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:  М.Д. Каташевцев |
| 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 20-2 |  |  |  | Теленьков М.А. |
|  | шифр группы |  | подпись |  | Фамилия И.О |
| Нормоконтроль |  |  |  |  | Каташевцев М.Д. |
|  |  |  | 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 | Теленькову Максиму Алексеевичу | |
|  | (фамилия, инициалы) | |
| Тема работы: | Разработка системы управления частицами на языке C# | | |
| Исходные данные: | | Вариант 24 | |
| 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 | | | |  | |  | Теленьков М.А. |
|  | | | | подпись | |  | Фамилия И.О. |

|  |  |
| --- | --- |
| Дата представления работы руководителю | 13 / 12 / 2021 г. |
| Руководитель курсовой работы |  |  | Каташевцев М.Д. |
|  | подпись | Фамилия И.О. |

Содержание

[Введение 4](#_Toc90313169)

[1 Индивидуальный вариант 5](#_Toc90313170)

[2 Внешний вид главного окна 6](#_Toc90313171)

[3 Код эмиттера 7](#_Toc90313172)

[4 Код специальных точек 10](#_Toc90313173)

[5 Код частиц 12](#_Toc90313174)

[6 Код формы 13](#_Toc90313175)

[7 Описание работы интерфейса 17](#_Toc90313176)

[Заключение 26](#_Toc90313177)

[Список использованной литературы 27](#_Toc90313178)

# 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 – линией

* добавить trackbar для изменения количества частиц в тик, гравитации, максимальной и минимальной продолжительности жизни частицы.
* добавить кнопки для изменения палитры цветов частиц
* дополнительно выводить количество активных частиц на данный тик

Реализовать точки перекрашивания частиц, попадая в радиус действия которой частицы меняют свой цвет на цвет, указанный у точки.

* реализовать раскрашивание частиц падающих с неба
* добавить trackbar с помощью которого можно перемещать точки перекрашивания
* добавить trackbar с помощью которого можно менять радиус точек перекрашивания
* добавить кнопку, которая будет переключать палитру

Реализовать точку-счетчик частиц, попадая в которую частица умирает, а на точке пишется сколько частиц она уже собрала.

* при клике мышкой добавлять новый счетчик
* при клике правой кнопкой мыши, удалять счетчик

# 2 Внешний вид главного окна

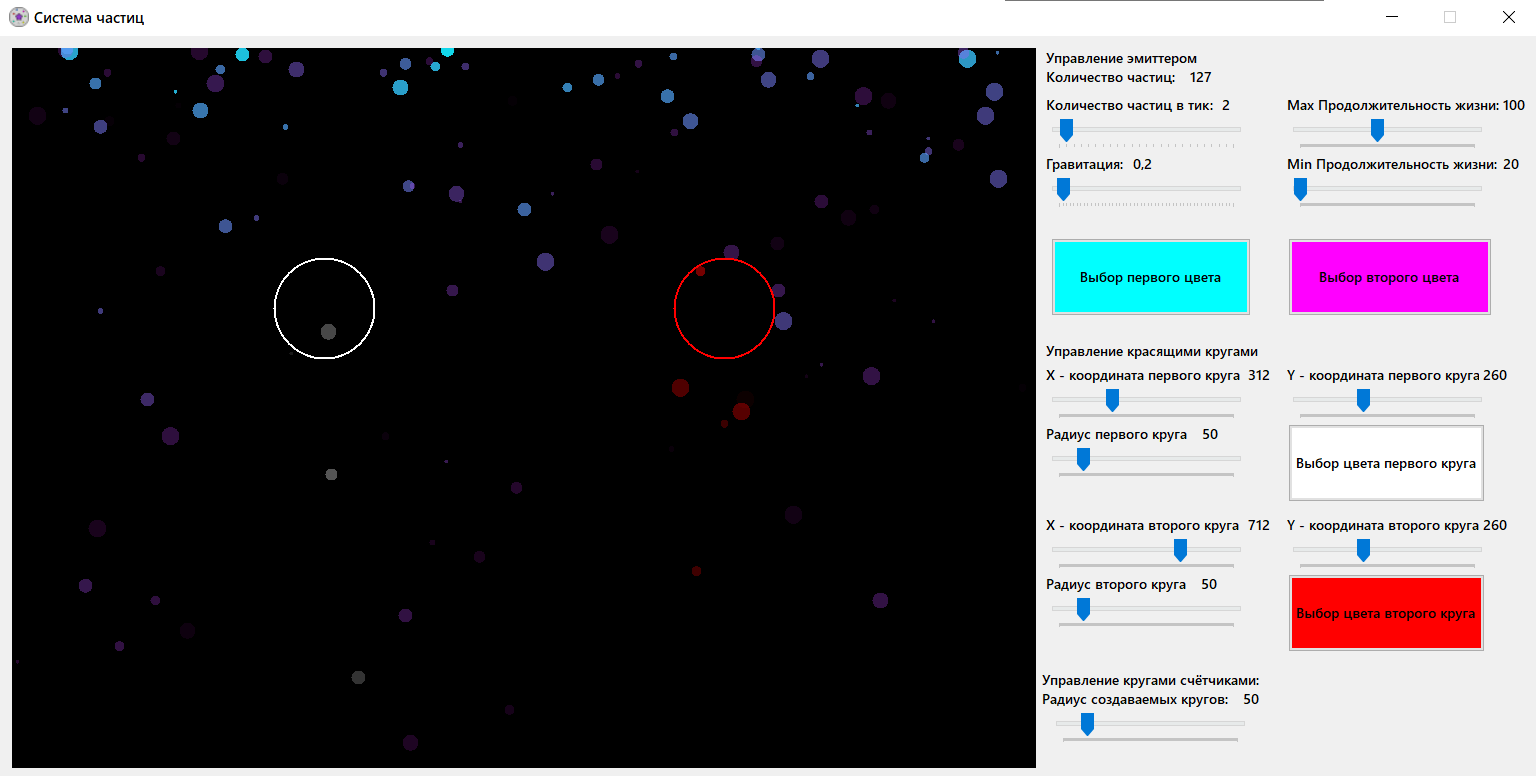
****

Рисунок 1.1

# 3 Код эмиттера

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace Coursework

{

public class Emitter

{

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

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

//Координаты и другие хар-ки эмиттера

public int X = 0;

public int Y = 0;

public int Direction = 0;

public int Spreading = 360;

public int SpeedMin = 1;

public int SpeedMax = 10;

public int RadiusMin = 2;

public int RadiusMax = 10;

public int LifeMin = 20;

public int LifeMax = 100;

public Color ColorFrom = Color.Aqua;

public Color ColorTo = Color.FromArgb(0, Color.Magenta);

public float GravityX = 0;

public float GravityY = 1;

public int ParticlesPerTick = 1;

//Обновляем состояние частиц

public void UpdateState()

{

int particlesToCreate = ParticlesPerTick;

foreach (var particle in particles)

{

if (particle.Life <= 0)

{

if (particlesToCreate > 0)

{

particlesToCreate -= 1;

ResetParticle(particle);

}

}

else

{

particle.SpeedX += GravityX;

particle.SpeedY += GravityY;

particle.Life -= 1;

//Воздействие импактов на частицу

foreach (var point in impactPoints)

{

point.ImpactParticle(particle);

}

particle.X += particle.SpeedX;

particle.Y += particle.SpeedY;

}

}

while (particlesToCreate >= 1)

{

particlesToCreate -= 1;

var particle = CreateParticle();

ResetParticle(particle);

particles.Add(particle);

}

}

//Отрисовываем частицы и impactpoint'ы

public void Render(Graphics g)

{

foreach (var particle in particles)

{

particle.Draw(g);

}

foreach (var point in impactPoints)

{

point.Render(g);

}

}

//Создание отдельной частицы

public virtual Particle CreateParticle()

{

var particle = new Particle();

particle.Color1 = ColorFrom;

particle.Color0 = ColorTo;

return particle;

}

//Переустановка характеристик частицы

public virtual void ResetParticle(Particle particle)

{

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

particle.Color1 = ColorFrom;

particle.Color0 = ColorTo;

particle.X = X;

particle.Y = Y;

float direction = Direction

+ (float)Particle.rand.Next(Spreading)

- Spreading / 2;

int speed = Particle.rand.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.rand.Next(RadiusMin, RadiusMax);

}

}

//Верхний или широкий эмиттер

public class WideEmitter : Emitter

{

//Параметры наследуются от обычного эмиттера

public int Width = 1;//Ширина эмиттера, обычно выставляется ширина pictureBox'а

public override void ResetParticle(Particle particle)

{

base.ResetParticle(particle);

particle.X = Particle.rand.Next(Width);

particle.Y = 0;

particle.SpeedY = 1;

particle.SpeedX = Particle.rand.Next(-2, 2);

}

}

}

# 4 Код специальных точек

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace Coursework

{

//Оновной класс - точка взаимодействия

public abstract class ImpactPoint

{

public float X;

public float Y;

public abstract void ImpactParticle(Particle particle);

public virtual void Render(Graphics g)

{

g.FillEllipse(new SolidBrush(Color.Red), X - 5, Y - 5, 10, 10);

}

}

//============================================================================

//Гравитон и антиГравитон, я их не использую, но оставил, на всякий случай

public class GravityPoint : ImpactPoint

{

public float Power = 100;

public override void ImpactParticle(Particle particle)

{

float gX = X - particle.X;

float gY = Y - particle.Y;

float r2 = (float)Math.Max(100, gX \* gX + gY \* gY);

particle.SpeedX += gX \* Power / r2;

particle.SpeedY += gY \* Power / r2;

}

}

public class AntiGravityPoint : ImpactPoint

{

public float Power = 100;

public override void ImpactParticle(Particle particle)

{

float gX = X - particle.X;

float gY = Y - particle.Y;

float r2 = (float)Math.Max(100, gX \* gX + gY \* gY);

particle.SpeedX -= gX \* Power / r2;

particle.SpeedY -= gY \* Power / r2;

}

public override void Render(Graphics g)

{

g.FillEllipse(new SolidBrush(Color.Blue), X - 5, Y - 5, 10, 10);

}

}

//==============================================================================

//Раскрашивающий частицы круг

public class PainterPoint : ImpactPoint

{

public Color PointColor = Color.White;

public float Rad = 15;

public override void ImpactParticle(Particle particle)

{

//Проверка попадания частицы в радиус круга

float gX = X - particle.X;

float gY = Y - particle.Y;

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

if (r - particle.Radius < Rad)

{

particle.Color1 = PointColor;

particle.Color0 = Color.FromArgb(0, PointColor);

}

}

public override void Render(Graphics g)

{

g.DrawEllipse(new Pen(new SolidBrush(PointColor), 2), X - Rad, Y - Rad, Rad \* 2, Rad \* 2);

}

}

//Круг счётчик

public class CounterPoint : ImpactPoint

{

public int Count = 0;

public float Rad = 15;

public override void ImpactParticle(Particle particle)

{

//Проверка попадания частицы в радиус круга

float gX = X - particle.X;

float gY = Y - particle.Y;

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

if (r - particle.Radius < Rad)

{

particle.Life = 0;

Count++;

}

}

public override void Render(Graphics g)

{

g.FillEllipse(new SolidBrush(Color.FromArgb(100, Color.Red)), X - Rad, Y - Rad, Rad \* 2, Rad \* 2);

g.DrawEllipse(new Pen(new SolidBrush(Color.White), 2), X - Rad, Y - Rad, Rad \* 2, Rad \* 2);

var stringFormat = new StringFormat();

stringFormat.Alignment = StringAlignment.Center;

stringFormat.LineAlignment = StringAlignment.Center;

g.DrawString(

$"{Count}",

new Font("Verdana", 10),

new SolidBrush(Color.White),

X,

Y,

stringFormat

);

}

}

}

# 5 Код частиц

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace Coursework

{

//Я избавился от класса ColorfulParticle ради удобства работы с перекрашивающим кругом

public class Particle

{

//Базовые характеристики частицы

public float X;

public float Y;

public int Radius;

public int Life;

public float SpeedX;

public float SpeedY;

public Color Color0;

public Color Color1;

public static Random rand = new Random();

public Particle()

{

float speed = 1 + rand.Next(10);

float direction = rand.Next(360);

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

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

Radius = 1 + rand.Next(10);

Life = 20 + rand.Next(100);

}

//Расчёт промежуточных цветов частицы

public static Color MixColor(Color color1, Color color0, float k)

{

return Color.FromArgb(

(int)(color0.A \* k + color1.A \* (1 - k)),

(int)(color0.R \* k + color1.R \* (1 - k)),

(int)(color0.G \* k + color1.G \* (1 - k)),

(int)(color0.B \* k + color1.B \* (1 - k))

);

}

public virtual void Draw(Graphics g)

{

float k = Math.Min(1f, Life / 100f); //Коэффицент прозрачности частицы

var color = MixColor(Color0, Color1, k);

var b = new SolidBrush(color);

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

b.Dispose();

}

}

}

# 6 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;

namespace Coursework

{

public partial class Form1 : Form

{

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

public Emitter emitter;

public PainterPoint paint1;

public PainterPoint paint2;

public Form1()

{

InitializeComponent();

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

//Сам эмиттер, я использую только его широкую версию

emitter = new WideEmitter

{

Width = picDisplay.Width,

GravityY = 0.2f,

ParticlesPerTick = 2,

};

//Добавляю эмиттер в список эмиттеров, для того что бы он обновлялся

emitters.Add(emitter);

//Два раскрашивающих круга

paint1 = new PainterPoint

{

PointColor = Color.White,

X = (picDisplay.Width / 2) - 200,

Y = (picDisplay.Height / 2) - 100,

Rad = 50,

};

paint2 = new PainterPoint

{

PointColor = Color.Red,

X = (picDisplay.Width / 2) + 200,

Y = (picDisplay.Height / 2) - 100,

Rad = 50,

};

//Добавляю раск. круги в список impactpoint'ов

emitter.impactPoints.Add(paint1);

emitter.impactPoints.Add(paint2);

ElementsStartPos();

}

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

{

foreach(var emitter in emitters)

{

//Обновление состояния всех элементов эмиттера

emitter.UpdateState();

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

{

g.Clear(Color.Black);

//Рендер всех элементов эмиттера

emitter.Render(g);

}

}

picDisplay.Invalidate();

ElementsUpdate();

}

//Задание начальных значений формы

public void ElementsStartPos()

{

XBar1.Maximum = picDisplay.Width;

YBar1.Maximum = picDisplay.Height;

XBar2.Maximum = picDisplay.Width;

YBar2.Maximum = picDisplay.Height;

XBar1.Value = (int)paint1.X;

YBar1.Value = (int)paint1.Y;

XBar2.Value = (int)paint2.X;

YBar2.Value = (int)paint2.Y;

RadBar1.Value = (int)paint1.Rad;

RadBar2.Value = (int)paint2.Rad;

PPTBar.Value = emitter.ParticlesPerTick;

LTBar.Value = emitter.LifeMax;

MinLTBar.Value = emitter.LifeMin;

GBar.Value = (int)(emitter.GravityY \* 10);

CRadBar.Value = 50;

FirstColorPick.BackColor = emitter.ColorFrom;

SecondColorPick.BackColor = Color.FromArgb(255, emitter.ColorTo);

ColorPick1.BackColor = paint1.PointColor;

ColorPick2.BackColor = paint2.PointColor;

}

//Обновление значений элементов(Лейблы и т.п.)

public void ElementsUpdate()

{

CounterLabel.Text = emitter.particles.Count(particle => particle.Life > 0).ToString();

PPTLabel.Text = PPTBar.Value.ToString();

LTLabel.Text = LTBar.Value.ToString();

MinLTLabel.Text = MinLTBar.Value.ToString();

GLabel.Text = (GBar.Value / 10f).ToString();

XLabel1.Text = XBar1.Value.ToString();

YLabel1.Text = YBar1.Value.ToString();

RadLabel1.Text = RadBar1.Value.ToString();

XLabel2.Text = XBar2.Value.ToString();

YLabel2.Text = YBar2.Value.ToString();

RadLabel2.Text = RadBar2.Value.ToString();

CRadLabel.Text = CRadBar.Value.ToString();

}

//=============================================================

//Много методов-событий от элементов формы.

private void XBar1\_Scroll(object sender, EventArgs e)

{

paint1.X = XBar1.Value;

}

private void YBar1\_Scroll(object sender, EventArgs e)

{

paint1.Y = YBar1.Value;

}

private void RadBar1\_Scroll(object sender, EventArgs e)

{

paint1.Rad = RadBar1.Value;

}

private void XBar2\_Scroll(object sender, EventArgs e)

{

paint2.X = XBar2.Value;

}

private void YBar2\_Scroll(object sender, EventArgs e)

{

paint2.Y = YBar2.Value;

}

private void RadBar2\_Scroll(object sender, EventArgs e)

{

paint2.Rad = RadBar2.Value;

}

private void ColorPick1\_Click(object sender, EventArgs e)

{

colorDialog.ShowDialog();

paint1.PointColor = colorDialog.Color;

ColorPick1.BackColor = paint1.PointColor;

}

private void ColorPick2\_Click(object sender, EventArgs e)

{

colorDialog.ShowDialog();

paint2.PointColor = colorDialog.Color;

ColorPick2.BackColor = paint2.PointColor;

}

private void PPTBar\_Scroll(object sender, EventArgs e)

{

emitter.ParticlesPerTick = PPTBar.Value;

}

private void GBar\_Scroll(object sender, EventArgs e)

{

emitter.GravityY = (GBar.Value / 10f);

}

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

{

colorDialog.ShowDialog();

emitter.ColorFrom = colorDialog.Color;

FirstColorPick.BackColor = emitter.ColorFrom;

}

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

{

colorDialog.ShowDialog();

emitter.ColorTo = Color.FromArgb(0, colorDialog.Color);

SecondColorPick.BackColor = Color.FromArgb(255, emitter.ColorTo);

}

//Два метода - события для ползунков, определяющих время жизни частиц.

//Они взаимозависимы, поэтому изменяют максимальное и минимальное значения друг-друга при каждом вызове

private void LTBar\_Scroll(object sender, EventArgs e)

{

emitter.LifeMax = LTBar.Value;

MinLTBar.Maximum = LTBar.Value;

LTBar.Minimum = MinLTBar.Value;

}

private void MinLTBar\_Scroll(object sender, EventArgs e)

{

emitter.LifeMin = MinLTBar.Value;

MinLTBar.Maximum = LTBar.Value;

LTBar.Minimum = MinLTBar.Value;

}

//Создание и удаление кругов-счётчиков

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

{

if(e.Button == MouseButtons.Left)

{

emitter.impactPoints.Add(new CounterPoint

{

X = e.X,

Y = e.Y,

Rad = CRadBar.Value,

});

}

else if (e.Button == MouseButtons.Right)

{

foreach(var point in emitter.impactPoints.ToArray())

{

//Если impactpoint - счётчик

if(point is CounterPoint)

{

//то проверяем на пересечение с местом клика мышью

CounterPoint cpoint = point as CounterPoint;

float gX = cpoint.X - e.X;

float gY = cpoint.Y - e.Y;

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

if(r <= cpoint.Rad)

{

//и при пересечении удаляем счётчик.

emitter.impactPoints.Remove(point);

}

}

}

}

}

//=================================================================

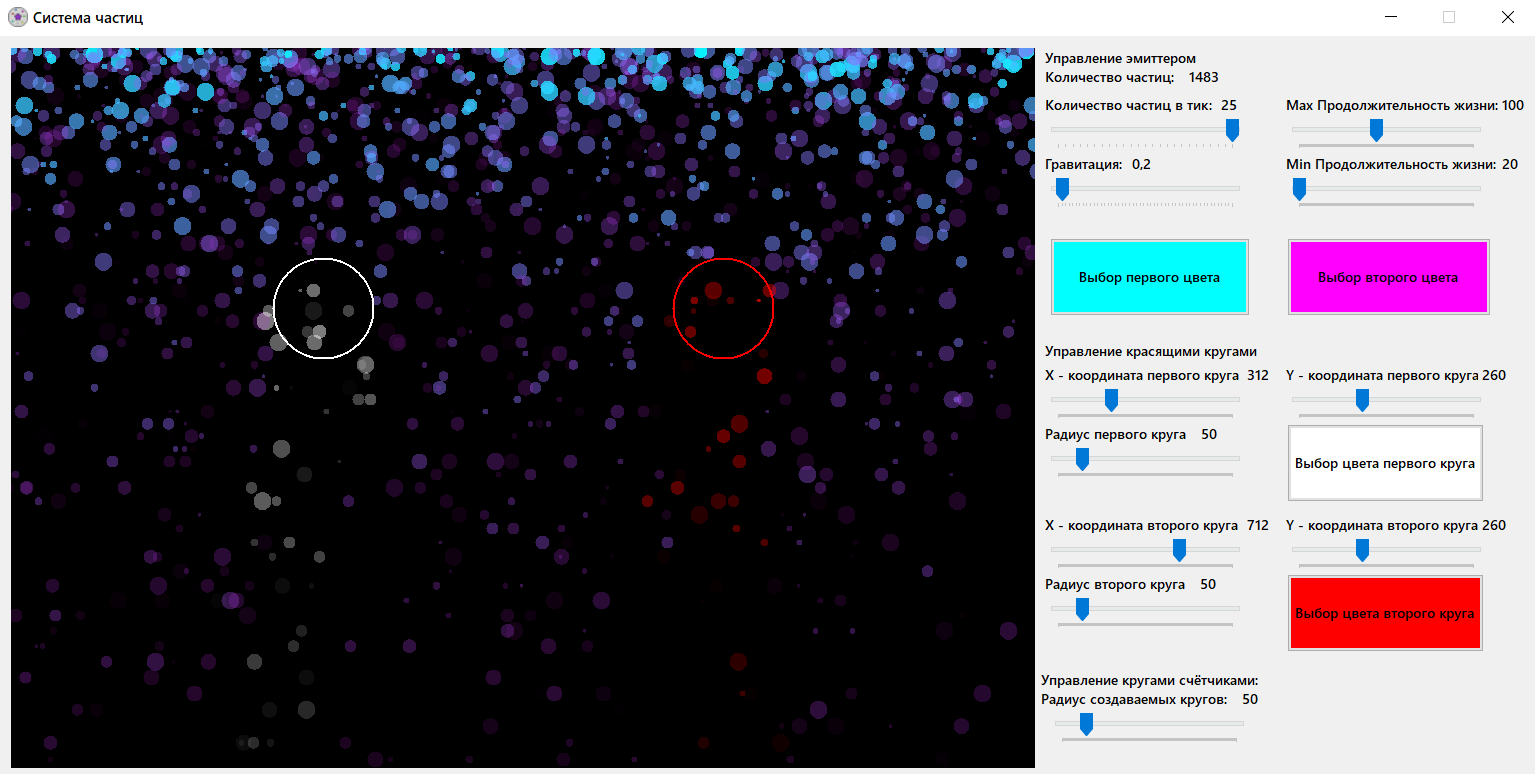
}

}

# 7 Описание работы интерфейса

**Управление эмиттером:**

Меняя trackback **Количество частиц в тик** мы меняем количество частиц, появляющихся за один тик (40мс):

  
Рисунок 2.1

Меняя trackback **Гравитация** мы меняем ускорение действующее на частицы по оси Y:

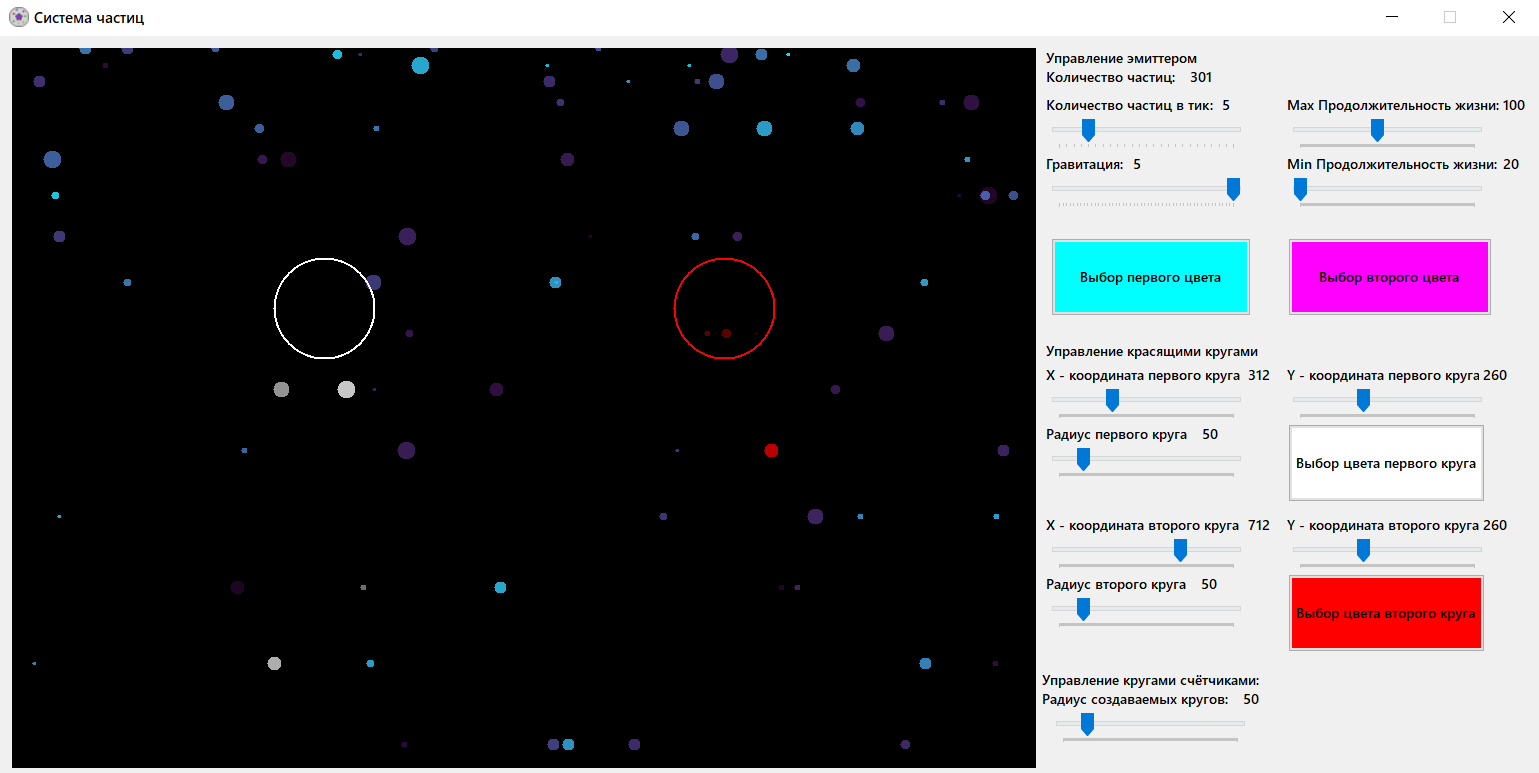


Рисунок 2.2

Меняя trackbar **Max Продолжительность жизни,** мы меняем максимальный запас жизни у появляющихся частиц:

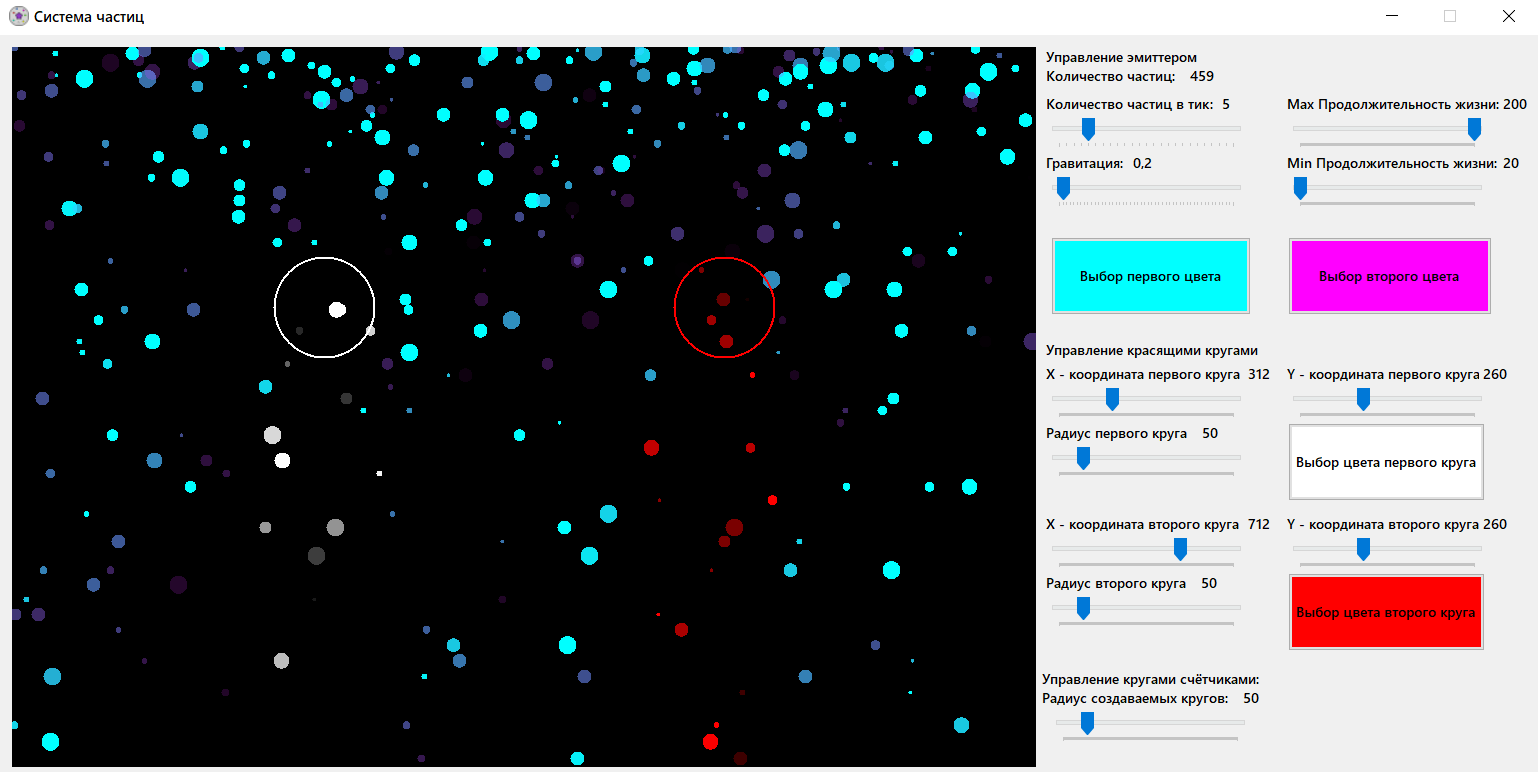


Рисунок 2.3

Меняя trackbar **Min Продолжительность жизни**, мы меняем минимальный запас жизни у появляющихся частиц:

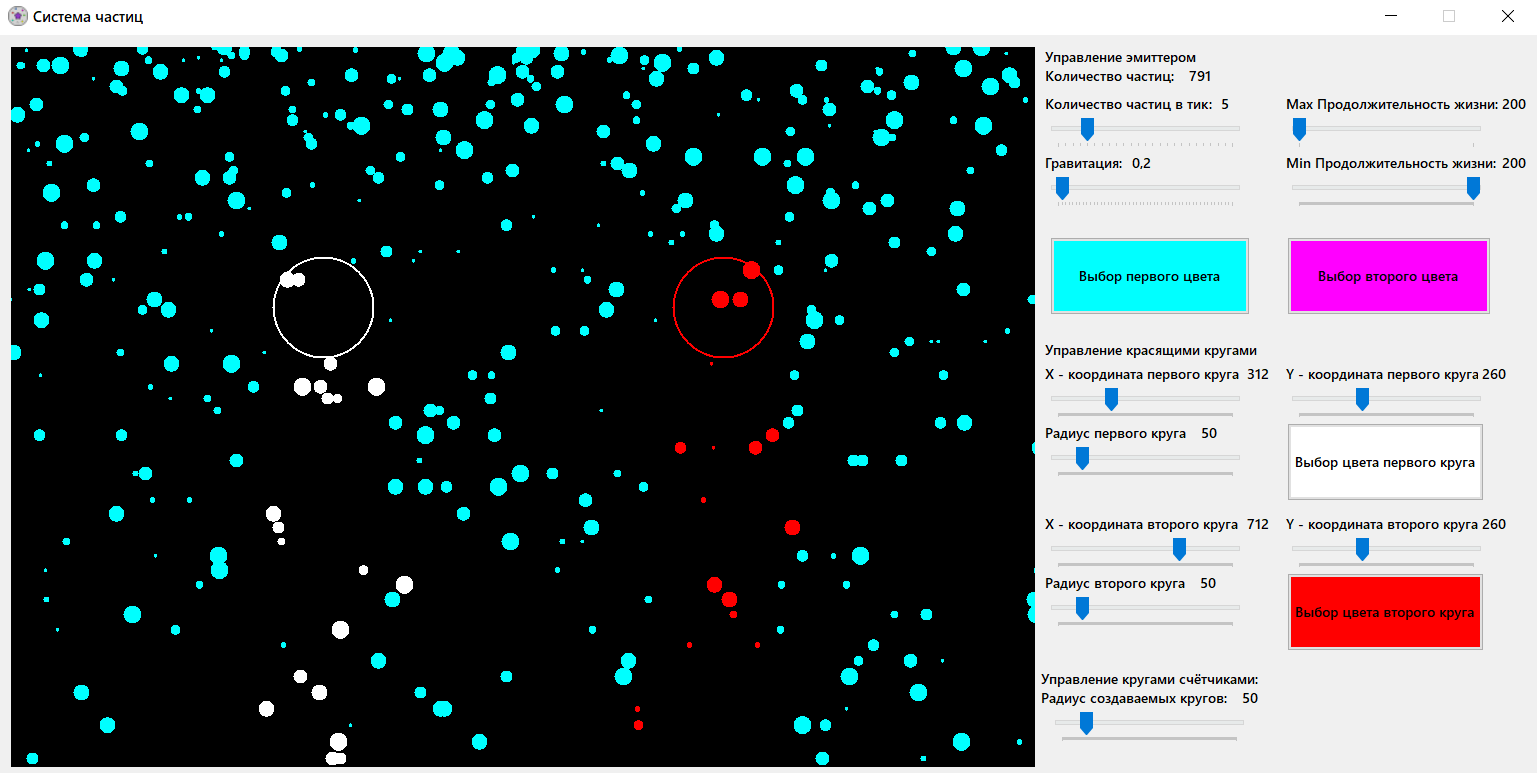


Рисунок 2.4

Нажимая на кнопку **Выбор первого цвета**, откроется палитра, в которой можно выбрать начальный цвет частиц:

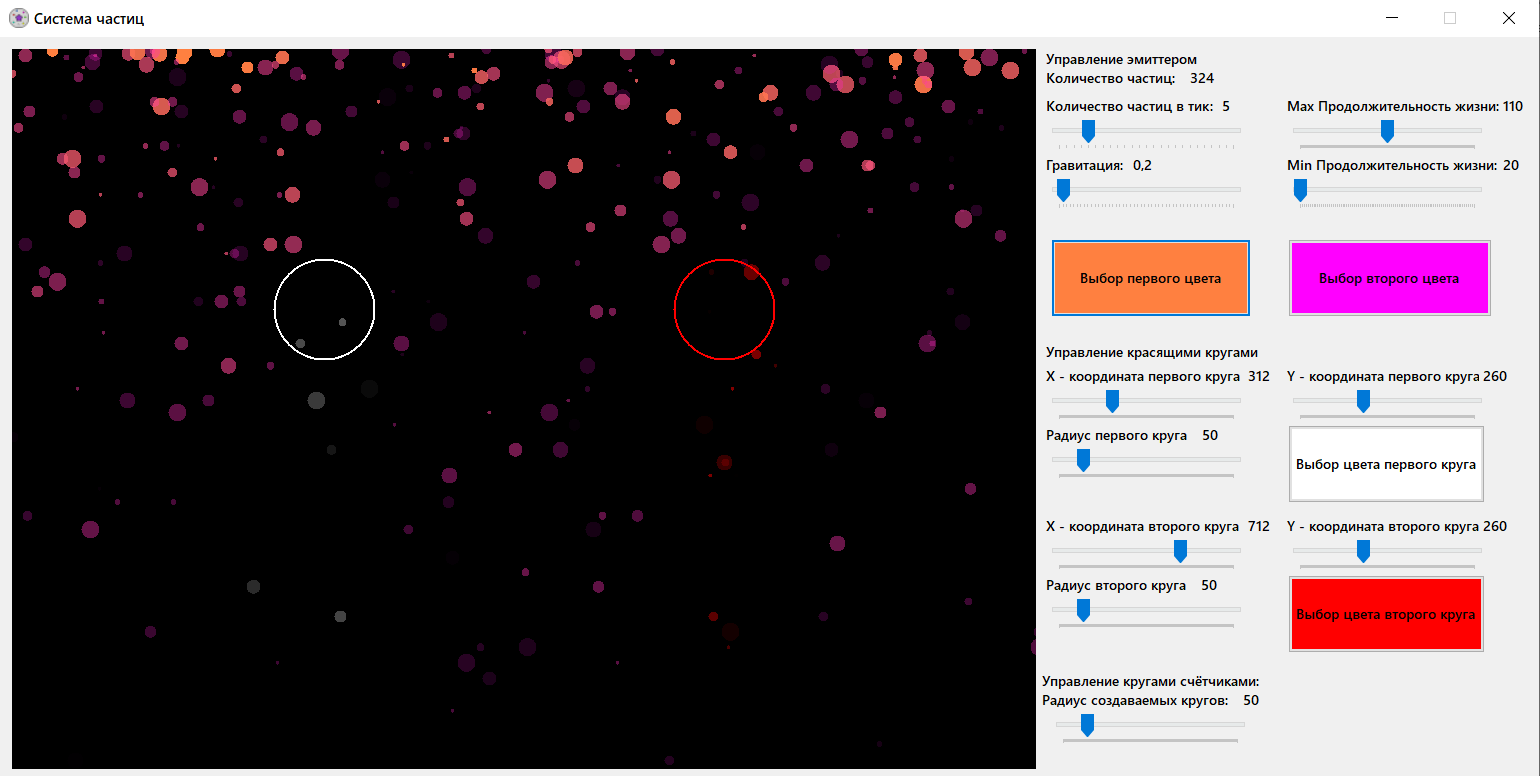


Рисунок 2.5

Нажимая на кнопку **Выбор второго цвета**, откроется палитра, в которой можно выбрать конечный цвет частиц:

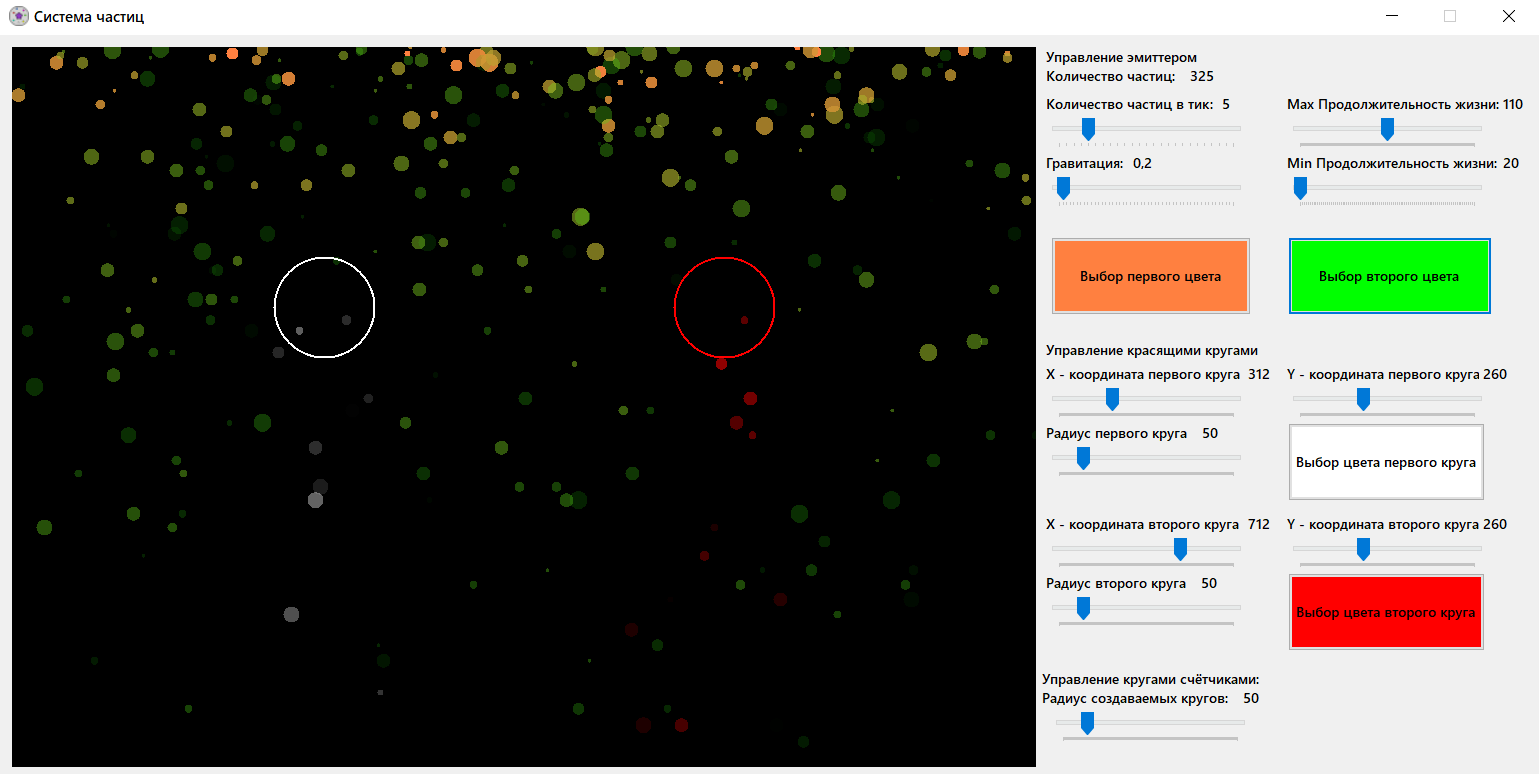


Рисунок 2.6

**Управление красящими кругами:**

Меняя trackbar **X – координата первого круга,** мы меняем положение первого круга по оси X:

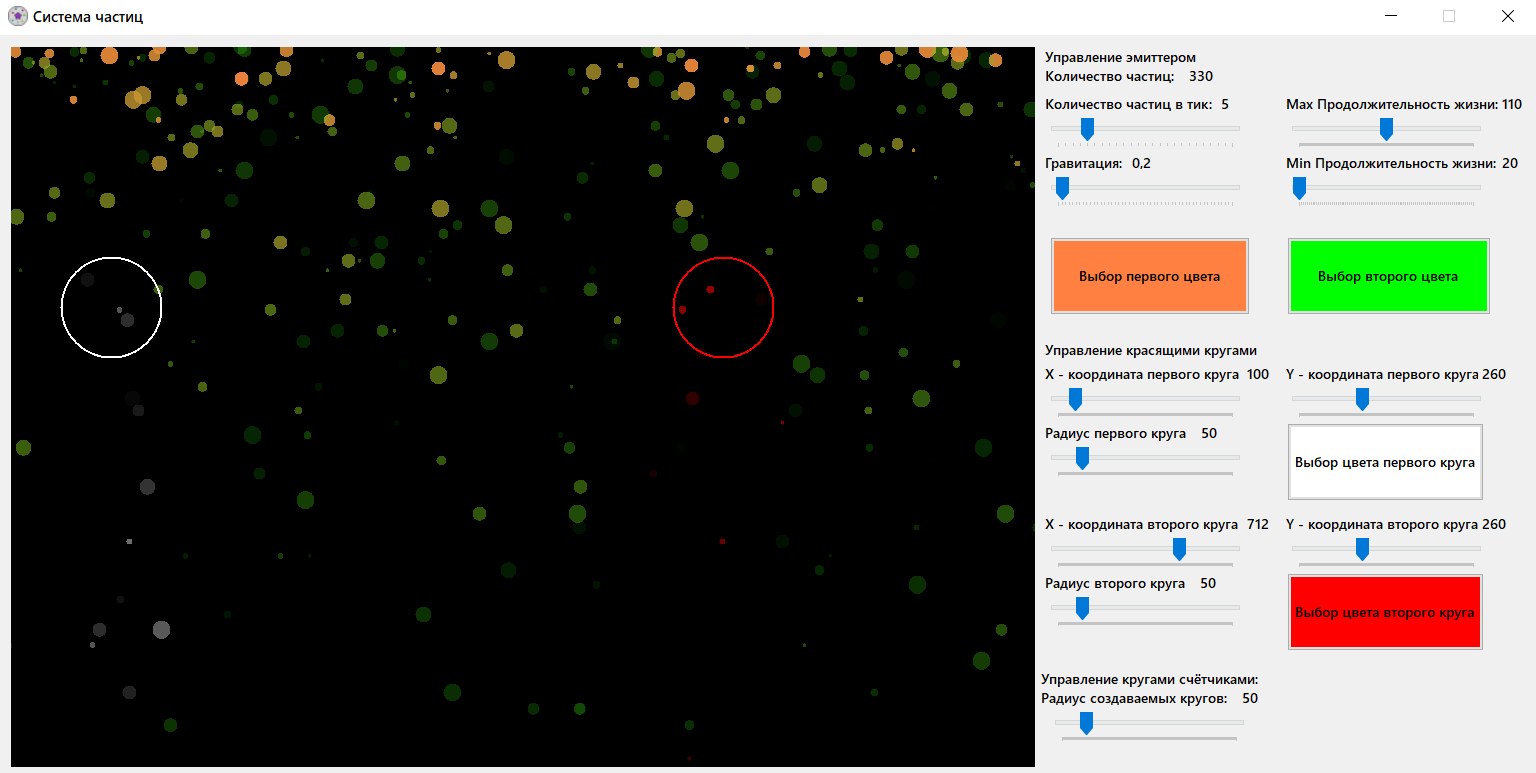


Рисунок 3.1

Меняя trackbar **Y – координата первого круга,** мы меняем положение первого круга по оси Y:

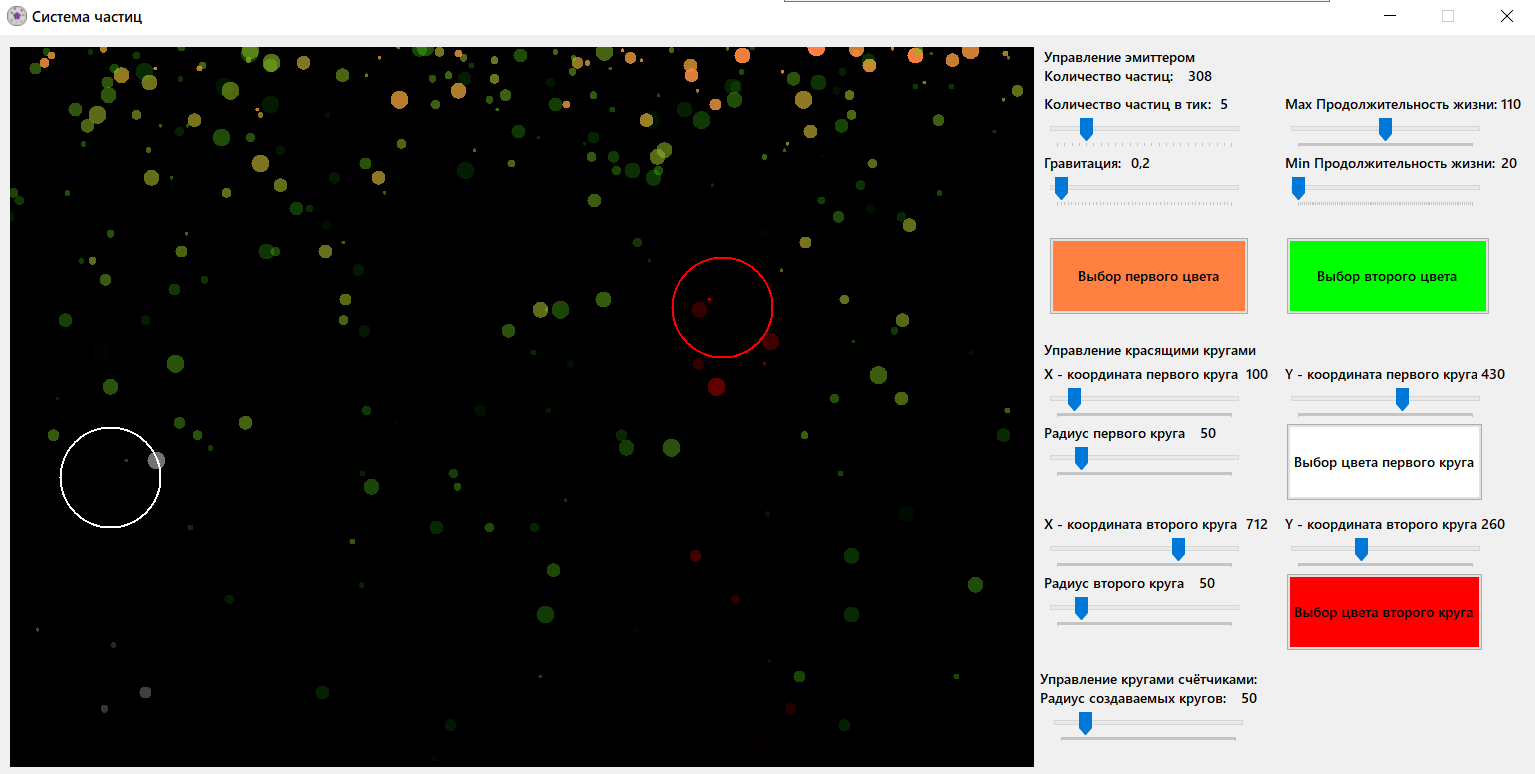
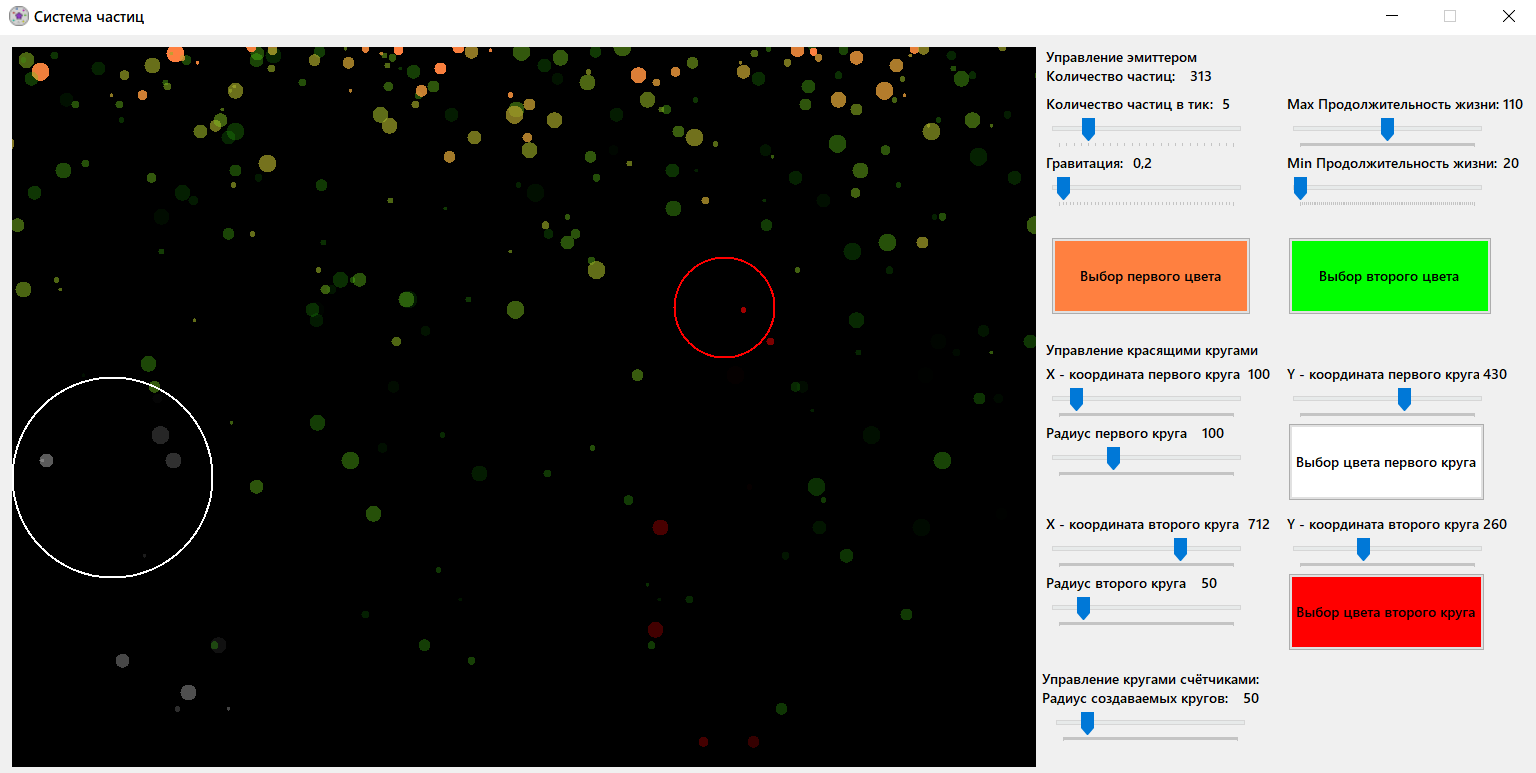


Рисунок 3.2

Меняя trackbar **Радиус первого круга,** мы меняем размер первого круга:

 Рисунок 3.3

Нажимая на кнопку **Выбор цвета первого круга**, откроется палитра, в которой можно выбрать цвет, в который круг будет перекрашивать частицы:

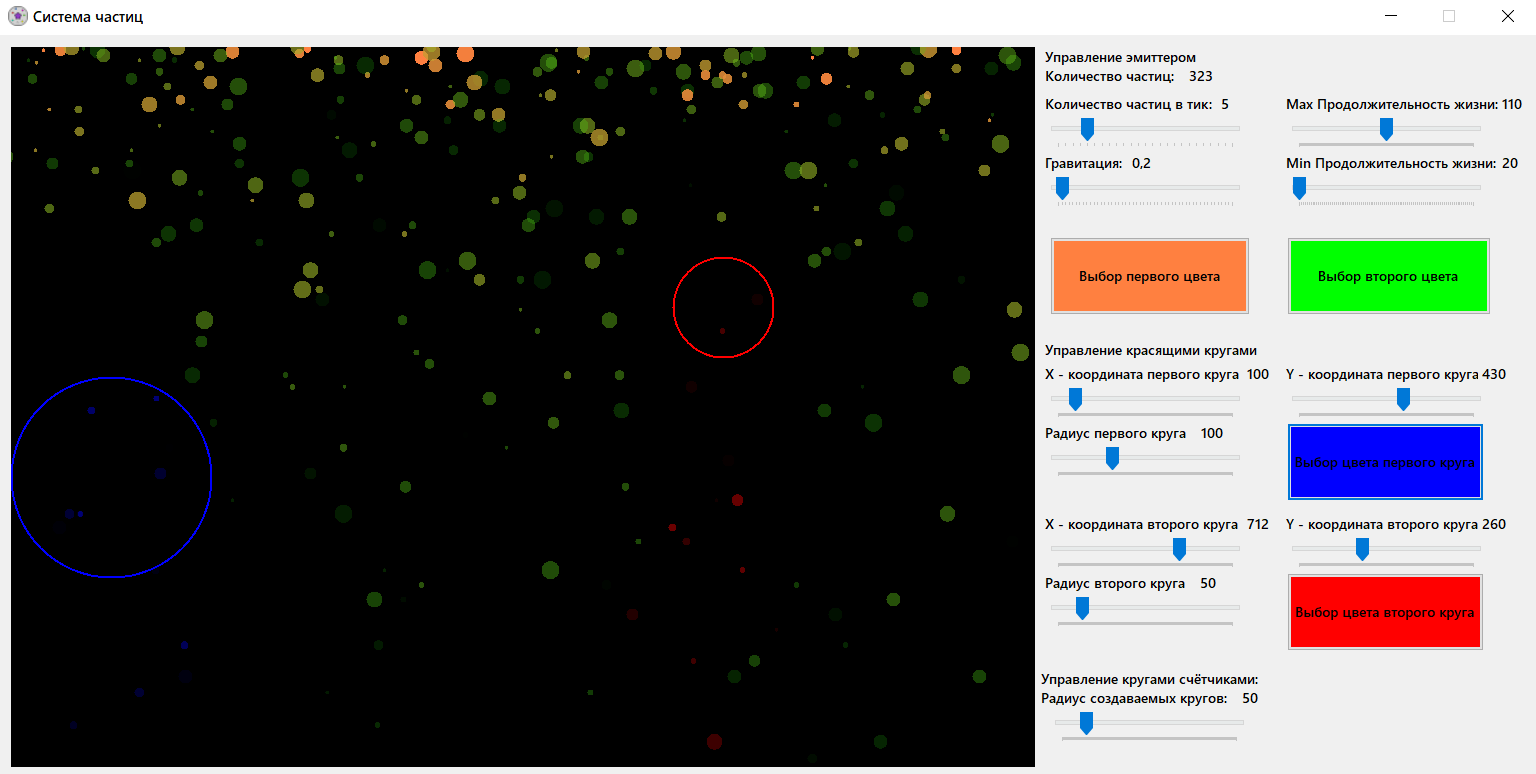


Рисунок 3.4

Меняя trackbar **X – координата второго круга,** мы меняем положение второго круга по оси X:

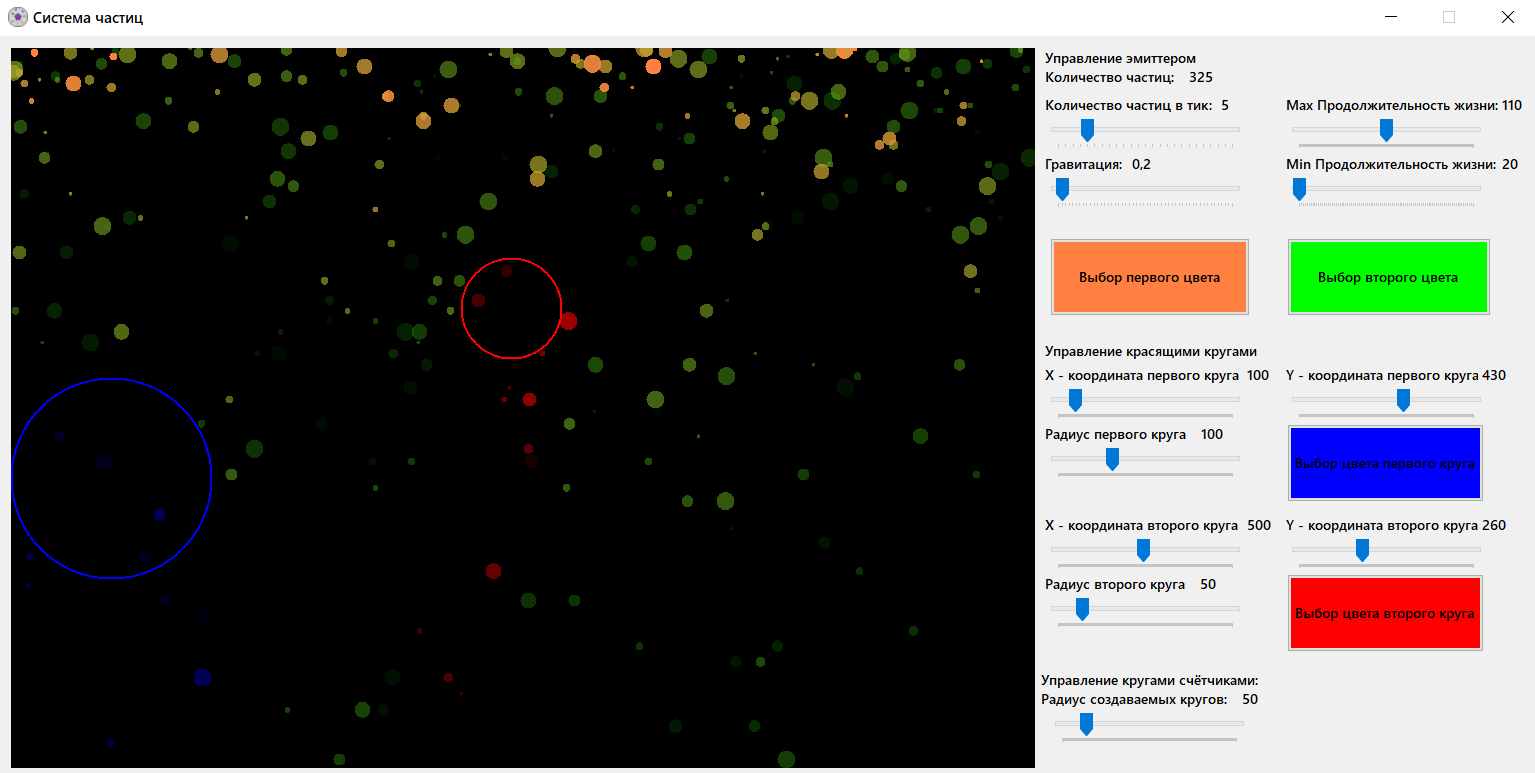


Рисунок 3.5

Меняя trackbar **Y – координата второго круга,** мы меняем положение второго круга по оси Y:

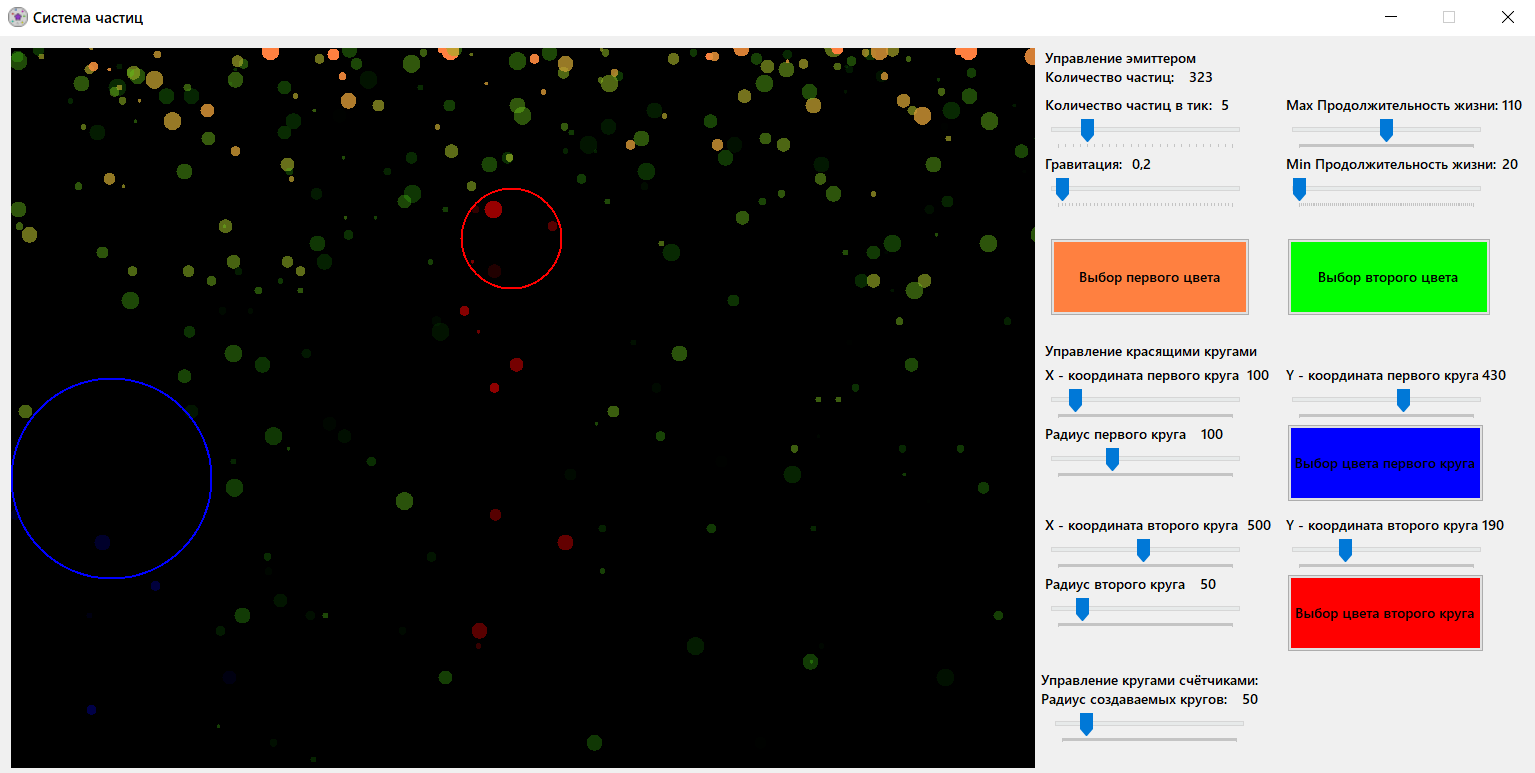


Рисунок 3.6

Меняя trackbar **Радиус второго круга,** мы меняем размер второго круга:

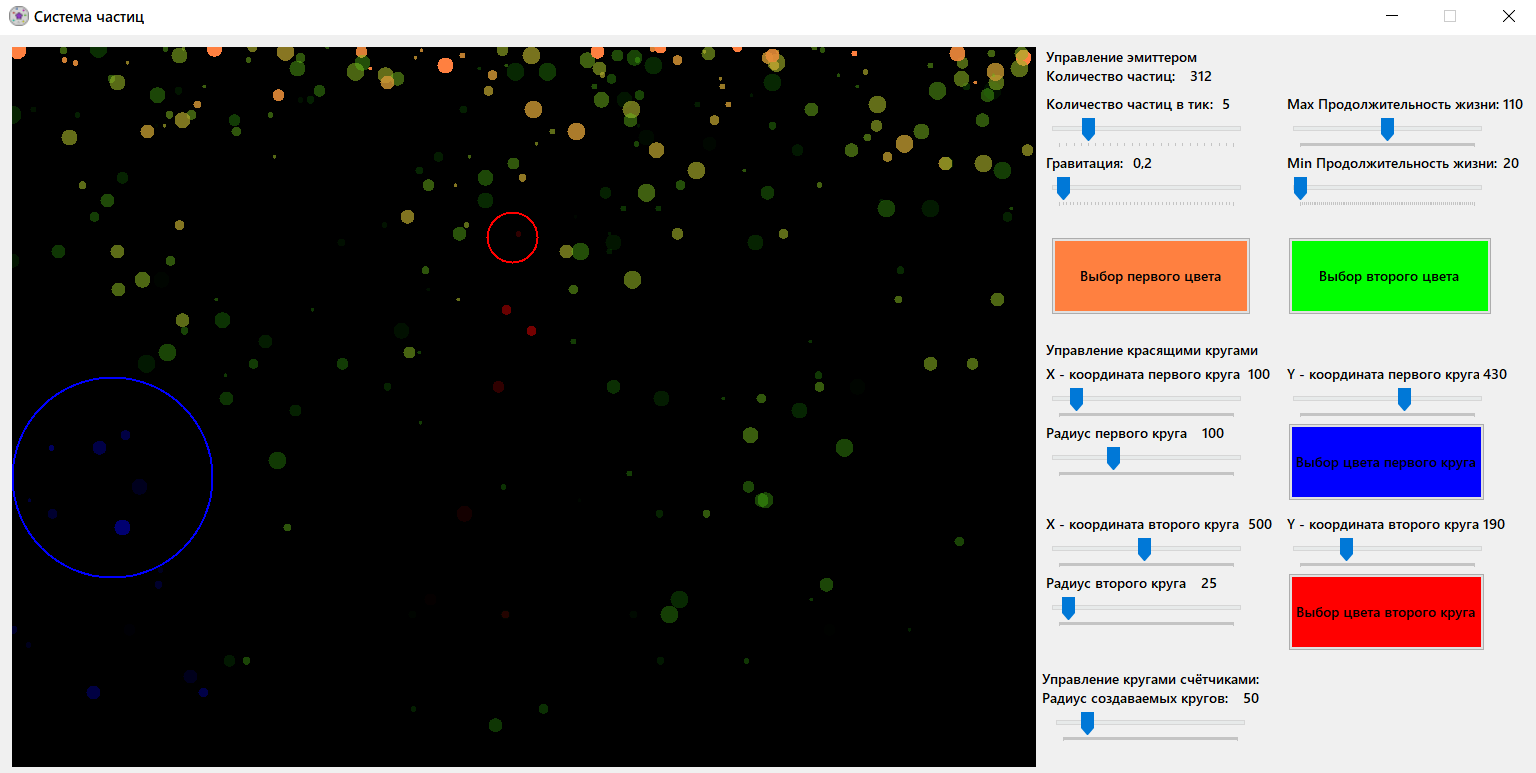


Рисунок 3.7

Нажимая на кнопку **Выбор цвета второго круга**, откроется палитра, в которой можно выбрать цвет, в который круг будет перекрашивать частицы:

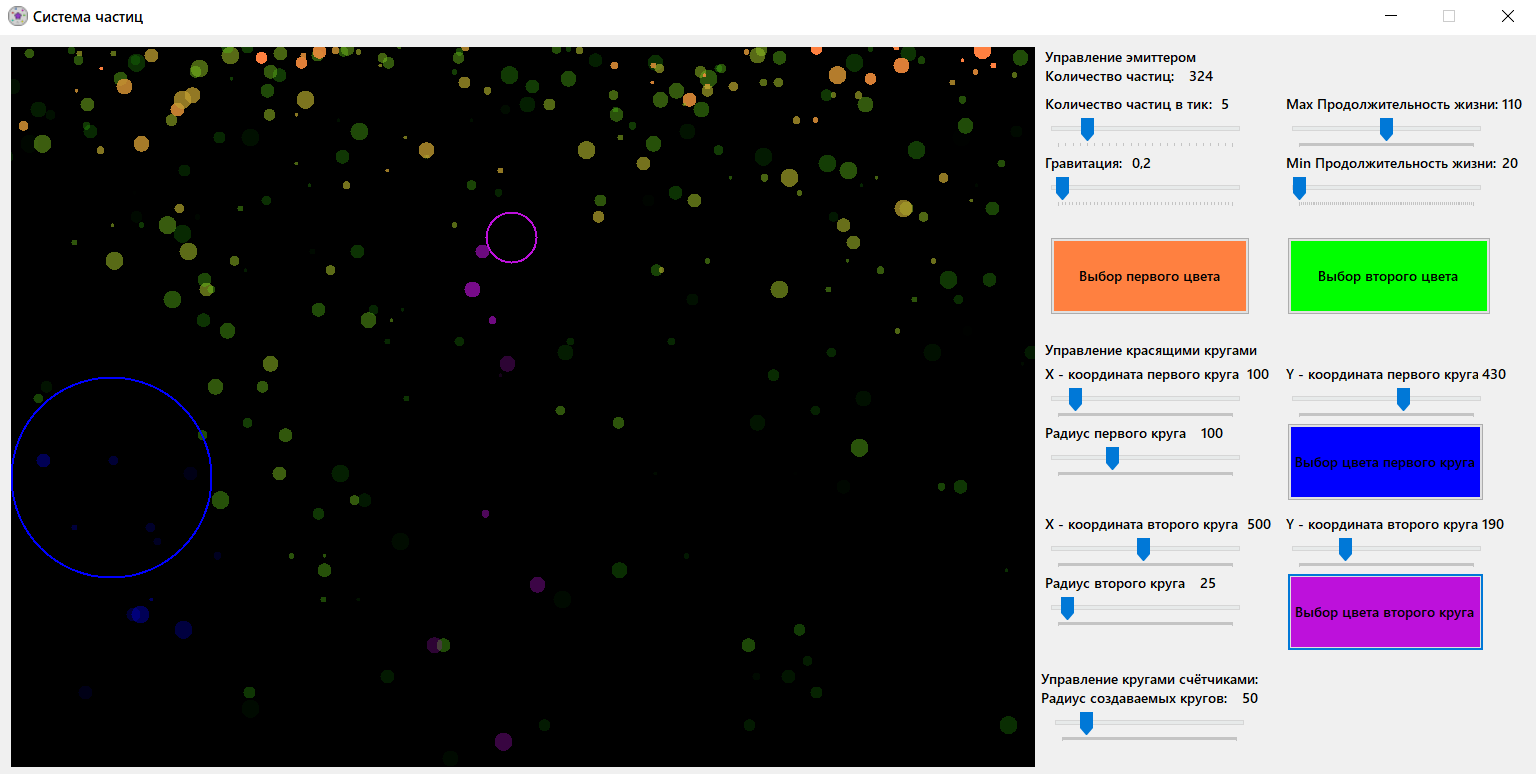


Рисунок 3.8

**Управление кругами счётчиками:**

Кликая левой кнопкой мыши по pictureBox мы создаём в месте клика круг – счётчик:

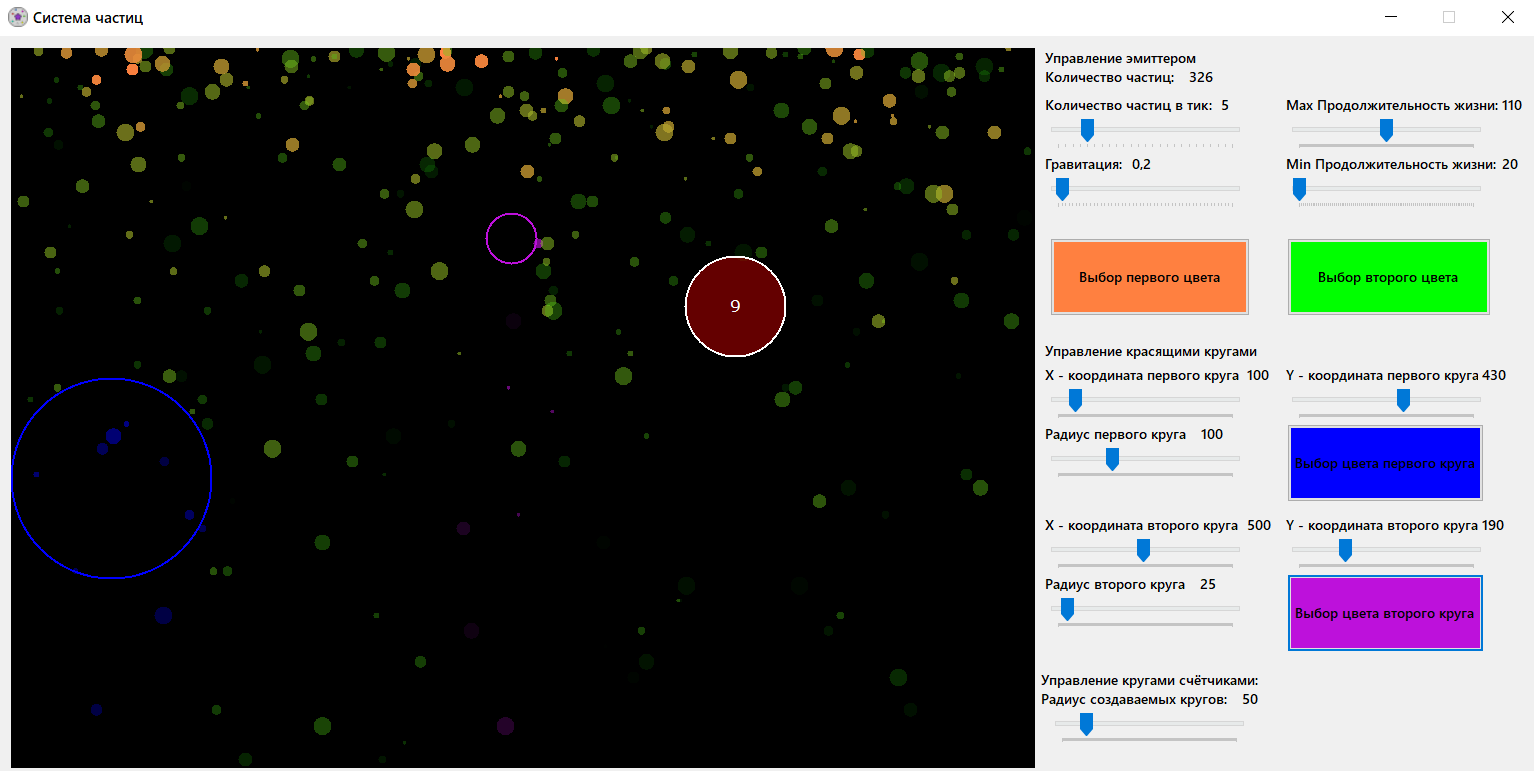


Рисунок 4.1

Кликая правой кнопкой мыши по существующему кругу – счётчику, мы удаляем этот круг:

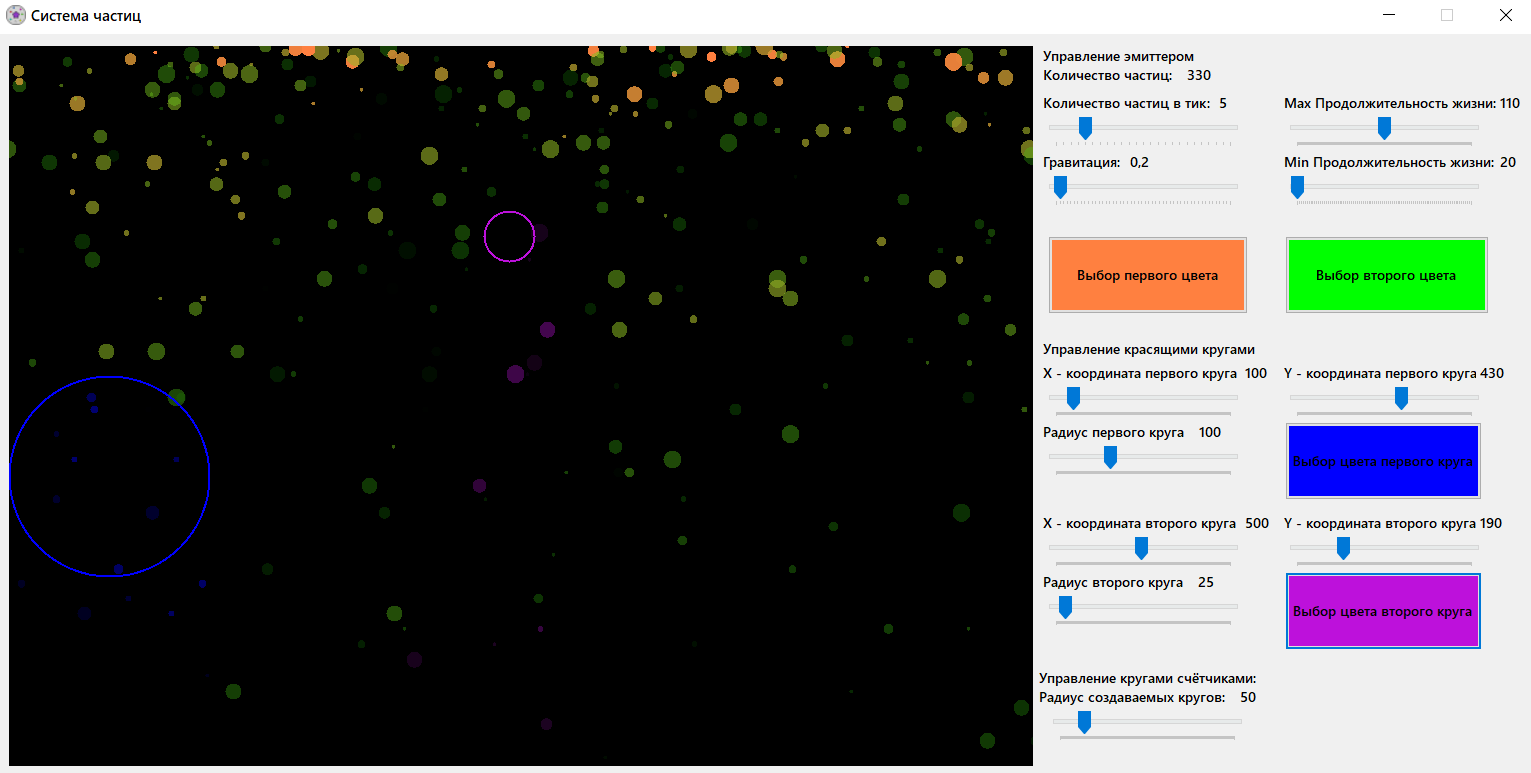


Рисунок 4.2

Меняя trackbar **Радиус создаваемых кругов,** мы меняем размер создаваемых кругов – счётчиков:

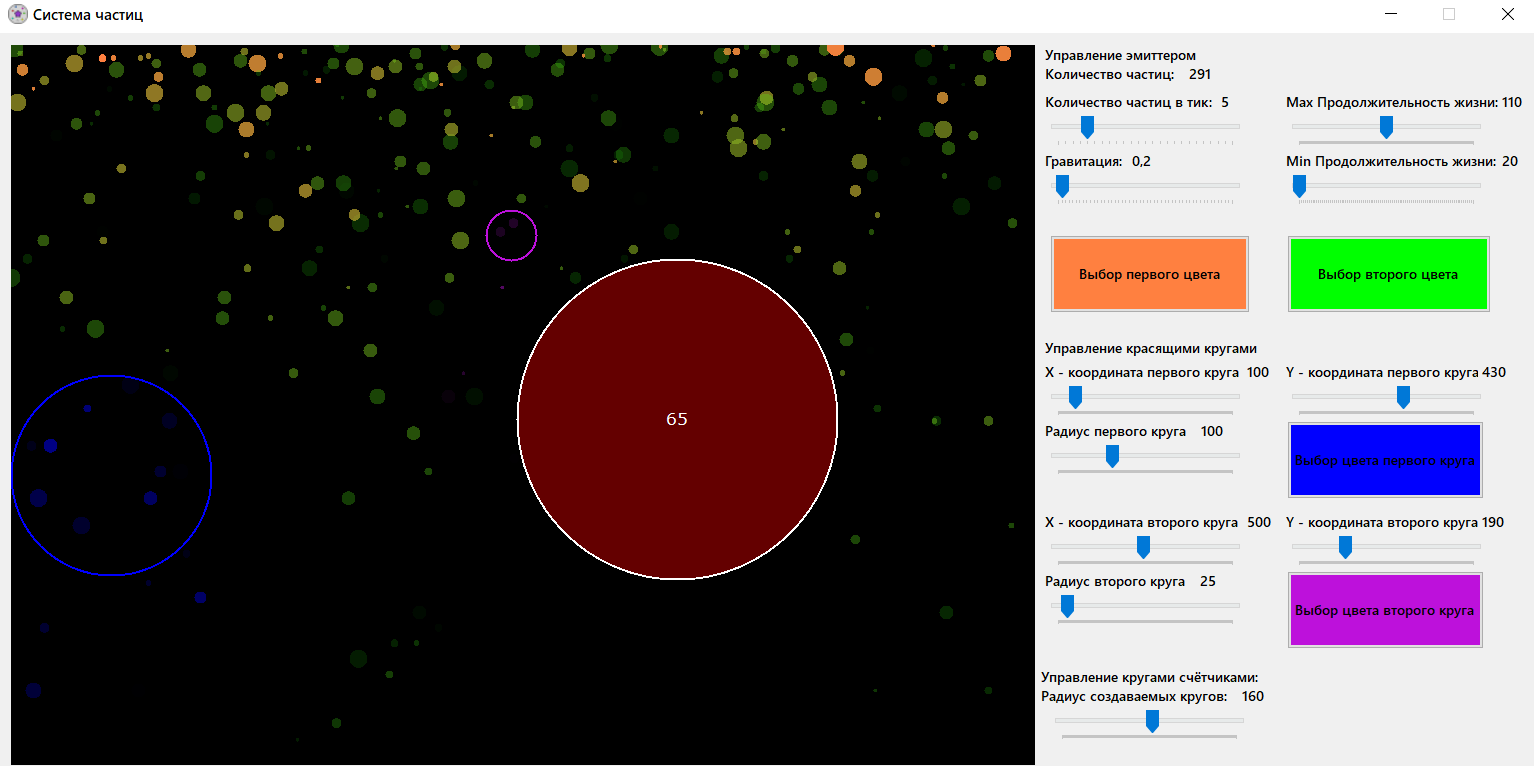


Рисунок 4.3

# Заключение

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

* Ползунок «Количество частиц в тик», определяющий количество частиц, появляющихся в один тик.
* Ползунок «Гравитация», определяющий ускорение, действующее на частицы по оси Y.
* Ползунок «Max продолжительность жизни», определяющий максимальный запас жизни у частиц.
* Ползунок «Min продолжительность жизни», определяющий минимальный запас жизни у частиц.
* Кнопка «Выбор первого цвета», позволяющая сменить начальный цвет частиц.
* Кнопка «Выбор второго цвета», позволяющая сменить конечный цвет частиц.
* Ползунок «X – координата первого круга», определяющий положение первого красящего круга на оси X.
* Ползунок «Y – координата первого круга», определяющий положение первого красящего круга на оси Y.
* Ползунок «Радиус первого круга», определяющий размер первого красящего круга.
* Кнопка «Выбор цвета первого круга», позволяющая сменить цвет, в который первый круг перекрашивает частицы.
* Ползунок «X – координата второго круга», определяющий положение второго красящего круга на оси X.
* Ползунок «Y – координата второго круга», определяющий положение второго красящего круга на оси Y.
* Ползунок «Радиус второго круга», определяющий размер второго красящего круга.
* Кнопка «Выбор цвета второго круга», позволяющая сменить цвет, в который второй круг перекрашивает частицы.
* Ползунок «Радиус создаваемых кругов», определяющий размер создаваемых кругов – счётчиков.
* Объект «Раскрашивающий круг», который раскрашивает пересекающиеся с ним частицы в свой цвет.
* Объект «Круг счётчик», который считает и удаляет пересекающиеся с ним частицы.

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: 10.12.2021).