 25, // Максимальная скорость частицы

ColorFrom = Color.Gold, // Начальный цвет частицы

ColorTo = Color.FromArgb(0, Color.Red), // Градация цвета частицы от золотого к красному

ParticlesPerTick = 10, // Начальное кол-во частиц за тик

X = picDisplay.Width / 2, // X координата частицы в пространстве (по центру picDisplay.X)

Y = picDisplay.Height / 2, // Y координата частицы в пространстве (по центру picDisplay.Y)

};

emitters.Add(this.emitter); // Добавляю эмиттер в список

}

private void timer\_Tick(object sender, EventArgs e) // Событие для таймера

{

emitter.UpdateState(); // Каждый тик обновляем систему

// Вывод кол-ва активных частиц

lblCountOfActiveParticles.Text = "Кол-во активных частиц: \n" + emitter.ParticlesCount.ToString();

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

{

g.Clear(Color.Black); // Очистка и присвоение picDisplay черного цвета

emitter.Render(g); // Рендер системы

}

picDisplay.Invalidate(); // Обновление picDisplay

}

private void tbCountOfPariclesPerTick\_Scroll(object sender, EventArgs e) // Кол-во частиц в такт или тик

{

emitter.ParticlesPerTick = tbCountOfPariclesPerTick.Value;

}

private void tbDirection\_Scroll(object sender, EventArgs e) // Направление движения частиц

{

emitter.Direction = tbDirection.Value;

}

private void tbLifeOfParticle\_Scroll(object sender, EventArgs e) // Продолжительность жизни частиц

{

emitter.LifeMax = tbLifeOfParticle.Value;

}

private void tbSpreading\_Scroll(object sender, EventArgs e) // Разброс частиц

{

emitter.Spreading = tbSpreading.Value;

}

private void tbSpeed\_Scroll(object sender, EventArgs e) // Скорость движения частиц

{

emitter.SpeedMin = tbSpeed.Value;

}

private void picDisplay\_MouseMove(object sender, MouseEventArgs e) // Дижение эмиттера по нажатию левой кнопки мыши

{

if (e.Button == MouseButtons.Left)

{

emitter.X = e.X;

emitter.Y = e.Y;

}

}

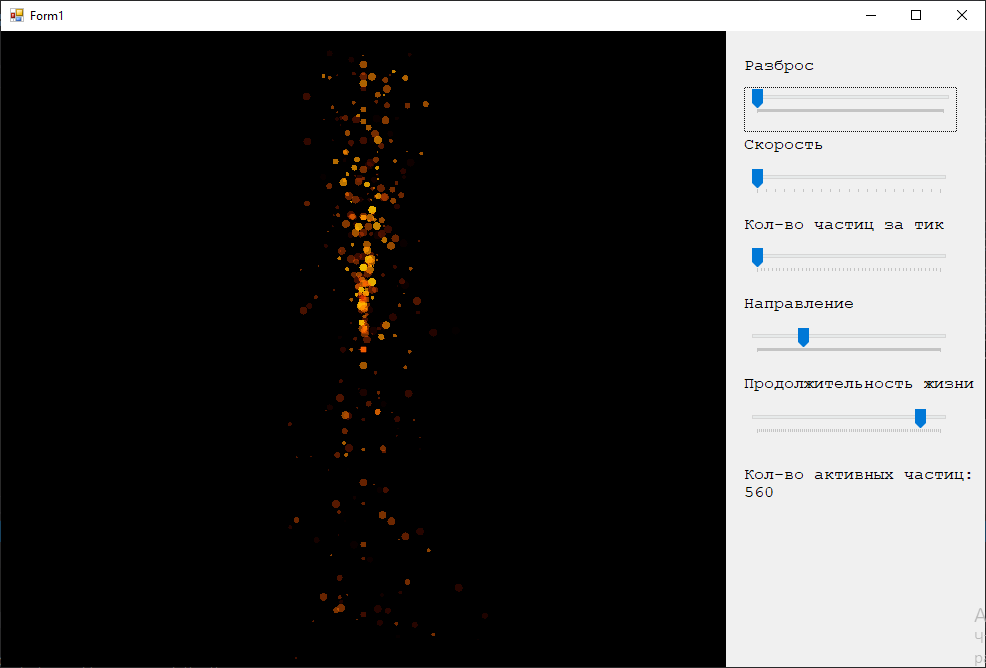
}

}

# 7

# Описание работы интерфейса

Меняя trackback **Скорость** мы меняем скорость движения:

  
Рисунок 7.1

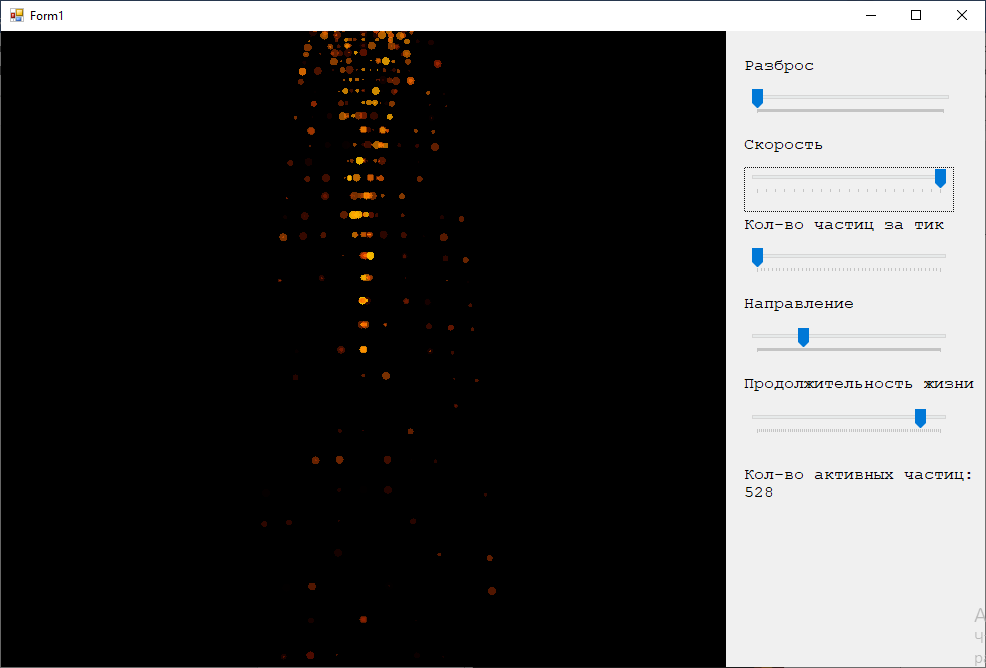


Рисунок 7.2

Меняя trackback **Разброс** мы меняем разброс частиц около заданого направления при генерации:

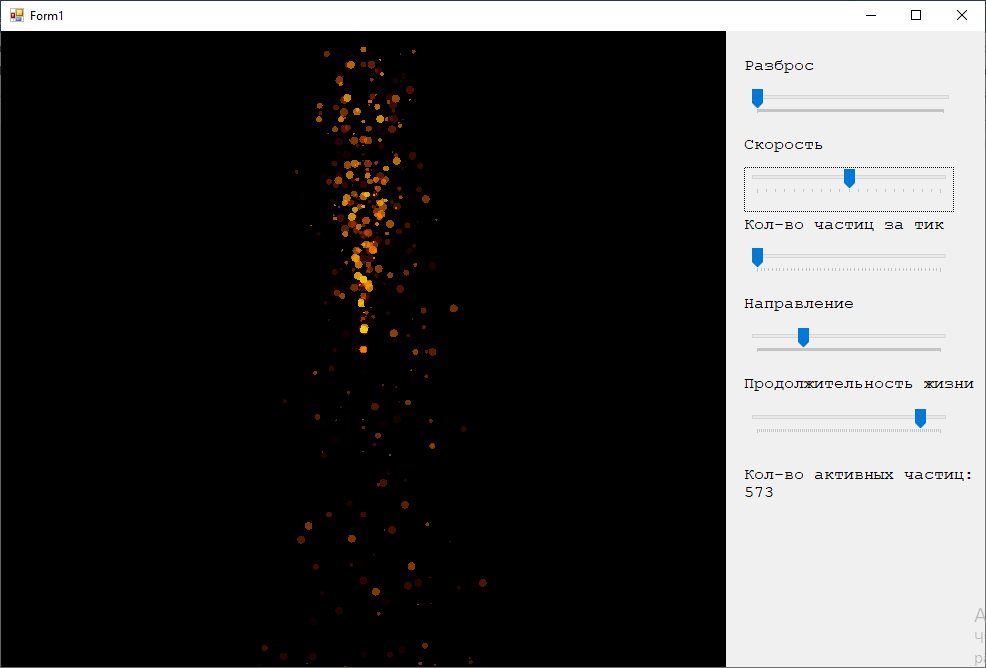


Рисунок 7.3

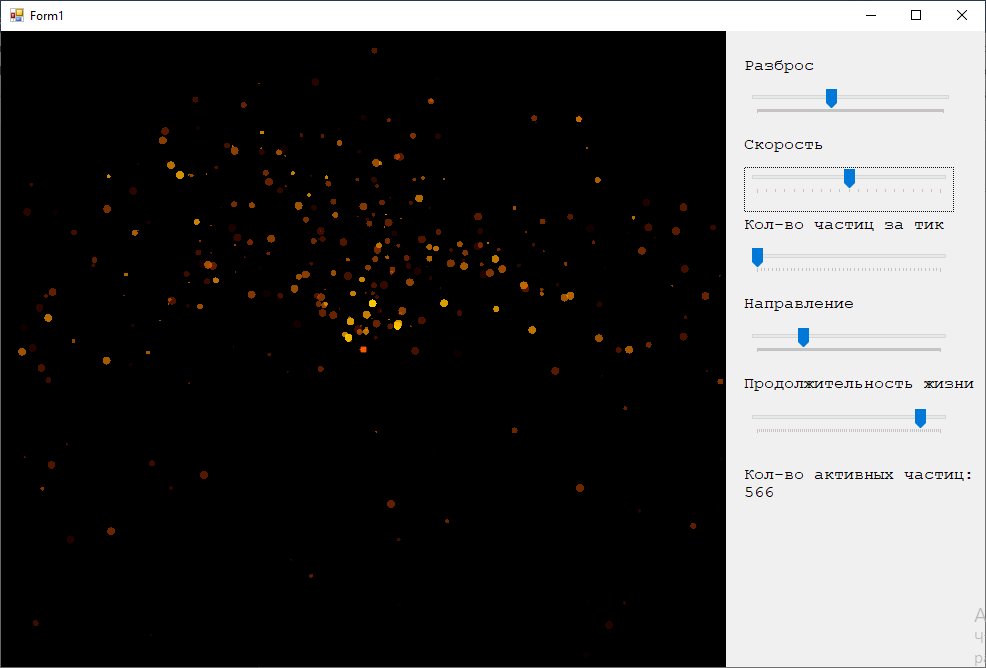


Рисунок 7.4

Меняя trackbar **Кол-во частиц за тик,** мы управляем количеством активных частиц в области генерации, при резком изменении частиц в большую сторону и с последюущем:

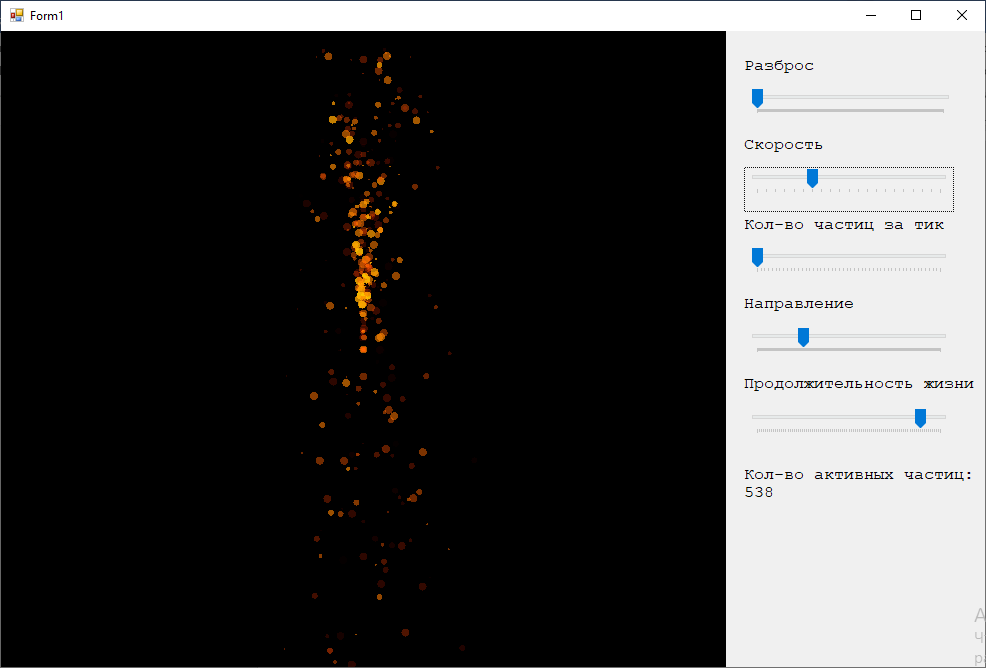


Рисунок 7.5

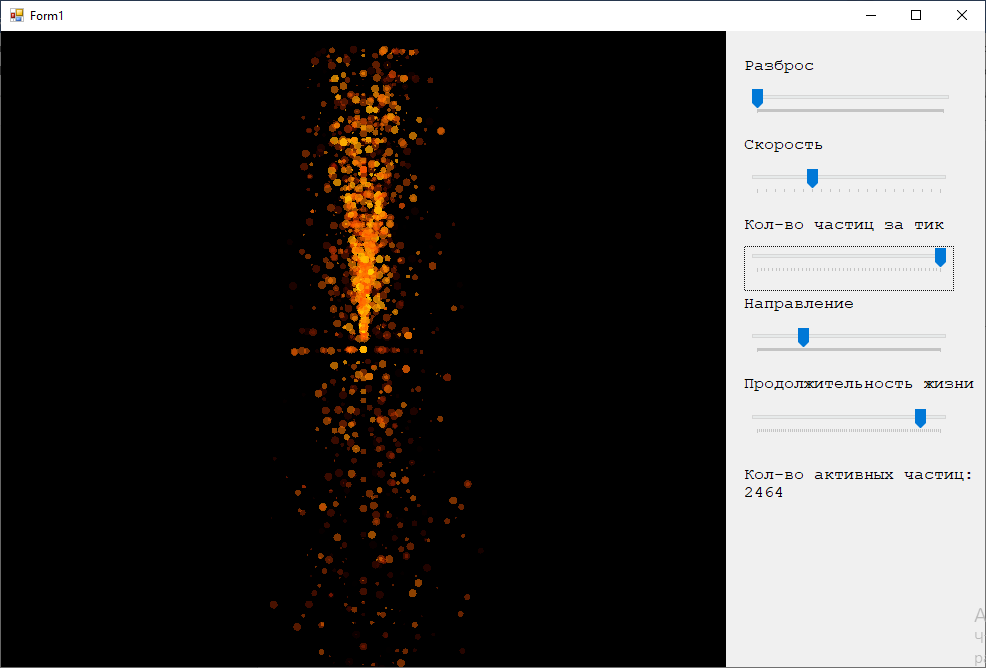


Рисунок 7.6

Меняя trackbar **Направление,** мы управляем направлением движения активных частиц:

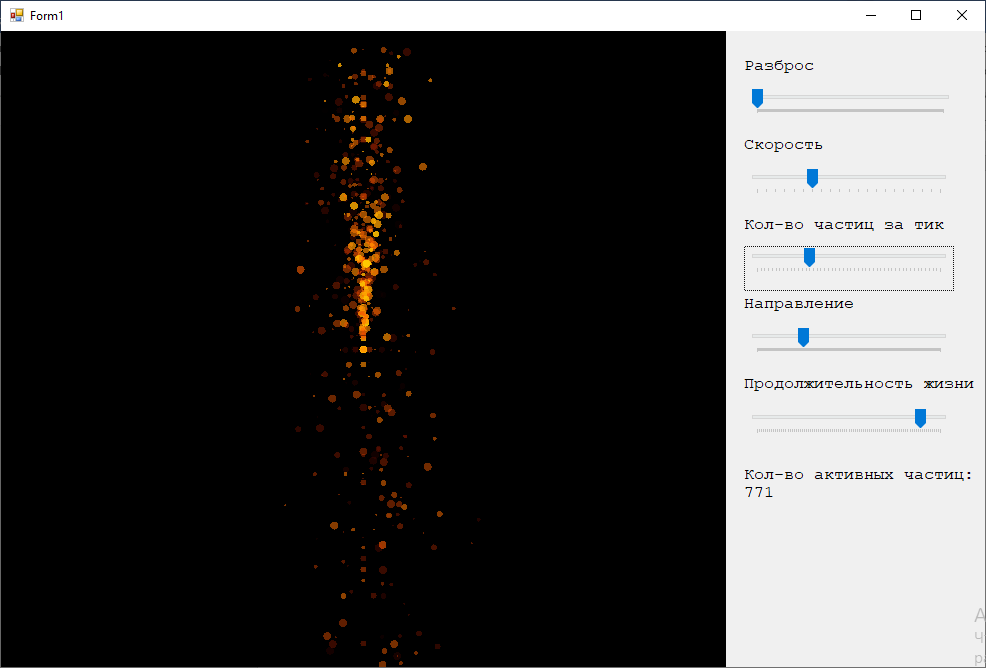


Рисунок 7.7

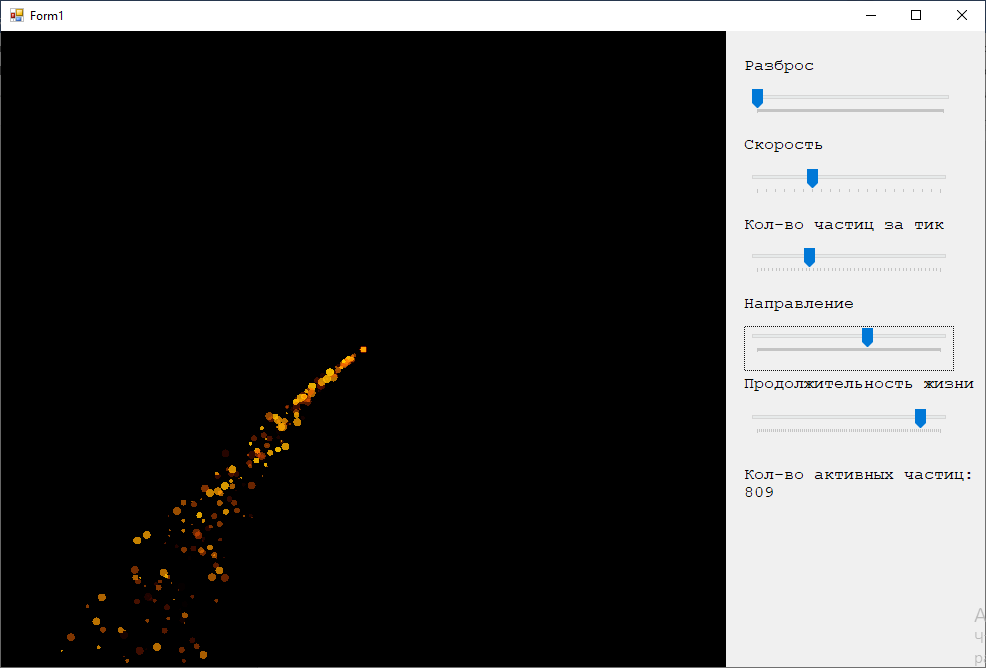


Рисунок 7.8

# Заключение

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

* Ползунок «Разброс», для управления разбросом вылета частиц около заданного направления
* Ползунок «Скорость», для управления скоростью движения частиц
* Ползунок «Кол-во частиц за тик», для управления количеством частиц
* Ползунок «Направление», для управления наклоном движения полета частиц
* Ползунок «Продолжительность жизни», для управления количеством «здоровья» частиц
* Надпись «Кол-во активных частиц», для вывода количества активных частиц

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.

# 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: 01.12.2021).