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:  А.Б. Столбов |
| И.О. Фамилия |

Технология моделирования системы частиц

наименование темы

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине

|  |
| --- |
| Технологии программирования |

|  |
| --- |
| 1.011.00.00 ПЗ |
| обозначение документа |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | ИСТб-19-2 |  |  |  | Комогорцева Ю.В. |
|  | шифр группы |  | подпись |  | Фамилия И.О |
| Нормоконтроль |  |  |  |  | Столбов А.Б. |
|  |  |  | подпись |  | Фамилия И.О |

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

Иркутск 2020 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 | Комогорцевой Юлии Владимировне | |
|  | (фамилия, инициалы) | |
| Тема работы: | Технология моделирования системы частиц | | |
| Исходные данные: | | Вариант 9 | |
| 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] // olive.tealeaf.su : чаинка, 2020. URL: <http://olive.tealeaf.su/particle-system.html> (дата обращения: 01.12.2020). | | | |
|  | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Графическая часть на | - | | листах. | |
| Дата выдачи задания |  | 23 / 11 / 2020 г. | | |
| Задание получил студент | | | |  | |  | Комогорцева Ю.В. |
|  | | | | подпись | |  | Фамилия И.О. |

|  |  |
| --- | --- |
| Дата представления работы руководителю | 21 / 12 / 2020 г. |
| Руководитель курсовой работы |  |  | Столбов А.Б. |
|  | 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 14](#_Toc58847162)

[5 Код частиц 17](#_Toc58847163)

[6 Код формы 30](#_Toc58847164)

[7 Описание работы интерфейса 44](#_Toc58847165)

[Заключение 53](#_Toc58847166)

[Список использованной литературы 54](#_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 debug режим , в котором рисуются вектора скорости каждой частицы, добавить кнопку для остановки симуляции движения частиц , добавить кнопку для пошагового выполнения симуляции , добавить trackbar для замедления симуляции ,а так же при наведении мышки на частицу, выводить ее координаты и значение Life.

1. Дополнительно добавить кнопочку которая позволяет выполнять симуляцию в обратную сторону, хотя бы на последние 10 шагов.

# 2 Внешний вид главного окна

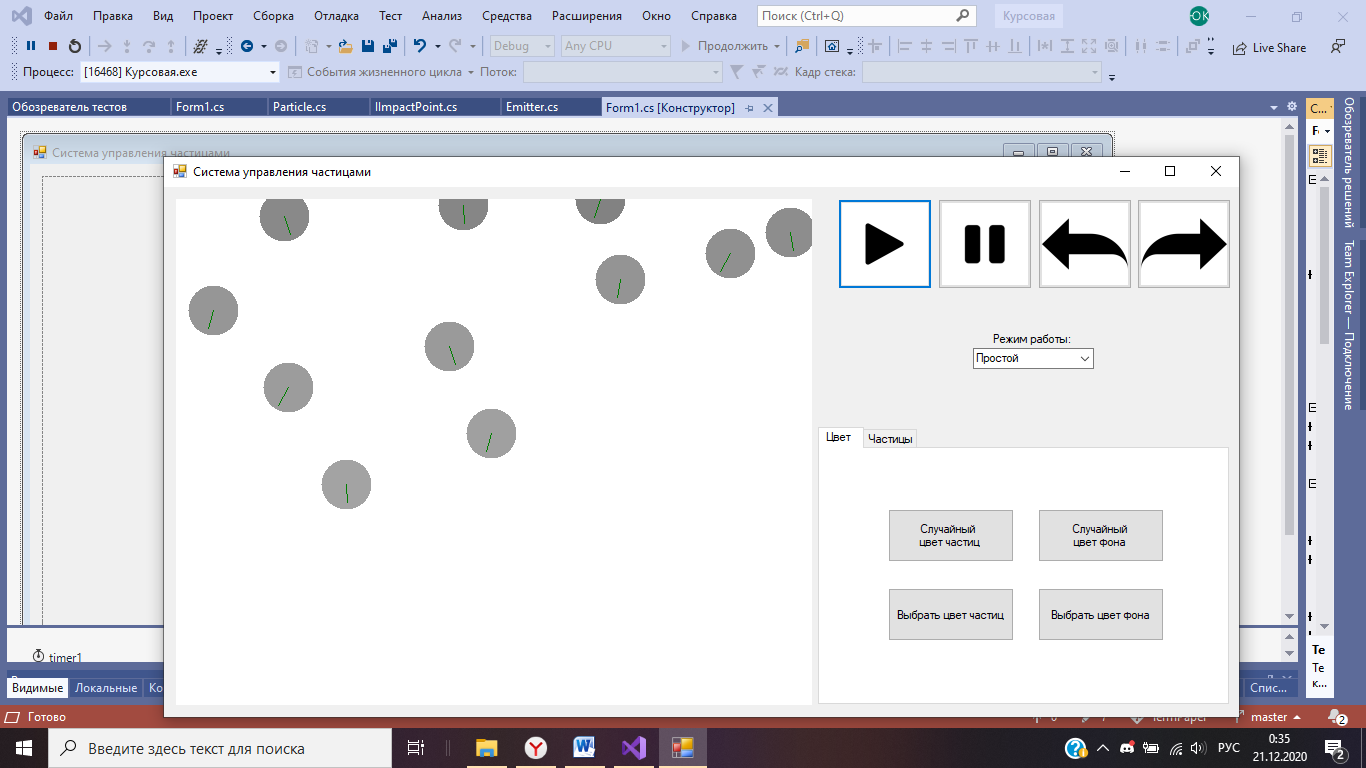


Рисунок 2.1

# 3 Код эмиттера

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing; // чтобы использовать Graphics

namespace Курсовая

{

public class Emitter

{

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

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

public List<List<ParticleColorful>> particlesHistory = new List<List<ParticleColorful>>(40);

public List<ParticleColorful> particlesRemove = new List<ParticleColorful>();

public int currentHistoryIndex = 0;

public bool ifAdd = true; //в первый раз ли достигается последняя граница списка истории

public int gravitationX = 0;

public int gravitationY = 0;

public int particlesCount ;

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

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

public float Speed = 0; // начальная минимальная скорость движения частицы

public int Size ; // минимальный радиус частицы

public int Life ; // минимальное время жизни частицы

public int ParticlesPerTick = 1;

public long tickRate = 30;

public long tickCount = 0;

public int Width; // длина экрана

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

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

public string figure ; //форма частиц

public void UpdateState()

{

if (tickCount % tickRate == 0)

{

int particlesToCreate = ParticlesPerTick;

int i = 0;

foreach (var particle in particles)

{

//влияние доп точек(окрашивателей)

foreach (var point in impactPoints)

{

point.ImpactParticle(particle);

}

particle.Life--;

//если частица умерла

if (particle.Life < 0)

{

//если нужно создавать новые частицы

if (ParticlesPerTick != 0)

{

//то обновляем характеристсики умерших

ResetParticle(particle);

}

//если нет то удаляем упавшую частицу

particlesRemove.Add(particle);

}

else

{

//если частица жива,то пусть падает дальше

particle.SpeedX += gravitationX;

particle.SpeedY += gravitationY;

particle.X += particle.SpeedX;

particle.Y += particle.SpeedY;

}

}

//очищение списка от лишних частиц

foreach (var particle in particlesRemove)

{

particles.Remove(particle);

}

particlesRemove.Clear();

//пока нужно создавать новые частицы

while (particlesToCreate >= 1)

{

particlesToCreate -= 1;

//создаем новые

var particle = CreateParticle();

ResetParticle(particle);

particles.Add(particle);

}

if (currentHistoryIndex < 39)

{

if (currentHistoryIndex >= particlesHistory.Count)

{

particlesHistory.Add(new List<ParticleColorful>());

}

foreach (var particle in particles)

{

ParticleColorful part = (ParticleColorful)particle.Clone();

particlesHistory[currentHistoryIndex].Add(part);

}

currentHistoryIndex++;

ifAdd = true;

}

else

{

if (!ifAdd) particlesHistory.RemoveAt(0);

ifAdd = false;

particlesHistory.Add(new List<ParticleColorful>());

foreach (var particle in particles)

{

ParticleColorful part = (ParticleColorful)particle.Clone();

particlesHistory[currentHistoryIndex].Add(part);

}

}

}

tickCount++;

if (tickCount < 0) tickCount = 0;

}

public virtual ParticleColorful CreateParticle()

{

var particle = new ParticleColorful();

//создаем новую частицу исходя из формы

switch (figure)

{

case "circle":

{

particle = new ParticleCircle();

break;

}

case "square":

{

particle = new ParticleSquare();

break;

}

case "star":

{

particle = new ParticleStar();

break;

}

case "snowflake":

{

particle = new ParticleSnowflake();

break;

}

}

particle.FromColor = ColorFrom;

particle.ToColor = ColorTo;

return particle;

}

// заполнение характеристик частицы

public virtual void ResetParticle(Particle particle)

{

particle.Life = Life;

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

particle.SpeedX = (int)(Math.Cos(direction / 180 \* Math.PI) \* Speed);

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

particle.Size =Size;

}

//отрисовка

public void Render(Graphics g)

{

foreach (var particle in particles)

{

particle.Draw(g);

}

foreach (var point in impactPoints) // тут теперь impactPoints

{

point.Render(g); // это добавили

}

}

}

public class TopEmitter : Emitter

{

public int Width; // длина экрана

//заполнение характеристик частиц

public override void ResetParticle(Particle particle)

{

base.ResetParticle(particle); // вызываем базовый сброс частицы, там жизнь переопределяется и все такое

// а теперь тут уже подкручиваем параметры движения

particle.X = Particle.rand.Next(Width); // позиция X -- произвольная точка от 0 до Width

particle.Y = 0; // ноль -- это верх экрана

particle.SpeedY = 1; // падаем вниз по умолчанию

particle.SpeedX = Particle.rand.Next(-10, 10); // разброс влево и вправа у частиц

}

}

}

# 4 Код специальных точек

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing; // чтобы использовать Graphics

namespace Курсовая

{

public abstract class IImpactPoint

{

public float width;

public float height;

public float XBegin;

public float YBegin;

public int thickness;

public Color colorSquare;

// абстрактный метод с помощью которого будем изменять состояние частиц

public abstract void ImpactParticle(ParticleColorful particle);

// базовый класс для отрисовки прямоугольников

public virtual void Render(Graphics g)

{

g.DrawRectangle(new Pen(colorSquare, thickness), XBegin , YBegin, width,height);

}

}

public class ColorPoint : IImpactPoint

{

// изменение цвета цастиц,которые попали в область влияние Цветовых областей

public override void ImpactParticle(ParticleColorful particle)

{

if (particle.X >= 0 & particle.X <= width & particle.Y >= YBegin & particle.Y <= YBegin+ height)

{

particle.FromColor = Color.Red;

}

if (particle.X >= width & particle.X <= width\*2 & particle.Y >= YBegin & particle.Y <= YBegin + height)

{

particle.FromColor = Color.Orange;

}

if (particle.X >= width\*2 & particle.X <=width\*3 & particle.Y >= YBegin & particle.Y <= YBegin + height)

{

particle.FromColor = Color.Yellow;

}

if (particle.X >= width\*3 & particle.X <= width \* 4 & particle.Y >= YBegin & particle.Y <= YBegin + height)

{

particle.FromColor = Color.Green;

}

if (particle.X >= width \* 4 & particle.X <= width \* 5 & particle.Y >= YBegin & particle.Y <= YBegin + height)

{

particle.FromColor = Color.DodgerBlue;

}

if (particle.X >= width \* 5 & particle.X <= width \* 6 & particle.Y >= YBegin & particle.Y <= YBegin + height)

{

particle.FromColor = Color.Blue;

}

if (particle.X >= width \* 6 & particle.X <= width\*7 & particle.Y >= YBegin & particle.Y <= YBegin + height)

{

particle.FromColor = Color.Violet;

}

}

}

}

# 5 Код частиц

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing; // чтобы использовать Graphics

namespace Курсовая

{

public class Particle

{

public int Size; // радиус частицы

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

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

public int SpeedX; // скорость перемещения по оси X

public int SpeedY; // скорость перемещения по оси Y

public static Random rand = new Random();

public float Life; // запас здоровья частицы

public Color FromColor;

public Color ToColor;

public Particle()

{

// генерируем произвольное направление и скорость

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

Size = 2 + rand.Next(10);

Life = 20 + rand.Next(100);

}

public Particle(Particle particle)

{

this.X = particle.X;

this.Y = particle.Y;

this.SpeedX = particle.SpeedX;

this.SpeedY = particle.SpeedY;

this.Life = particle.Life;

this.Size = particle.Size;

}

//отрисовка

public virtual void Draw(Graphics g){ }

//отрисовка рамки при выделение мышкой

public virtual void DrawFrame(Graphics g){ }

//проверяет попала ли мышка в частицу

public virtual bool ifMouseInFigure(Graphics g, int xMouse, int yMouse)

{

return false;

}

//клонирование частиц

public virtual Particle Clone() {

ParticleColorful instanse = (ParticleColorful)Activator.CreateInstance(this.GetType());

instanse.Size = this.Size;

instanse.SpeedX = this.SpeedX;

instanse.SpeedY = this.SpeedY;

instanse.X = this.X;

instanse.Y = this.Y;

instanse.Life = this.Life;

return instanse;

}

//вывод информации о них

public void ShowInfo(Graphics g)

{

g.FillRectangle(

new SolidBrush(Color.FromArgb(125, Color.White)),

this.X,

this.Y - this.Size,

60,

50

);

g.DrawString(

$"X : {this.X}\n" +

$"Y : {this.Y}\n" +

$"Life : {this.Life}",

new Font("Verdana", 10),

new SolidBrush(Color.Black),

this.X,

this.Y - this.Size

);

}

}

//наследник класса Particle,окрашивает частицы

public class ParticleColorful : Particle

{

public ParticleColorful() { }

//клонирование

public override Particle Clone()

{

ParticleColorful instanse = (ParticleColorful)Activator.CreateInstance(this.GetType());

instanse.Size = this.Size;

instanse.SpeedX = this.SpeedX;

instanse.SpeedY = this.SpeedY;

instanse.X = this.X;

instanse.Y = this.Y;

instanse.Life = this.Life;

instanse.FromColor = this.FromColor;

instanse.ToColor = this.ToColor;

return instanse;

}

public ParticleColorful(ParticleColorful particleColorful)

{

this.X = particleColorful.X;

this.Y = particleColorful.Y;

this.Size = particleColorful.Size;

this.SpeedX = particleColorful.SpeedX;

this.SpeedY = particleColorful.SpeedY;

this.Life = particleColorful.Life;

this.FromColor = particleColorful.FromColor;

this.ToColor = particleColorful.ToColor;

}

//смешивание цветов

public static Color mixColor(Color color1, Color color2, float k)

{

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))

);

}

}

//наследник класса ParticleColorful, делает форму частиц кругом

public class ParticleCircle : ParticleColorful

{

//отрисовка кругов

public override void Draw(Graphics g)

{

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

var color = mixColor(ToColor, FromColor, k);

var b = new SolidBrush(color);

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

b.Dispose();

g.DrawLine(new Pen(Brushes.Green), new Point((int)X, (int)Y),

new Point((int)(X + (int)SpeedX),

(int)(Y + Size / 4 \* 3)));

}

//отрисовка рамки для попадание мышкой в частицу

public override void DrawFrame(Graphics g)

{

g.DrawEllipse(new Pen(Brushes.Black), X - Size,Y - Size,Size \* 2,Size \* 2);

}

//проверяет попала ли мышка в частицу

public override bool ifMouseInFigure(Graphics g,int xMouse,int yMouse)

{

float gX = xMouse - this.X;

float gY = yMouse - this.Y;

double r = Math.Sqrt(gX \* gX + gY \* gY); // считаем расстояние от центра точки до центра частицы

if (r + this.Size <= this.Size \* 2 || r + this.Size <= this.Size \* 2) // если частица оказалось внутри эллипса

{

return true;

}

return false;

}

}

//наследник класса ParticleColorful, делает форму частиц квадратом

public class ParticleSquare : ParticleColorful

{

//отрисовка квадаратов

public override void Draw(Graphics g)

{

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

var color = mixColor(ToColor, FromColor, k);

var b = new SolidBrush(color);

g.FillRectangle(b, X, Y, Size, Size);

b.Dispose();

g.DrawLine(new Pen(Brushes.Green), new Point((int)(X + Size / 2), (int)(Y + Size / 2)),

new Point((int)(X + Size / 2 + (int)SpeedX),

(int)(Y + Size / 4 \* 3)));

}

//отрисовка рамки для попадание мышкой в частицу

public override void DrawFrame(Graphics g)

{

g.DrawRectangle(new Pen(Color.Black),X, Y, Size, Size);

}

//проверяет попала ли мышка в частицу

public override bool ifMouseInFigure(Graphics g, int xMouse, int yMouse)

{

float centerX = this.X + this.Size / 2;

float centerY = this.Y + this.Size / 2;

// проверяю, находится ли точка внутри прямоугольника

if (xMouse <= centerX + this.Size / 2 && xMouse >= centerX - this.Size / 2 &&

yMouse <= centerY + this.Size / 2 && yMouse >= centerY - this.Size / 2)

{

return true;

}

return false;

}

}

//наследник класса ParticleColorful, делает форму частиц звездой

public class ParticleStar : ParticleColorful

{

//отрисовка Звезды

public override void Draw(Graphics g)

{

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

var color = mixColor(ToColor, FromColor, k);

var b = new SolidBrush(color);

PointF[] points = new PointF[2 \* 5 + 1];

double a = 0, da = Math.PI / 5, l;

for (int k1 = 0; k1 < 2 \* 5 + 1; k1++)

{

l = k1 % 2 == 0 ? Size \* 2 : Size;

points[k1] = new PointF((float)(X + l \* Math.Cos(a)), (float)(Y + l \* Math.Sin(a)));

a += da;

}

g.FillPolygon(b, points);

b.Dispose();

g.DrawLine(new Pen(Brushes.Green), new Point((int)X, (int)Y),

new Point((int)(X + (int)SpeedX),

(int)(Y + Size / 4 \* 3)));

}

//отрисовка рамки для попадание мышкой в частицу

public override void DrawFrame(Graphics g)

{

double alpha = 0; // поворот

PointF[] points = new PointF[2 \* 5 + 1];

double a = alpha, da = Math.PI / 5, l;

for (int k = 0; k < 2 \* 5 + 1; k++)

{

l = k % 2 == 0 ? this.Size \* 2 : this.Size;

points[k] = new PointF((float)(this.X + l \* Math.Cos(a)), (float)(this.Y + l \* Math.Sin(a)));

a += da;

}

g.DrawLines(Pens.Black, points);

}

//проверяет попала ли мышка в частицу

public override bool ifMouseInFigure(Graphics g, int xMouse, int yMouse)

{

float gX = xMouse - this.X;

float gY = yMouse - this.Y;

double r = Math.Sqrt(gX \* gX + gY \* gY); // считаем расстояние от центра точки до центра частицы

if (r + this.Size <= this.Size \* 2 || r + this.Size <= this.Size \* 2) // если частица оказалось внутри эллипса

{

return true;

}

return false;

}

}

//наследник класса ParticleColorful, делает форму частиц снежинкой

public class ParticleSnowflake : ParticleColorful

{

//толщина линии из которой строится снежинка

public float thicknessLines;

//клонирование

public override Particle Clone()

{

ParticleSnowflake instanse = (ParticleSnowflake)Activator.CreateInstance(this.GetType());

instanse.Size = this.Size;

instanse.SpeedX = this.SpeedX;

instanse.SpeedY = this.SpeedY;

instanse.X = this.X;

instanse.Y = this.Y;

instanse.Life = this.Life;

instanse.FromColor = this.FromColor;

instanse.ToColor = this.ToColor;

instanse.thicknessLines = this.thicknessLines;

return instanse;

}

//отрисовка снежинки

public override void Draw(Graphics g)

{

thicknessLines = Size / 15;

Point[] points ={

new Point(X, Y),new Point(X+Size, Y),

new Point(X, Y),new Point(X-Size,Y),

new Point(X, Y),new Point(X, Y+Size),

new Point(X, Y),new Point(X, Y-Size),

new Point(X, Y),new Point(X+(Size/2), Y+(Size/2)),

new Point(X, Y),new Point(X-(Size/2), Y+(Size/2)),

new Point(X, Y),new Point(X+(Size/2),Y-(Size/2)),

new Point(X, Y),new Point(X-(Size/2), Y-(Size/2)),

};

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

//Draw lines to screen.

g.DrawLines(new Pen(mixColor(ToColor, FromColor, k), thicknessLines), points);

g.DrawLine(new Pen(mixColor(ToColor, FromColor, k), thicknessLines), new Point(X, Y - (Size / 2)),new Point(X - (Size / 2), Y - Size));

g.DrawLine(new Pen(mixColor(ToColor, FromColor, k), thicknessLines), new Point(X, Y - (Size / 2)), new Point(X + (Size / 2), Y - Size));

g.DrawLine(new Pen(mixColor(ToColor, FromColor, k), thicknessLines), new Point(X, Y + (Size / 2)),new Point(X - (Size / 2), Y + Size));

g.DrawLine(new Pen(mixColor(ToColor, FromColor, k), thicknessLines), new Point(X, Y + (Size / 2)), new Point(X + (Size / 2), Y + Size));

g.DrawLine(new Pen(mixColor(ToColor, FromColor, k), thicknessLines), new Point(X+ (Size / 2), Y ), new Point(X + Size, Y + (Size/2)));

g.DrawLine(new Pen(mixColor(ToColor, FromColor, k), thicknessLines), new Point(X + (Size / 2), Y), new Point(X + Size, Y - (Size / 2)));

g.DrawLine(new Pen(mixColor(ToColor, FromColor, k), thicknessLines), new Point(X - (Size / 2), Y), new Point(X - Size, Y + (Size / 2)));

g.DrawLine(new Pen(mixColor(ToColor, FromColor, k), thicknessLines), new Point(X - (Size / 2), Y), new Point(X - Size, Y - (Size / 2)));

g.DrawLine(new Pen(Brushes.Green, thicknessLines), new Point((int)X, (int)Y),new Point((int)(X + (int)SpeedX), (int)(Y + Size )));

}

//проверяет попала ли мышка в частицу

public override bool ifMouseInFigure(Graphics g, int xMouse, int yMouse)

{

// проверяю, находится ли точка внутри прямоугольника

if (xMouse <= this.X+Size && xMouse >= this.X - Size &&

yMouse <= this.Y+Size && yMouse >= this.Y - Size )

{

return true;

}

return false;

}

}

}

# 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 Курсовая

# {

# public partial class Form1 : Form

# {

# Emitter emitter;

# public Color colorPicture=Color.White;

# bool ifRun = true;

# bool stepPermission = false;

# int xMouse;

# int yMouse;

# public Form1()

# {

# InitializeComponent();

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

# // а тут теперь вручную создаем

# emitter = new TopEmitter

# {

# Width = picDisplay.Width,

# gravitationY = 5

# };

# //задаем начальные значения частиц исходя из начальных значений трекбаров

# emitter.Size = tbSize.Value \* 5;

# emitter.Life = 10 \* tbLife.Value;

# emitter.ParticlesPerTick = tbNumber.Value;

# emitter.figure = "circle";

# }

# 

# private void timer1\_Tick(object sender, EventArgs e)

# {

# //если пользователь задал скорость больше 0

# if ((tbSpeed.Value != 0 && ifRun) || stepPermission)

# {

# //то выполняем остальное

# emitter.UpdateState();

# }

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

# {

# g.Clear(colorPicture);

# emitter.Render(g);

# foreach (var particle in emitter.particles)

# {

# if (particle.ifMouseInFigure(g, xMouse ,yMouse))

# {

# //если мышка попала в частицу,то рисуем рамку вокруг частицы и выводим информацию

# particle.DrawFrame(g);

# particle.ShowInfo(g);

# }

# }

# }

# picDisplay.Invalidate();

# stepPermission = false;

# }

# //количество тиков

# private void ChangeTick()

# {

# switch (tbSpeed.Value)

# {

# case 0:

# ifRun = false;

# break;

# case 1:

# emitter.tickRate = 30;

# break;

# case 2:

# emitter.tickRate = 25;

# break;

# case 3:

# emitter.tickRate = 20;

# break;

# case 4:

# emitter.tickRate = 15;

# break;

# case 5:

# emitter.tickRate = 10;

# break;

# case 6:

# emitter.tickRate = 7;

# break;

# case 7:

# emitter.tickRate = 5;

# break;

# case 8:

# emitter.tickRate = 3;

# break;

# case 9:

# emitter.tickRate = 2;

# break;

# case 10:

# emitter.tickRate = 1;

# break;

# }

# }

# //кнопочки

# private void Start\_Click(object sender, EventArgs e)

# {

# ifRun = true;

# ChangeTick();

# }

# private void Stop\_Click(object sender, EventArgs e)

# {

# ifRun = false;

# }

# private void Step\_Click(object sender, EventArgs e)

# {

# ifRun = false;

# if (emitter.currentHistoryIndex < emitter.particlesHistory.Count - 1 && emitter.currentHistoryIndex != 19)

# {

# //поставить значения дальше по списку

# emitter.particles.RemoveRange(0, emitter.particles.Count);

# foreach (ParticleColorful particle in emitter.particlesHistory[emitter.currentHistoryIndex + 1])

# {

# ParticleColorful part = (ParticleColorful)particle.Clone();

# emitter.particles.Add(part);

# }

# emitter.currentHistoryIndex++;

# }

# else

# {

# emitter.tickCount += (emitter.tickRate - emitter.tickCount % emitter.tickRate);

# stepPermission = true;

# }

# }

# private void picDisplay\_MouseMove(object sender, MouseEventArgs e)

# {

# xMouse = e.X;

# yMouse = e.Y;

# }

# private void StepBack\_Click(object sender, EventArgs e)

# {

# ifRun = false;

# if (emitter.currentHistoryIndex >= 1)

# {

# //вернуться на значения из списка

# emitter.particles.RemoveRange(0, emitter.particles.Count);

# foreach (ParticleColorful particle in emitter.particlesHistory[emitter.currentHistoryIndex - 1])

# {

# ParticleColorful part = (ParticleColorful)particle.Clone();

# emitter.particles.Add(part);

# }

# emitter.currentHistoryIndex--;

# }

# }

# private void tbSpeed\_ValueChanged(object sender, EventArgs e)

# {

# ifRun = true;

# ChangeTick();

# }

# private void tbNumber\_Scroll\_1(object sender, EventArgs e)

# {

# emitter.ParticlesPerTick = tbNumber.Value;

# }

# private void tbSize\_Scroll(object sender, EventArgs e)

# {

# emitter.Size = 5 \* tbSize.Value;

# }

# //рандомный цвет частиц

# private void RandomColorParticles\_Click(object sender, EventArgs e)

# {

# Random random = new Random();

# int R = random.Next(255);

# int G = random.Next(255);

# int B = random.Next(255);

# emitter.ColorFrom = Color.FromArgb(R, G, B);

# emitter.ColorTo = colorPicture;

# }

# //рандомный цвет фона

# private void RandomColorPictures\_Click(object sender, EventArgs e)

# {

# Random random = new Random();

# int R = random.Next(255);

# int G = random.Next(255);

# int B = random.Next(255);

# var g = Graphics.FromImage(picDisplay.Image);

# colorPicture = Color.FromArgb(R, G, B);

# emitter.ColorTo = colorPicture;

# }

# //Выбрать цвет частиц

# private void ColorParticles\_Click(object sender, EventArgs e)

# {

# ColorDialog MyDialog = new ColorDialog();

# MyDialog.AllowFullOpen = false;

# MyDialog.Color = emitter.ColorFrom;

# if (MyDialog.ShowDialog() == DialogResult.OK)

# emitter.ColorFrom = MyDialog.Color;

# emitter.ColorTo = colorPicture;

# }

# //Выбрать цвет фона

# private void ColorPictures\_Click(object sender, EventArgs e)

# {

# ColorDialog MyDialog = new ColorDialog();

# MyDialog.AllowFullOpen = false;

# MyDialog.Color = colorPicture;

# if (MyDialog.ShowDialog() == DialogResult.OK)

# colorPicture = MyDialog.Color;

# emitter.ColorTo = colorPicture;

# }

# private void cmbForm\_SelectedIndexChanged(object sender, EventArgs e)

# {

# //комбобокс-для формы частиц

# switch (cmbForm.Text)

# {

# case "Круг":

# emitter.figure = "circle";

# break;

# case "Квадрат":

# emitter.figure = "square";

# break;

# case "Звезда":

# emitter.figure = "star";

# break;

# case "Снежинки":

# emitter.figure = "snowflake";

# break;

# }

# //очистка всех списков,сверка количества

# emitter.particles.Clear();

# emitter.particlesHistory.Clear();

# emitter.currentHistoryIndex = 0;

# tbNumber\_Scroll\_1(sender, e);

# }

# private void tbLife\_Scroll(object sender, EventArgs e)

# {

# emitter.Life =10\* tbLife.Value;

# }

# private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

# {

# //комбобокс-для режима работы

# switch (comboBox1.Text) {

# case "Простой":

# {

# //отключение влияния точек-окрашивателей

# emitter.impactPoints.Clear();

# }

# break;

# case "Окрашивание":

# {

# //создание точек-окрашивателей

# float w = picDisplay.Width / 7;

# emitter.impactPoints.Add(new ColorPoint

# {

# colorSquare=Color.Red,

# thickness=4,

# XBegin=0,

# YBegin=100,

# height=40,

# width=w

# });

# emitter.impactPoints.Add(new ColorPoint

# {

# colorSquare = Color.Orange,

# thickness = 4,

# XBegin = w+4,

# YBegin = 100,

# height = 40,

# width =w

# });

# emitter.impactPoints.Add(new ColorPoint

# {

# colorSquare = Color.Yellow,

# thickness = 4,

# XBegin = w\*2+4,

# YBegin = 100,

# height = 40,

# width = w

# });

# emitter.impactPoints.Add(new ColorPoint

# {

# colorSquare = Color.Green,

# thickness = 4,

# XBegin = w\*3+4,

# YBegin = 100,

# height = 40,

# width = w

# });

# emitter.impactPoints.Add(new ColorPoint

# {

# colorSquare = Color.DodgerBlue,

# thickness = 4,

# XBegin = w\*4+4,

# YBegin = 100,

# height = 40,

# width = w

# });

# emitter.impactPoints.Add(new ColorPoint

# {

# colorSquare = Color.Blue,

# thickness = 4,

# XBegin = w\*5+4,

# YBegin = 100,

# height = 40,

# width = w

# });

# emitter.impactPoints.Add(new ColorPoint

# {

# colorSquare = Color.Violet,

# thickness = 4,

# XBegin = w\*6+4,

# YBegin = 100,

# height = 40,

# width = w

# });

# }

# break;

# }

# }

# }

# }

# 7 Описание работы интерфейса

Нажимая RandomColorParticles и RandomColorPictures **Случайный цвет частиц и Случайный цвет фона** мы меняем цвет фона и цвет новых частицы:

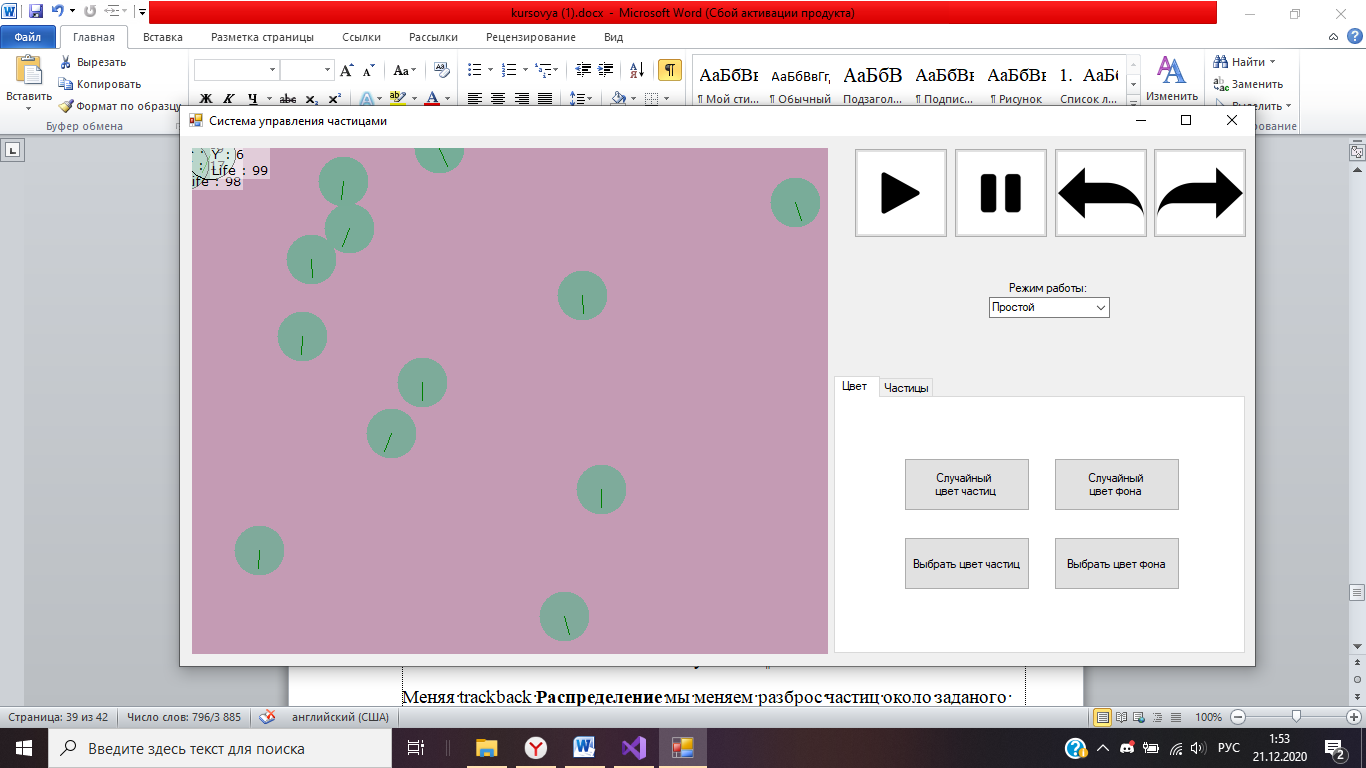


Рисунок 7.1

Нажимая ColorParticles и ColorPictures **Выбрать цвет частиц и Выбрать цвет фона** мы меняем цвет фона и цвет новых частицы:

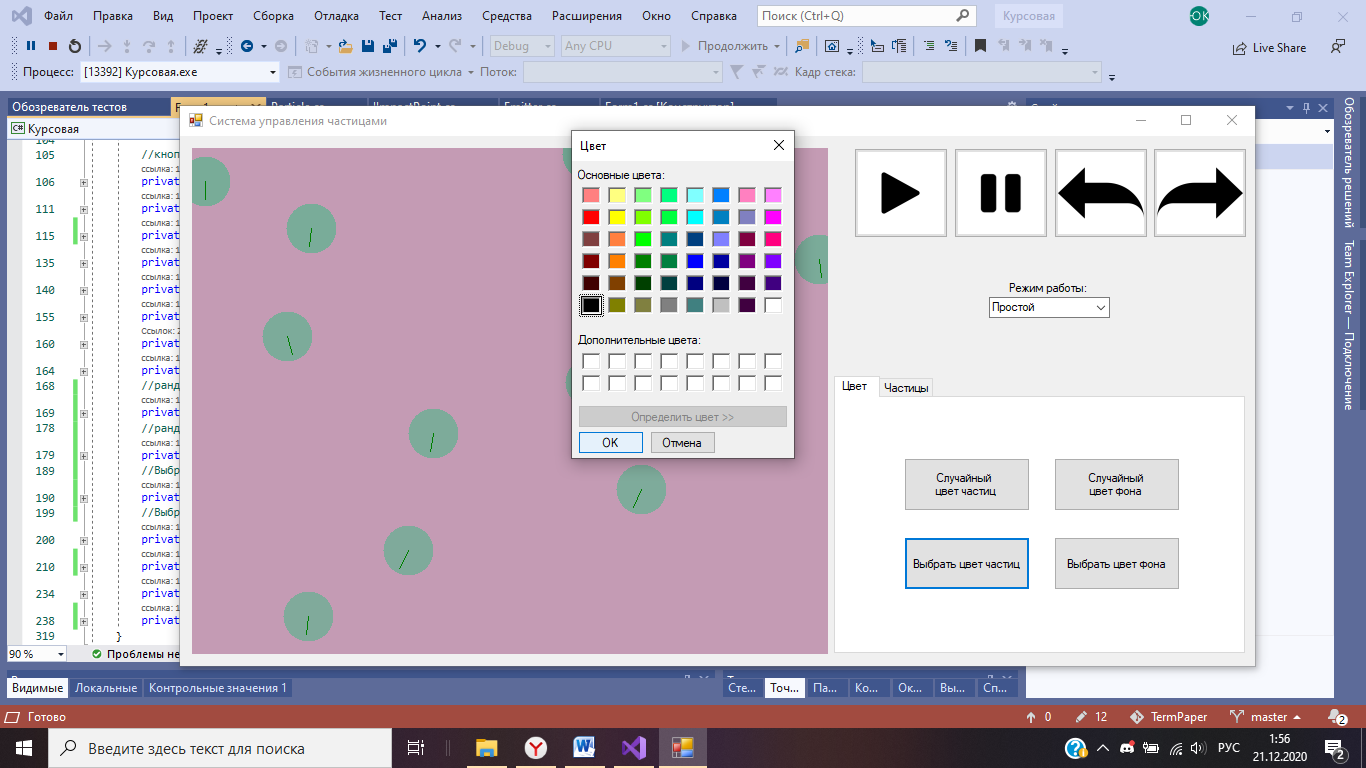


Рисунок 7.2

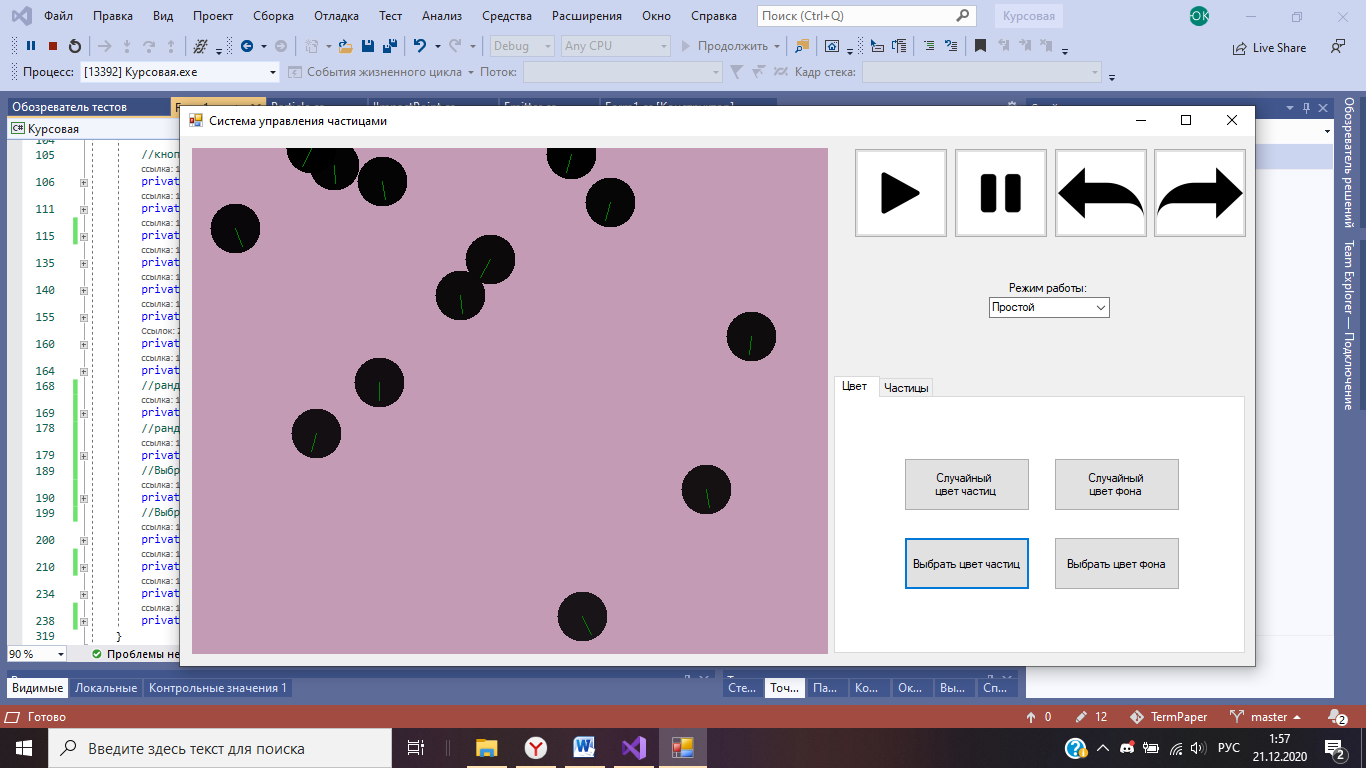


Рисунок 7.3

Меняя comboBox1 **Режим работы:окрашивание/простой,** мы управляем режимом падения частиц.Если окрашивание,то как только частица попадает в область окрашивания,то меняет цвет в зависимости о цвета области.Если простой режим ,то окрашивание частиц не происходит.

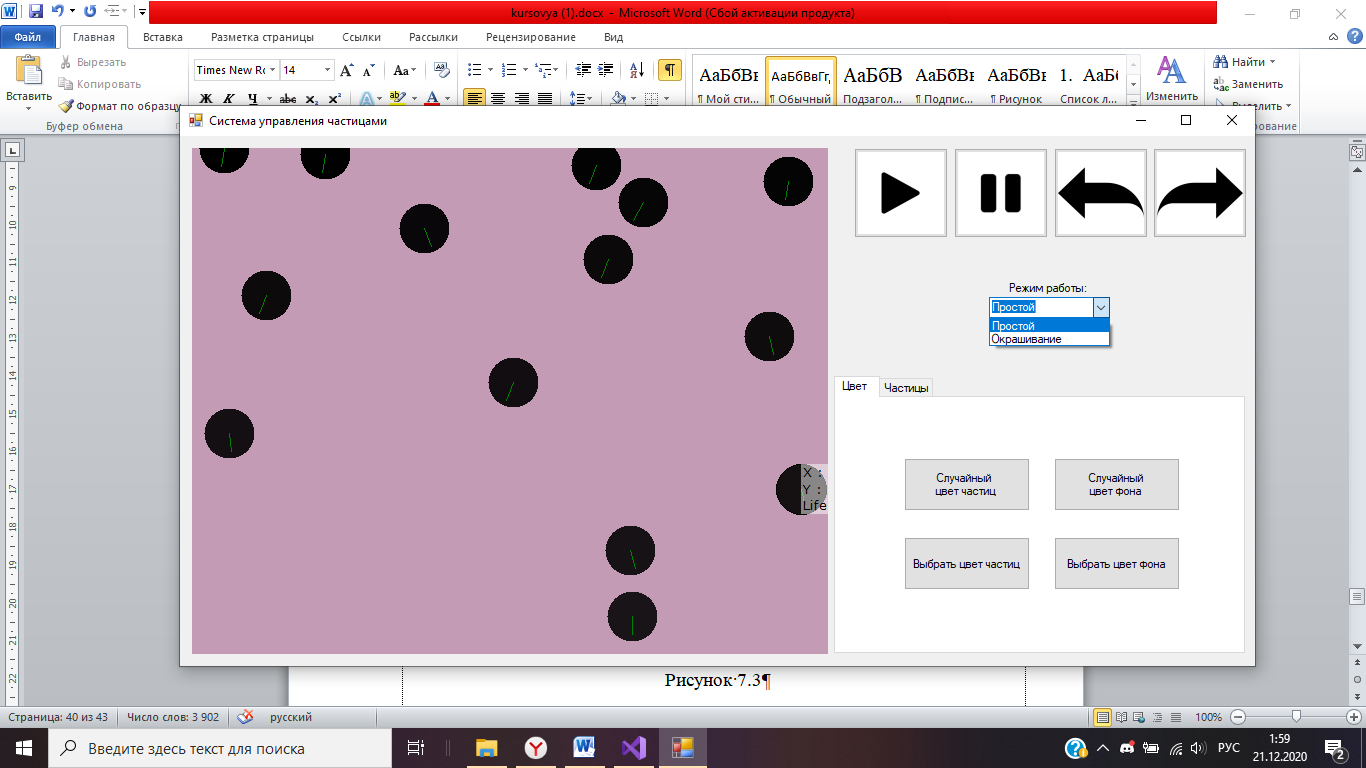


Рисунок 7.4

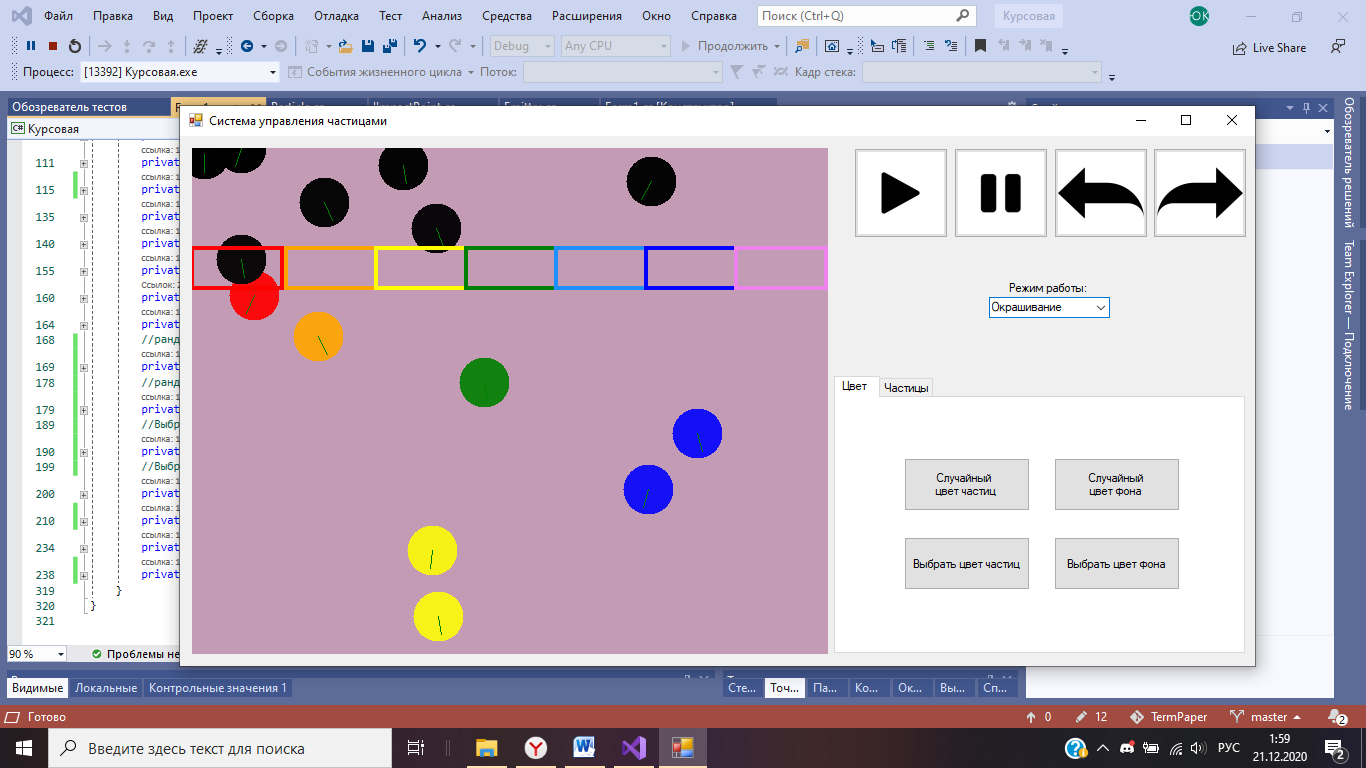


Рисунок 7.5

Меняя tbSize **Размер частиц,** мы управляем размером новых частиц.

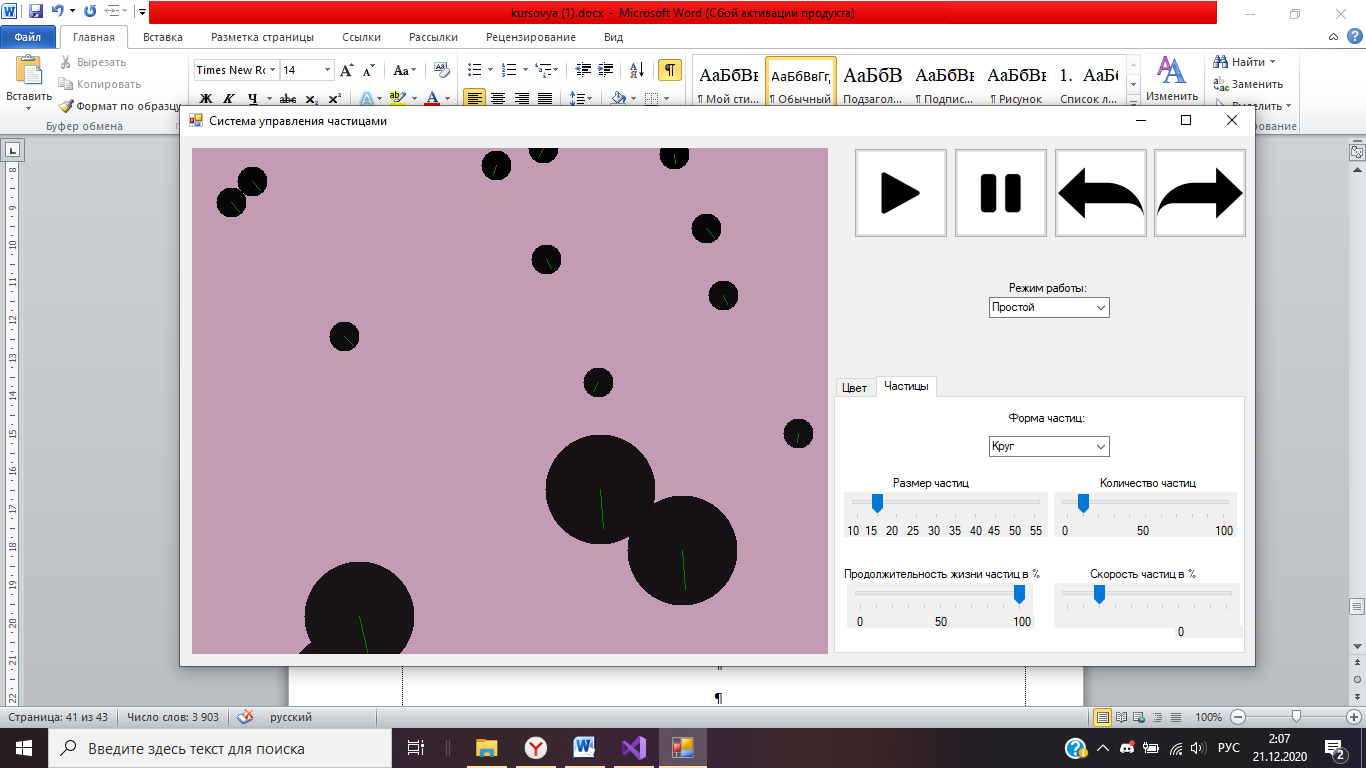


Рисунок 7.6

Меняя tbNumber **Количество частиц,** мы управляем количеством создающихся частиц.

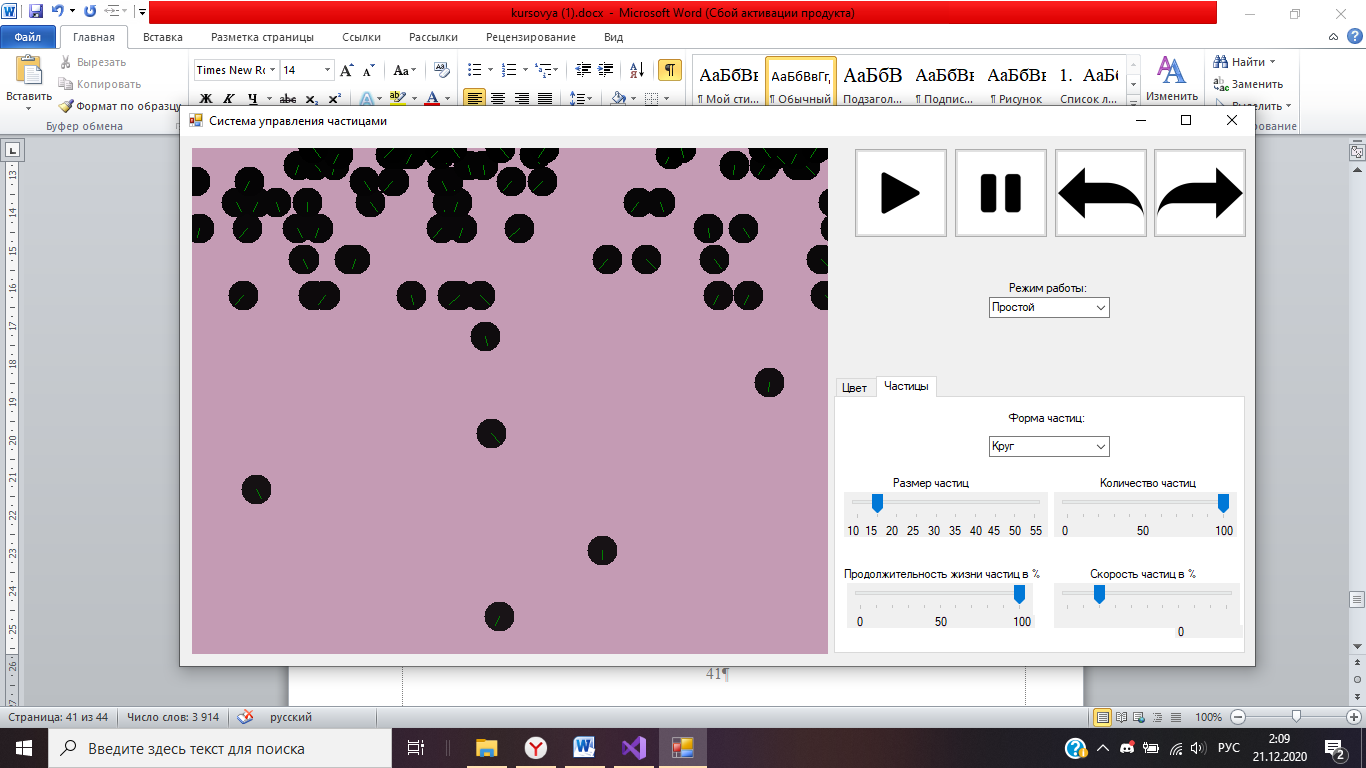


Рисунок 7.7

Меняя tbLife **Продолжительность жизни частиц,** мы управляем продолжительностью жизни создающихся частиц(видно по яркости частицы).

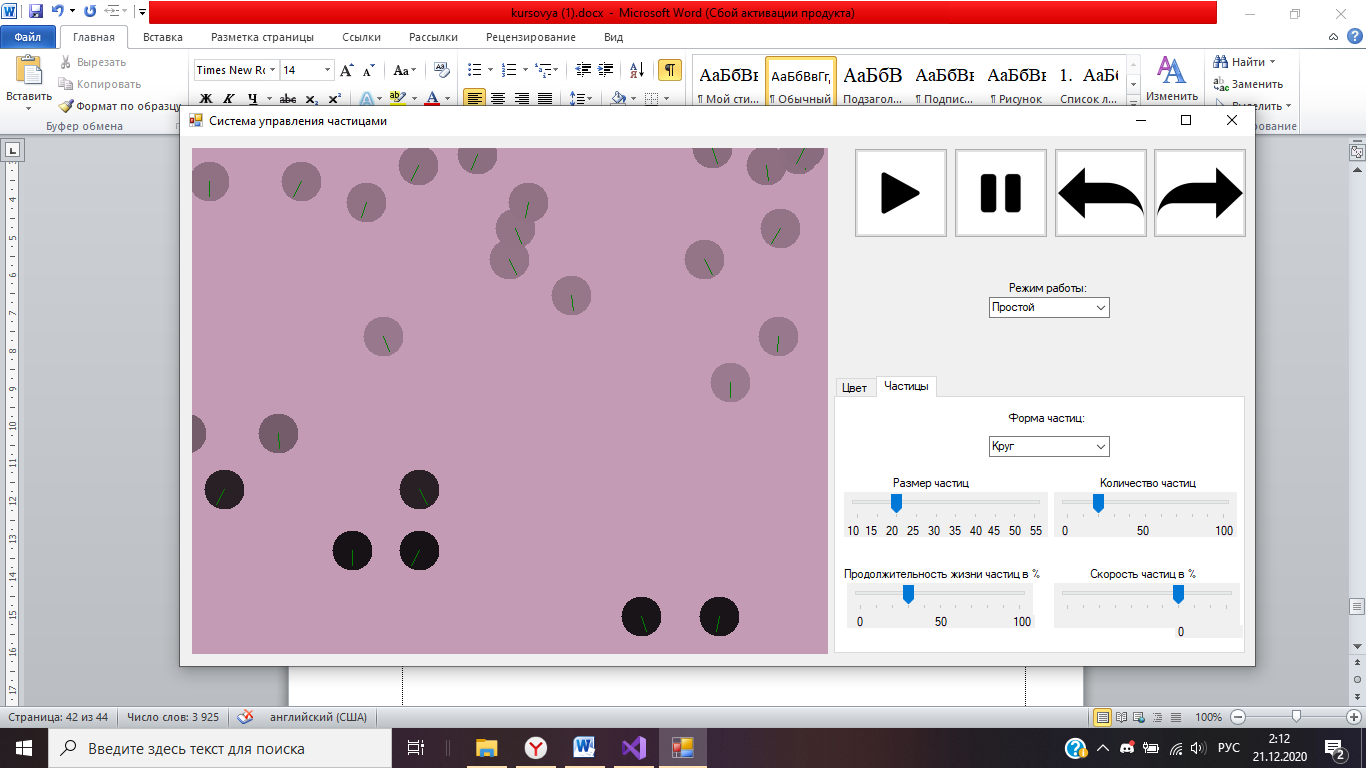


Рисунок 7.8

Меняя cmbForm **Форма частиц:круг/квадрат/звезда/снежинка,** мы управляем формой новых частиц.

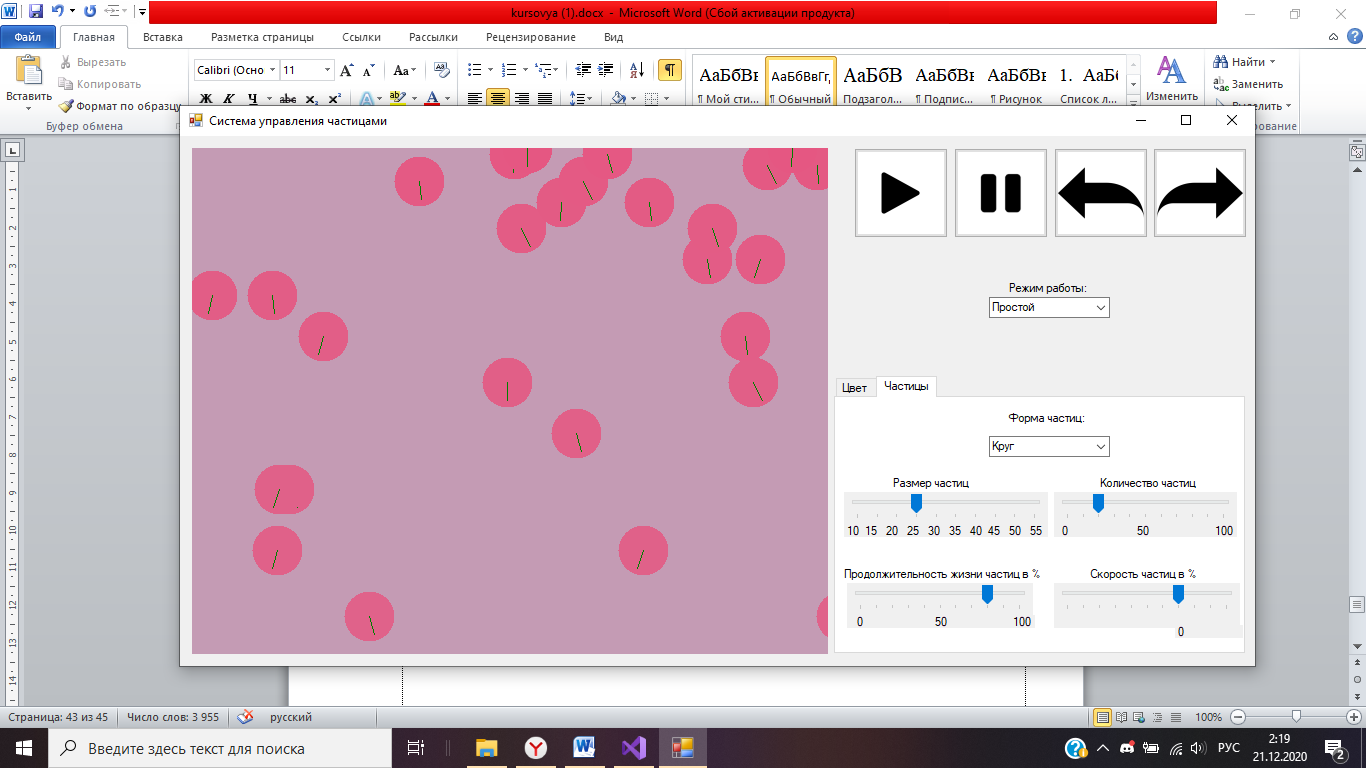


Рисунок 7.9-Форма :круг

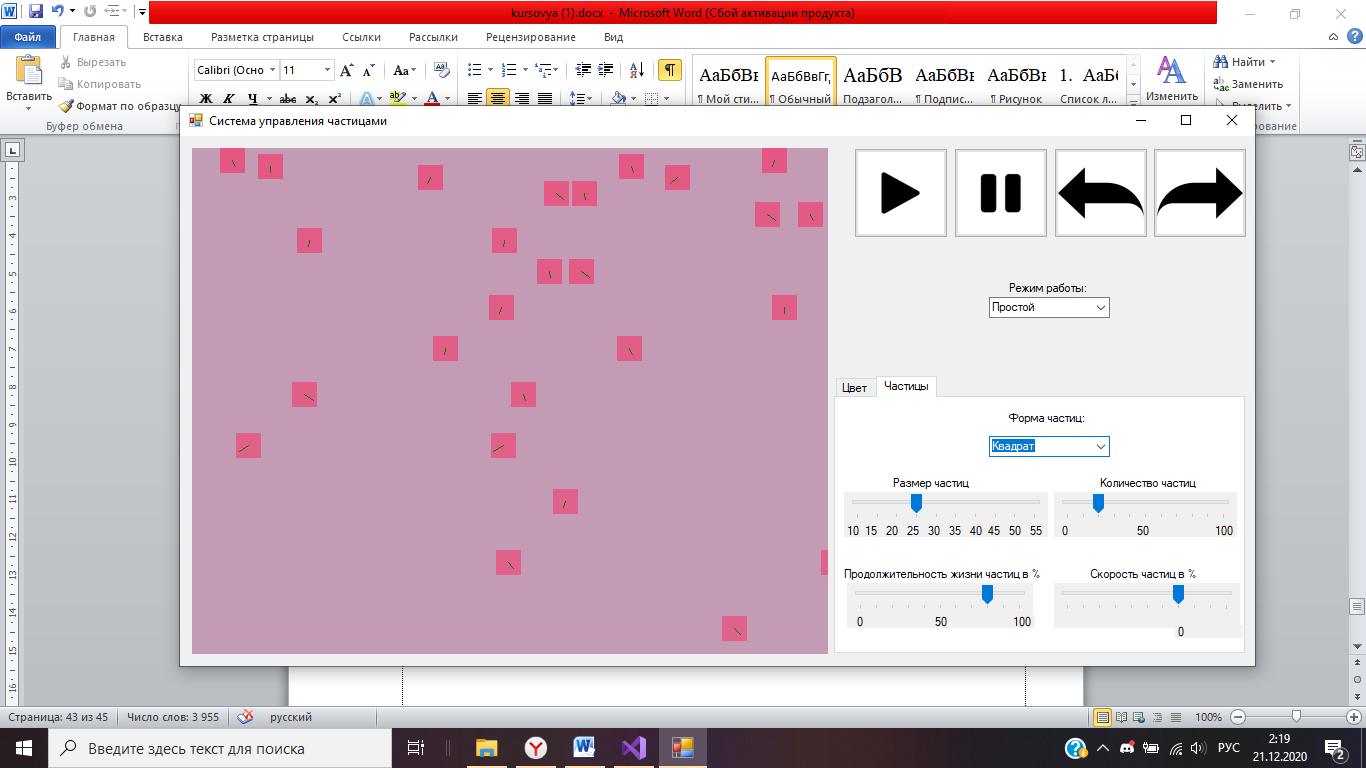


Рисунок 7.10-Форма:квадрат

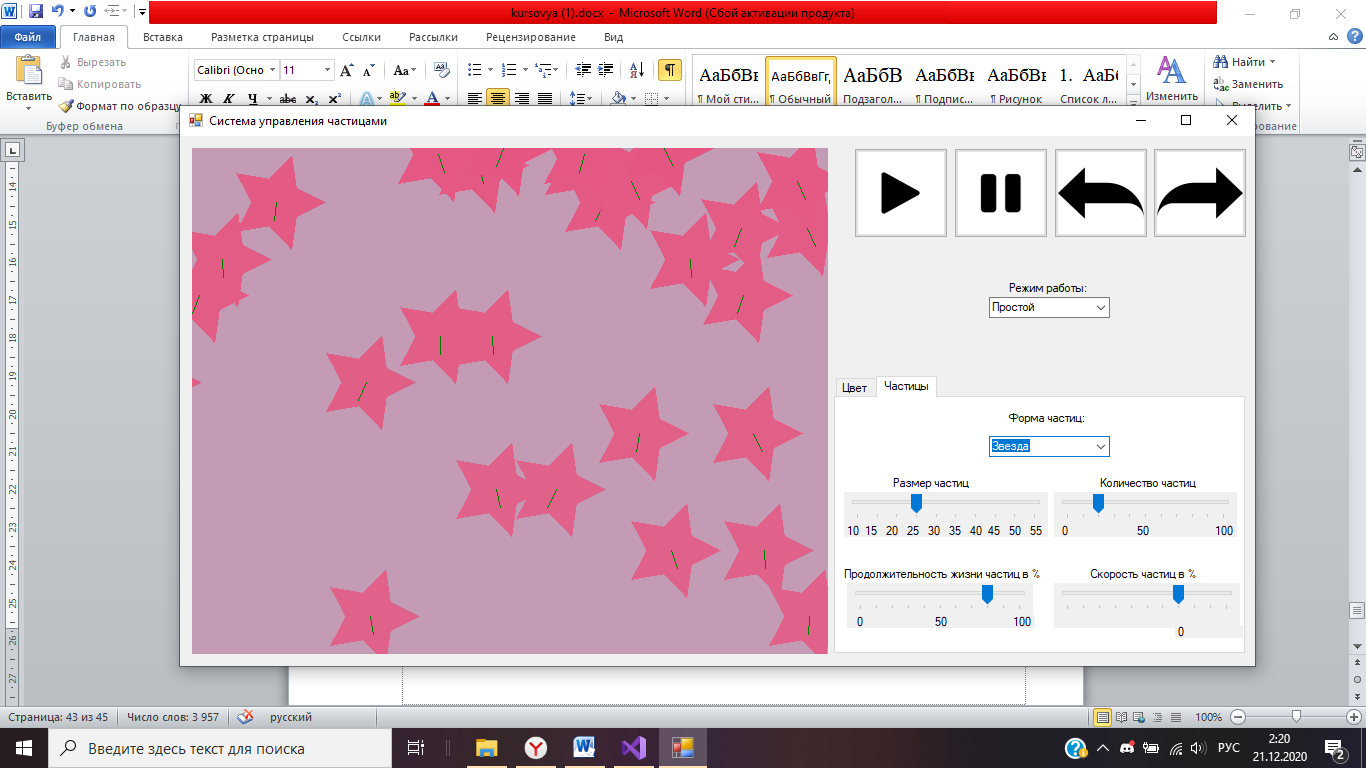


Рисунок 7.11-Форма:звезда

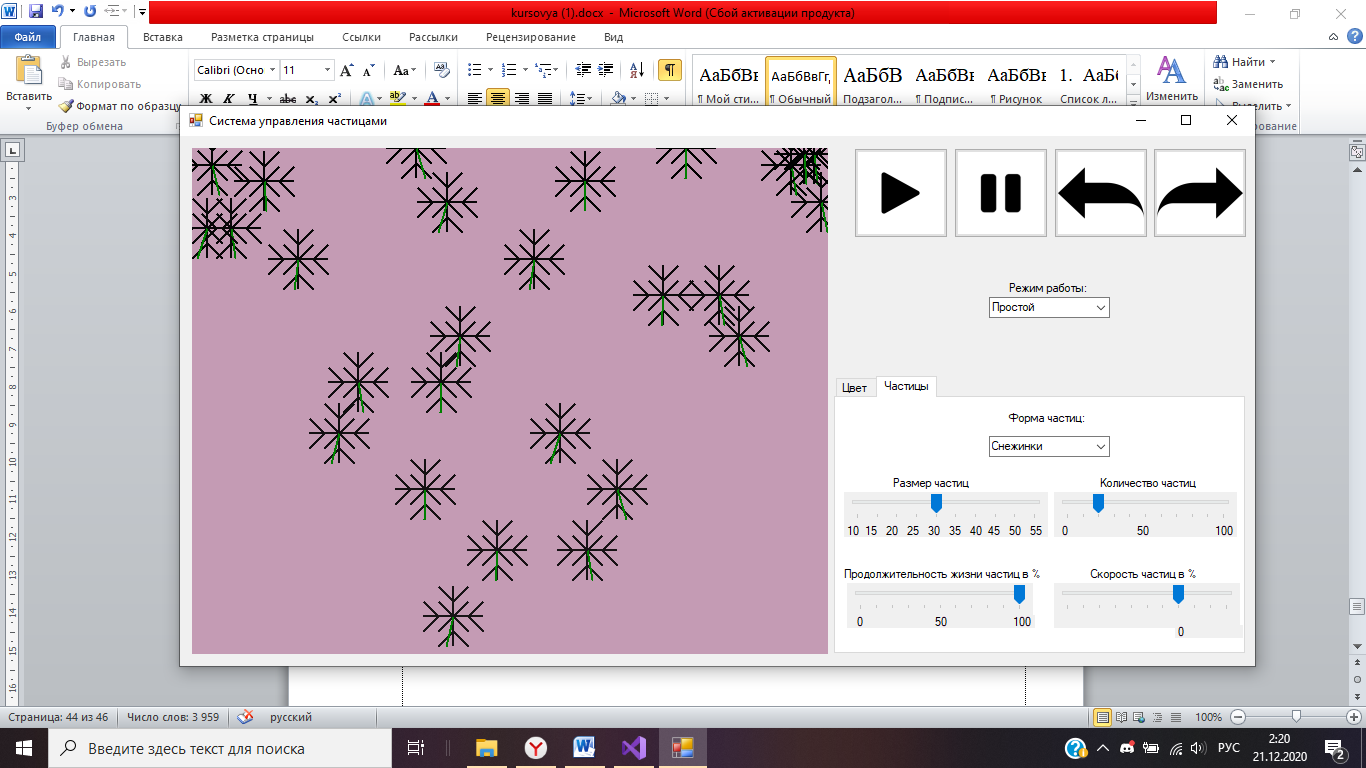


Рисунок 7.12-Форма:снежинки

Меняя tbSpeed **Скорость частиц,** мы управляем скоростью падения частиц. Нет возможности отобразить это в рисунке.

Нажимая Stop,StepBack, Step, Start- **Стоп** а затем **Шаг назад(стрелка влево), Шаг вперед(Стрелка вправо), Старт** мы останавливаем падение и возвращаемся на один шаг назад,делаем шаг вперед и продолжается падение.При нажатие Stop,StepBack, Step система падения останавливается, пока не нажмется Start или еще шаги вперед/назад.

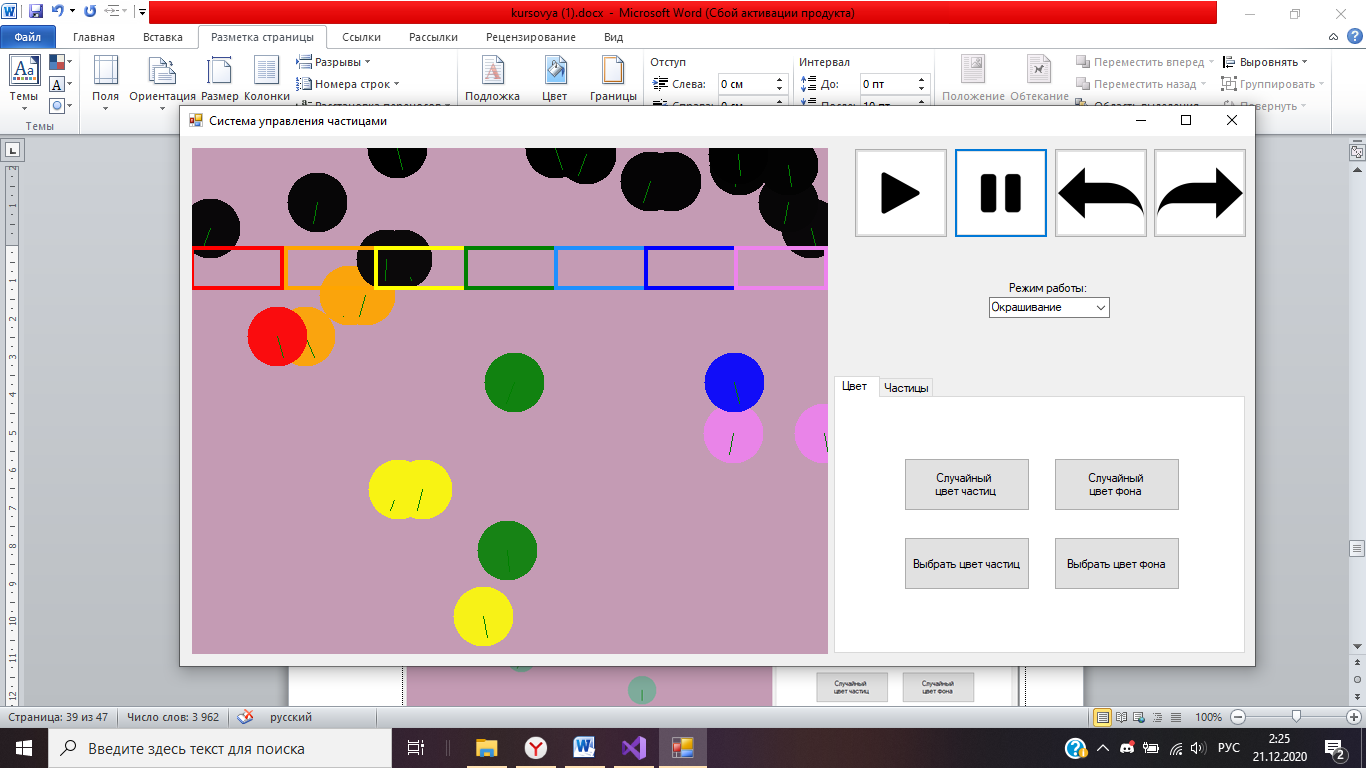


Рисунок 7.13-Остановка

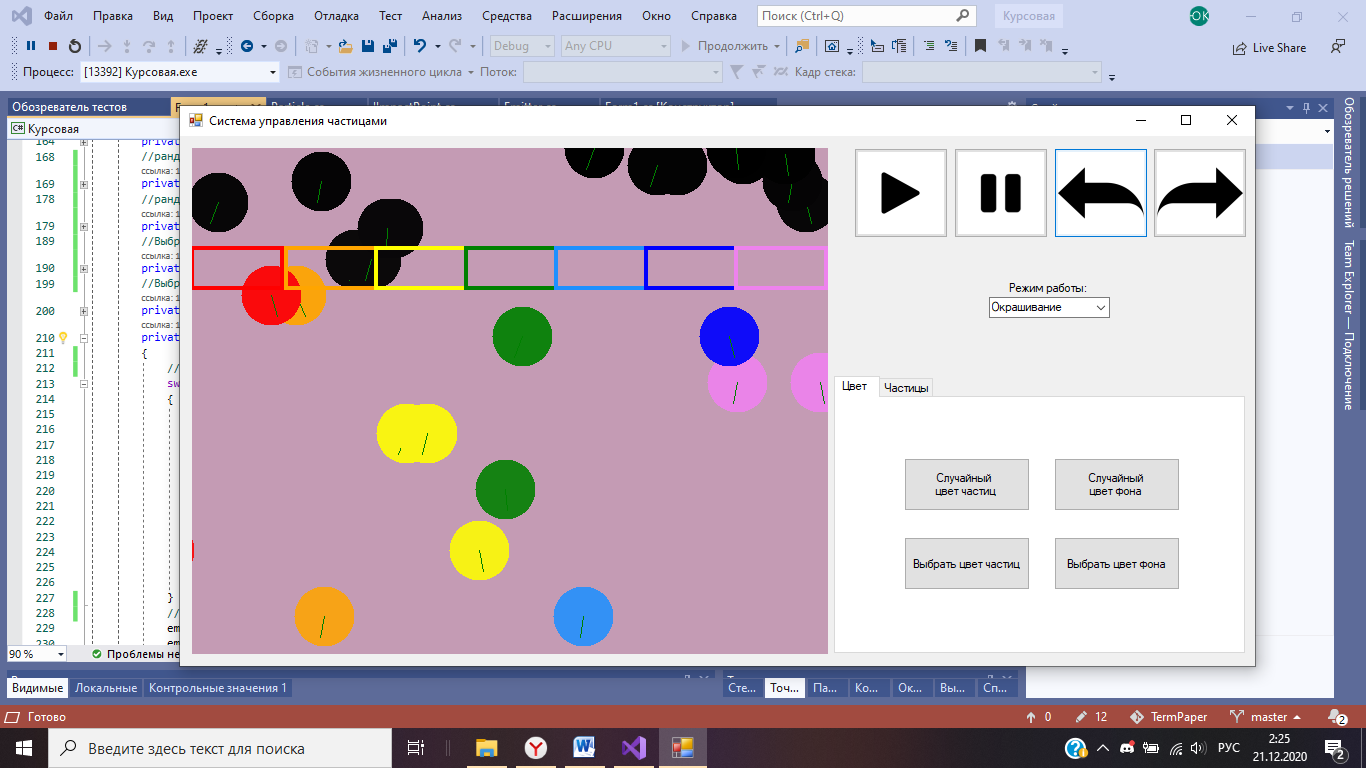


Рисунок 7.14-Шаг назад

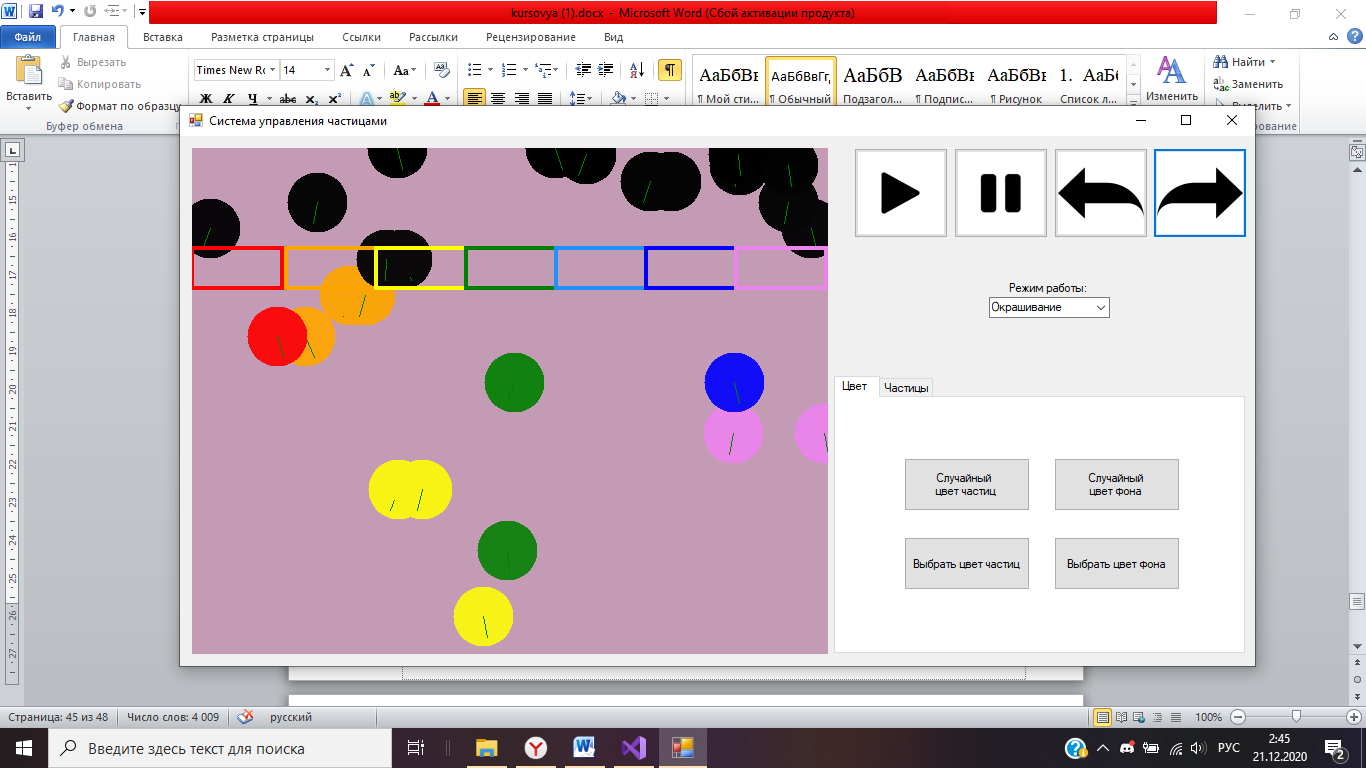


Рисунок 7.15-Шаг вперед

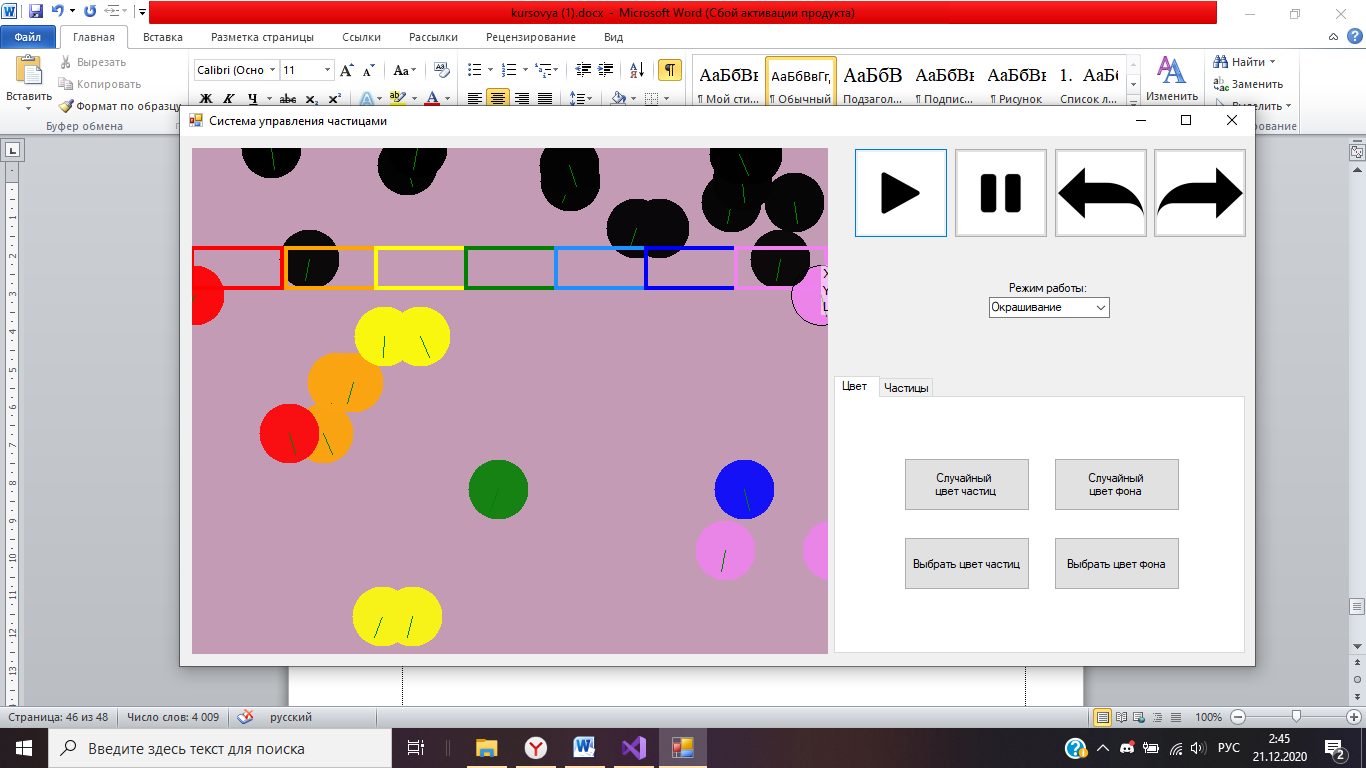


Рисунок 7.16-Старт

При наведение мышки на частицу ,выводится информация о частице(Координаты,жизнь):

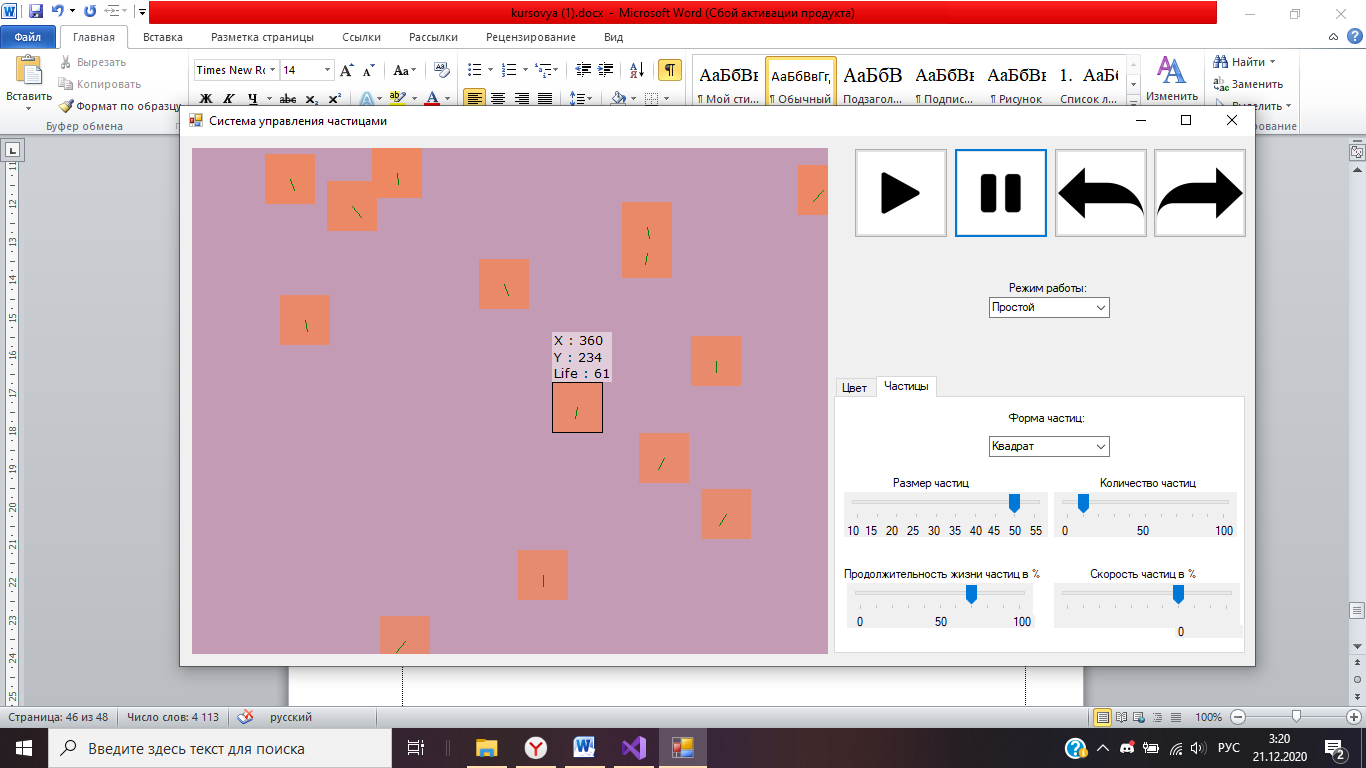


Рисунок 7.17

# Заключение

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

* Ползунок «Скорость», для управления скоростью падения частиц
* Ползунок «Продолжительность жизни», для управления продолжительности жизни частиц(видно по яркости частицы)
* Ползунок «Количество частиц» - для управления количеством частицы на форме
* Ползунок «Размер частиц» - для управления размером частицы на форме
* Кнопка «Старт», для запуска падения
* Кнопка «Стоп», для остановки падения
* Кнопка «Шаг назад», для запуска падения на один тик назад
* Кнопка «Шаг вперед», для запуска падения на один тик вперед
* Кнопка «Случайный цвет фона» и «Случайный цвет частиц», для изменения цвета частиц и фона на случайный
* Кнопка «Выбрать цвет частиц» и «Выбрать цвет фона» , для изменения цвета частиц и фона на выбранный
* Бокс «Форма частиц» , дает возможность задать форму частиц: круг/квадрат/звезда/снежинка
* Бокс «Режим работы» , дает возможность задать режим работы : простой/окрашивание
* Специальная точка «Радуга» которая изменяет цвет частиц после того как они проходят определенные координаты
* Специальная возможность, которая при наведение мышки на частицу, выводит информацию о ней

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] // olive.tealeaf.su : чаинка, 2020. URL: <http://olive.tealeaf.su/particle-system.html> (дата обращения: 01.12.2020).