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 |

|  |
| --- |
| 1.001.00.00 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 2 | |
| 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 | 19 / 12 / 2020 г. |
| Руководитель курсовой работы |  |  | Столбов А.Б. |
|  | подпись | Фамилия И.О. |

Содержание

[Введение 4](#_Toc59719916)

[1 Индивидуальный вариант 5](#_Toc59719917)

[2 Внешний вид главного окна 6](#_Toc59719918)

[3 Код эмиттера 7](#_Toc59719919)

[4 Код частиц 11](#_Toc59719920)

[6 Код формы 14](#_Toc59719921)

[7 Описание работы интерфейса 19](#_Toc59719922)

[Заключение 27](#_Toc59719923)

[Список использованной литературы 28](#_Toc59719924)

# 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:

1. изменения направления эмиттера
2. градуса распределения частиц
3. изменения скорости вылета частиц
4. изменения количества частиц за тик
5. продолжительности жизни

# 2 Внешний вид главного окна

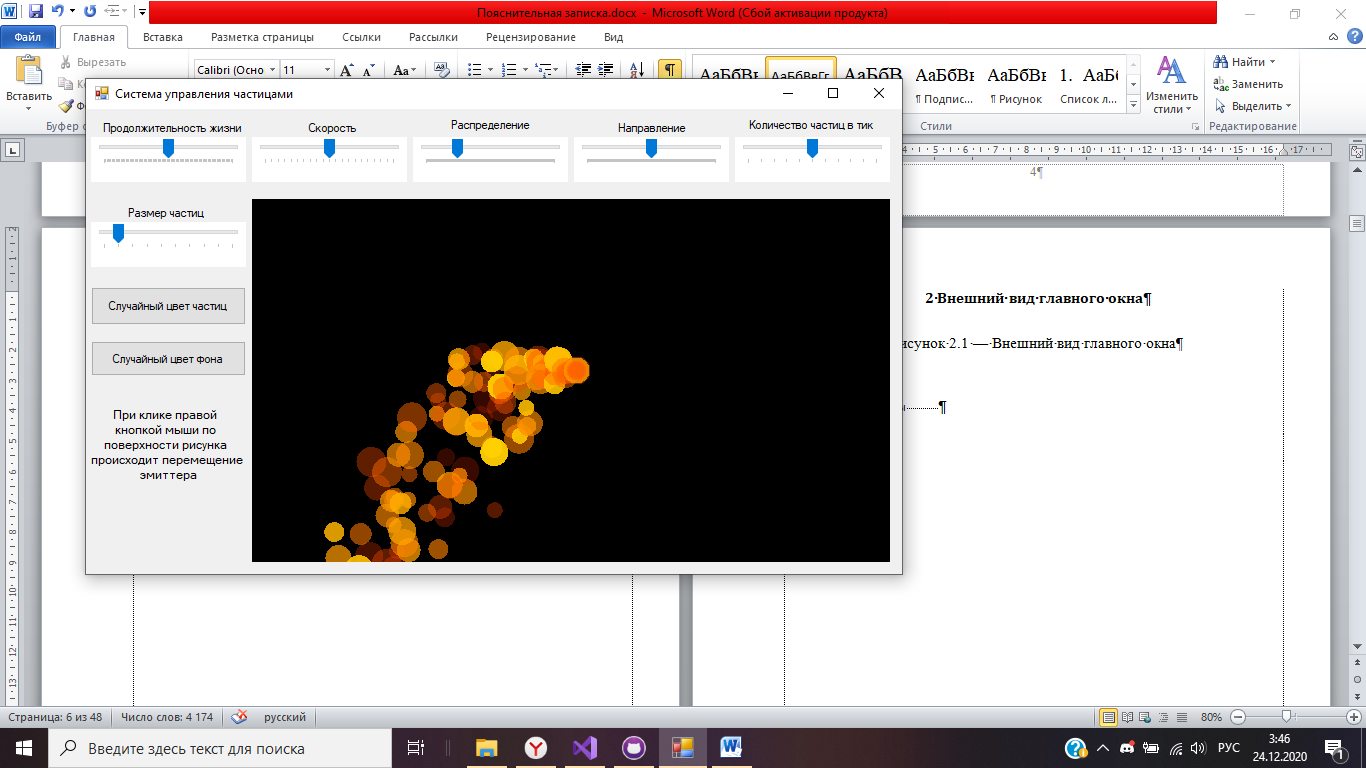


Рисунок 2.1 — Внешний вид главного окна

# 3 Код эмиттера

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace CourseWork2

{

class Emitter

{

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

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

public int Y; // соответствующая координата Y

public float GravitationX = 0;

public float GravitationY = 1; // пусть гравитация будет силой один пиксель за такт, нам хватит

public int Direction = 15; // 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 = 20; // минимальное время жизни частицы

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

public int ParticlesPerTick = 1;

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

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

public void UpdateState()

{

int particlesToCreate = ParticlesPerTick;

foreach (var particle in particles)

{

if (particle.Life <= 0) // если частицы умерла

{

if (particlesToCreate > 0)

{

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

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

ResetParticle(particle);

}

}

else

{

// это не трогаем

particle.SpeedX += GravitationX;

particle.SpeedY += GravitationY;

particle.X += particle.SpeedX;

particle.Y += particle.SpeedY;

}

}

while (particlesToCreate >= 1)

{

particlesToCreate -= 1;

var particle = CreateParticle();

ResetParticle(particle);

particles.Add(particle);

}

}

public virtual void ResetParticle(Particle particle)

{

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

particle.X = X;

particle.Y = Y;

var direction = Direction

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

- Spreading / 2;

var 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);

}

/\* добавил метод \*/

public virtual Particle CreateParticle()

{

var particle = new Particle();

particle.FromColor = ColorFrom;

particle.ToColor = ColorTo;

return particle;

}

// функция рендеринга

public void Render(Graphics g)

{

// утащили сюда отрисовку частиц

foreach (var particle in particles)

{

particle.Draw(g);

}

}

}

}

# 4 Код частиц

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace CourseWork2

{

class Particle

{

public int Radius; // радиус частицы

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

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

public float SpeedX; // скорость перемещения по оси X

public float SpeedY;

public float Life; // запас здоровья частицы

public static Random rand = new Random();

public Color FromColor;

public Color ToColor;

public Particle()

{

// генерируем произвольное направление и скорость

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

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

// рассчитываем вектор скорости

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

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

// а это не трогаем

Radius = 2 + rand.Next(10);

Life = 20 + rand.Next(100);

}

// для смеси 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))

);

}

// ну и отрисовку перепишем

public void Draw(Graphics g)

{

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

// так как k уменьшается от 1 до 0, то порядок цветов обратный

var color = MixColor(ToColor, FromColor, 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 CourseWork2

{

public partial class Form1 : Form

{

Color picColor = Color.Black;

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

Emitter emitter; // добавим поле для эмиттера

public Form1()

{

InitializeComponent();

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

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

{

Direction = trackBar2.Value,

Spreading = trackBar3.Value,

SpeedMin = trackBar5.Value,

SpeedMax = trackBar5.Value+3,

ColorFrom = Color.Gold,

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

ParticlesPerTick = trackBar1.Value,

X = picDisplay.Width / 2,

Y = picDisplay.Height / 2,

RadiusMin = trackBar6.Value \* 4,

RadiusMax = trackBar6.Value \* 4+8,

};

emitters.Add(this.emitter); // все равно добавляю в список emitters, чтобы он рендерился и обновлялся

}

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

{

emitter.UpdateState(); // тут теперь обновляем эмиттер

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

{

g.Clear(picColor);

emitter.Render(g); // а тут теперь рендерим через эмиттер

}

picDisplay.Invalidate();

}

private void button1\_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 = picColor;

}

private void button2\_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);

picColor = Color.FromArgb(R, G, B);

emitter.ColorTo = picColor;

}

private void trackBar1\_Scroll(object sender, EventArgs e)

{

emitter.ParticlesPerTick = trackBar1.Value;

}

private void trackBar2\_Scroll(object sender, EventArgs e)

{

emitter.Direction = trackBar2.Value;

}

private void trackBar3\_Scroll(object sender, EventArgs e)

{

emitter.Spreading = trackBar3.Value;

}

private void trackBar5\_Scroll(object sender, EventArgs e)

{

if (trackBar5.Value==0) {

emitter.SpeedMin = trackBar5.Value;

emitter.SpeedMax = trackBar5.Value ;

}

else {

emitter.SpeedMin = trackBar5.Value;

emitter.SpeedMax= trackBar5.Value+3;

}

}

private void trackBar4\_Scroll(object sender, EventArgs e)

{

if (trackBar4.Value==0) {

emitter.LifeMax = 0;

emitter.LifeMin = 0;

}

else if (trackBar4.Value>70) {

emitter.LifeMax = 100;

emitter.LifeMin = trackBar4.Value;

}

else {

emitter.LifeMax = trackBar4.Value+30;

emitter.LifeMin = trackBar4.Value;

}

}

private void trackBar6\_Scroll(object sender, EventArgs e)

{

if (trackBar6.Value == 0)

{

emitter.RadiusMax = trackBar6.Value;

emitter.RadiusMin = trackBar6.Value;

}

else

{

emitter.RadiusMax = trackBar6.Value\*4+8;

emitter.RadiusMin = trackBar6.Value\*4;

}

}

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

{

emitter.X = e.X;

emitter.Y = e.Y;

}

}

}

# 7 Описание работы интерфейса

Нажимая button1 и button2 **Случайный цвет частиц и Случайный цвет фона** мы меняем цвет фона и цвет новых частицы:

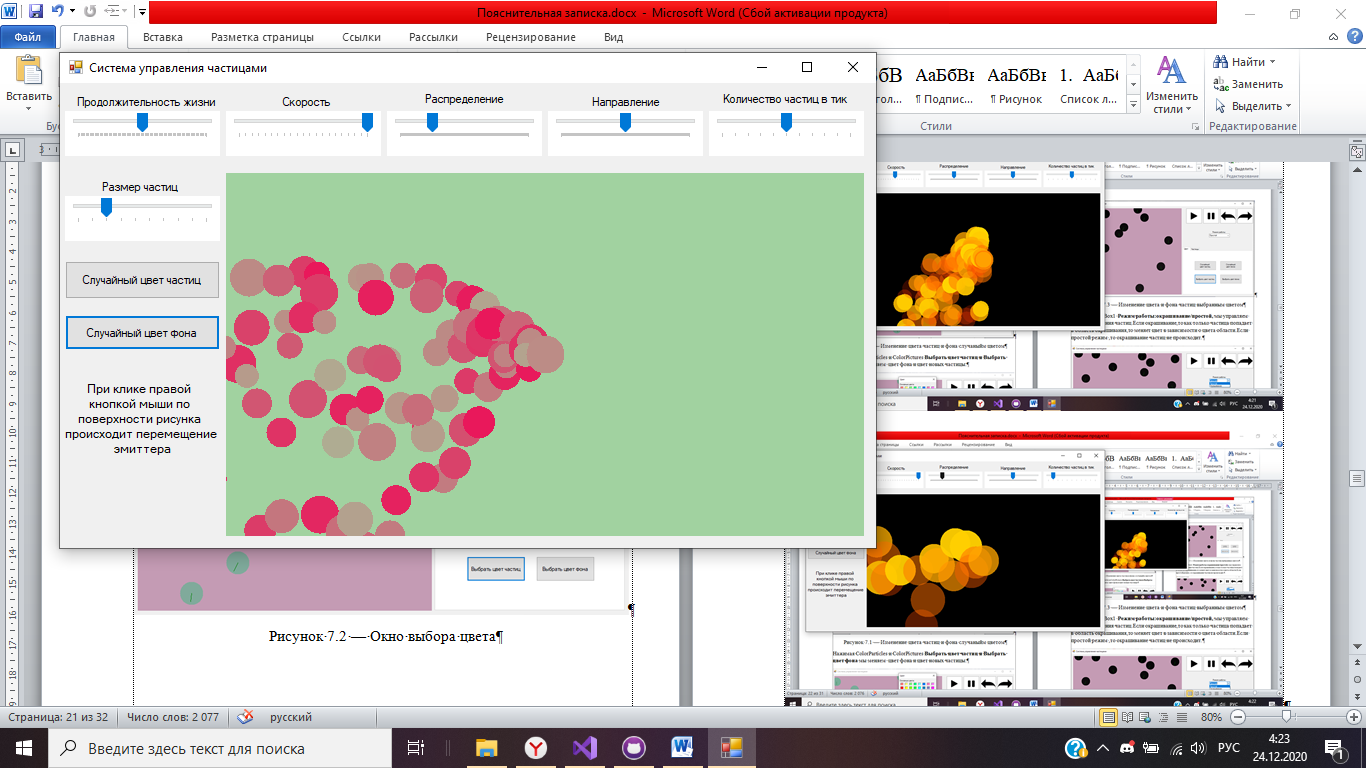
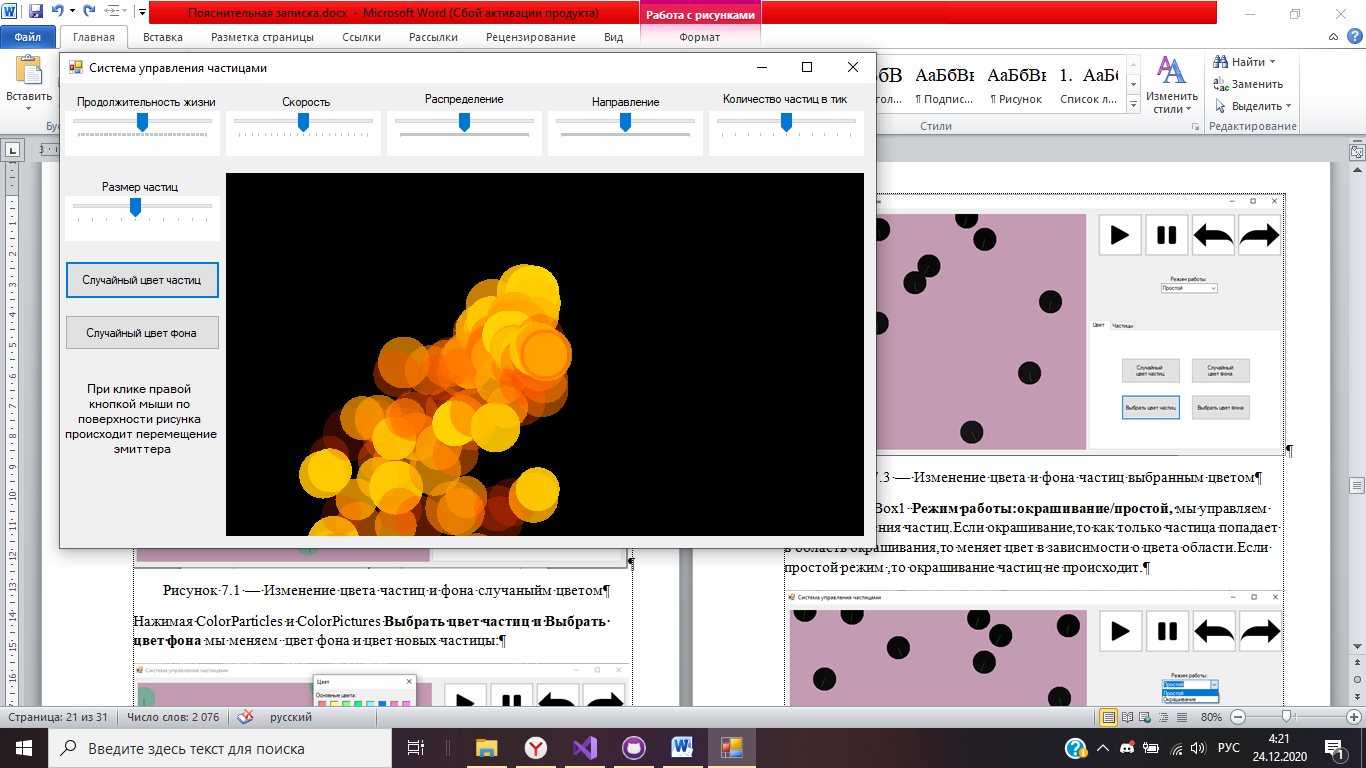


Рисунок 7.1 — Изменение цвета частиц и фона случаныйм цветом

Двигая ползунок trackBar6 **Размер частиц** изменяется размер частиц

Рисунок 7.2 — До изменения размера частиц

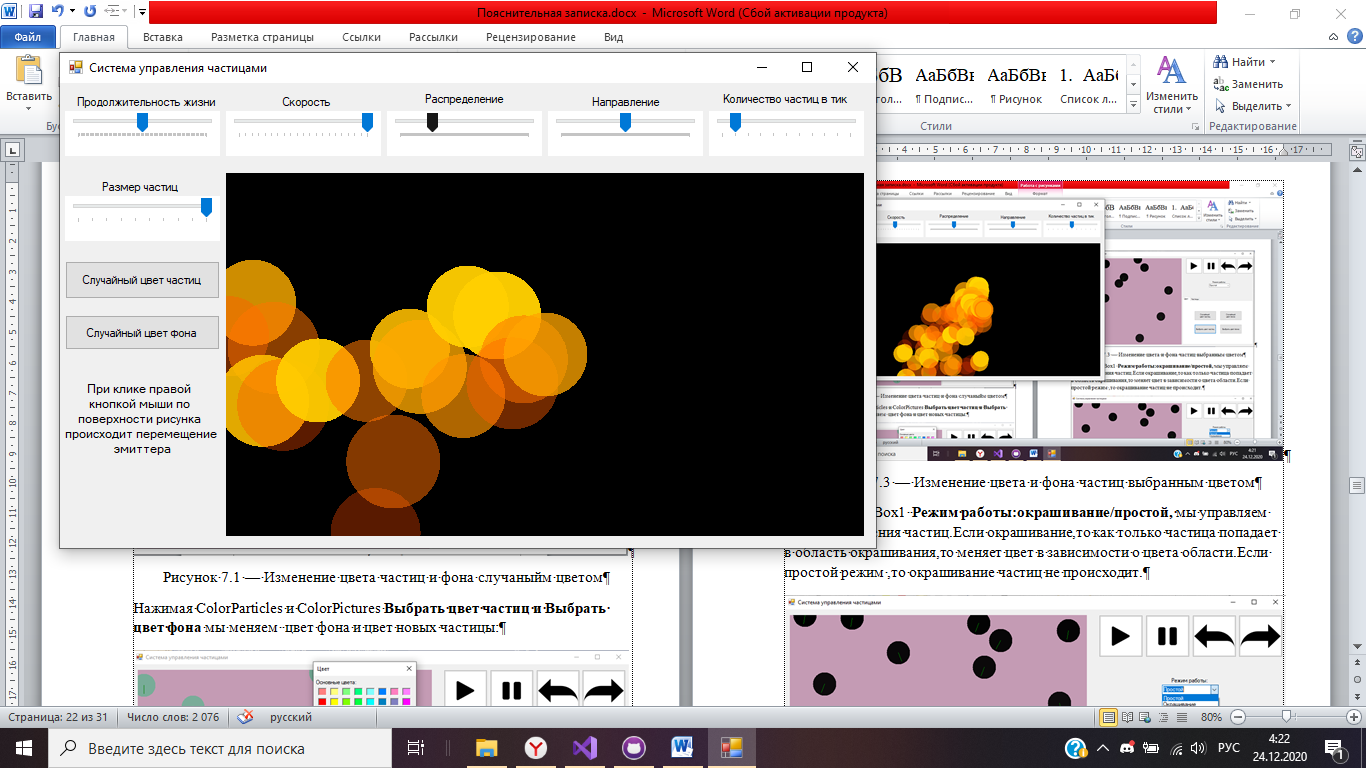
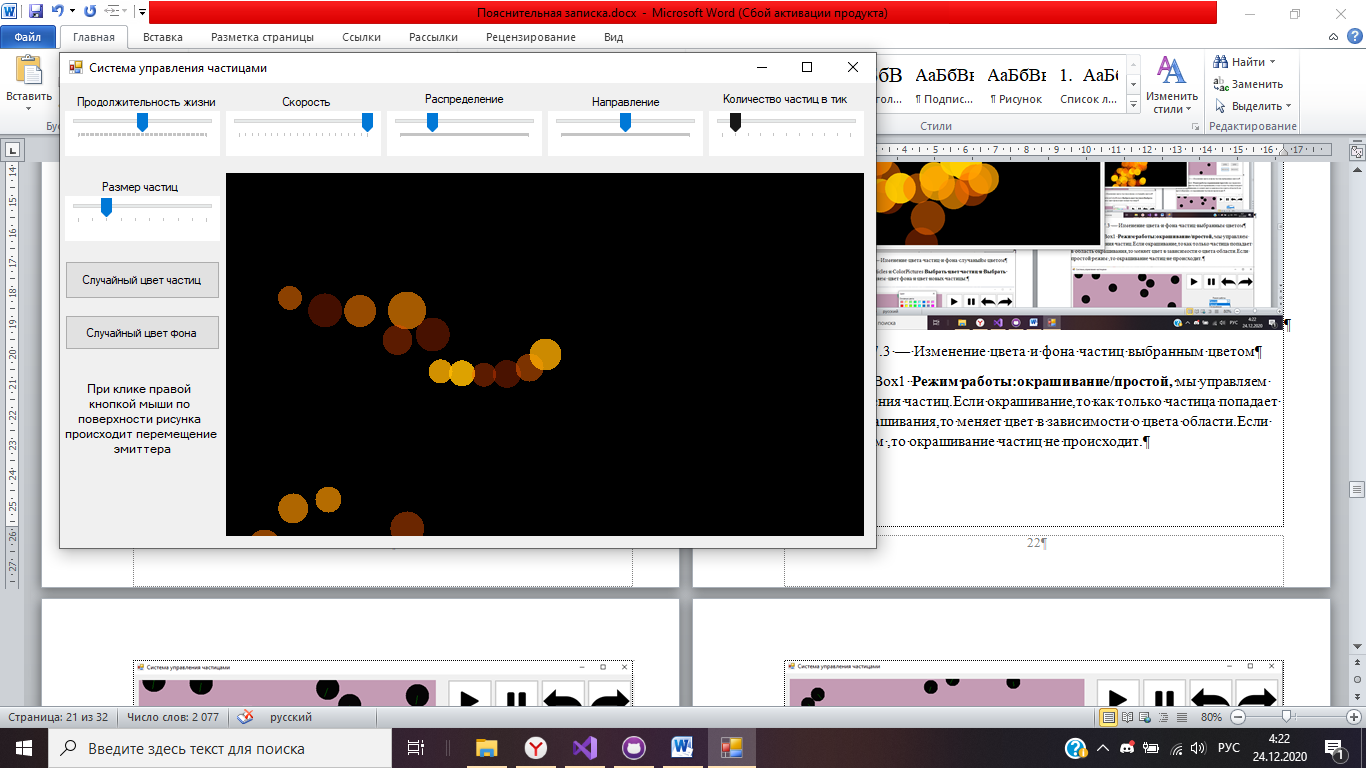


Рисунок 7.3 — После изменения размера частиц

Двигая ползунок trackBar1 **Количество частиц** изменяется количество частиц.

 Рисунок 7.4 — Небольшое количество частиц

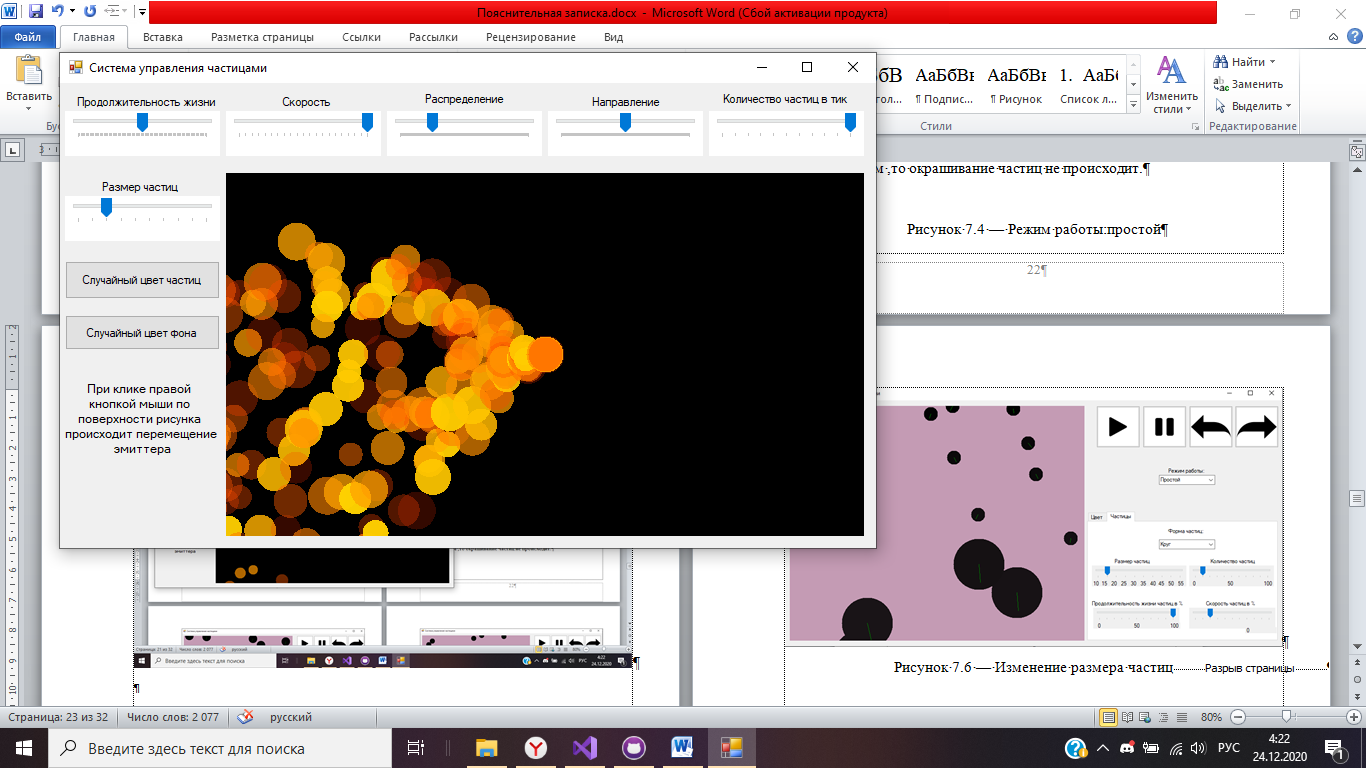


Рисунок 7.5 — Большое количество частиц

Двигая ползунок trackBar2 **Направление** изменяется направление вылетающих частиц .

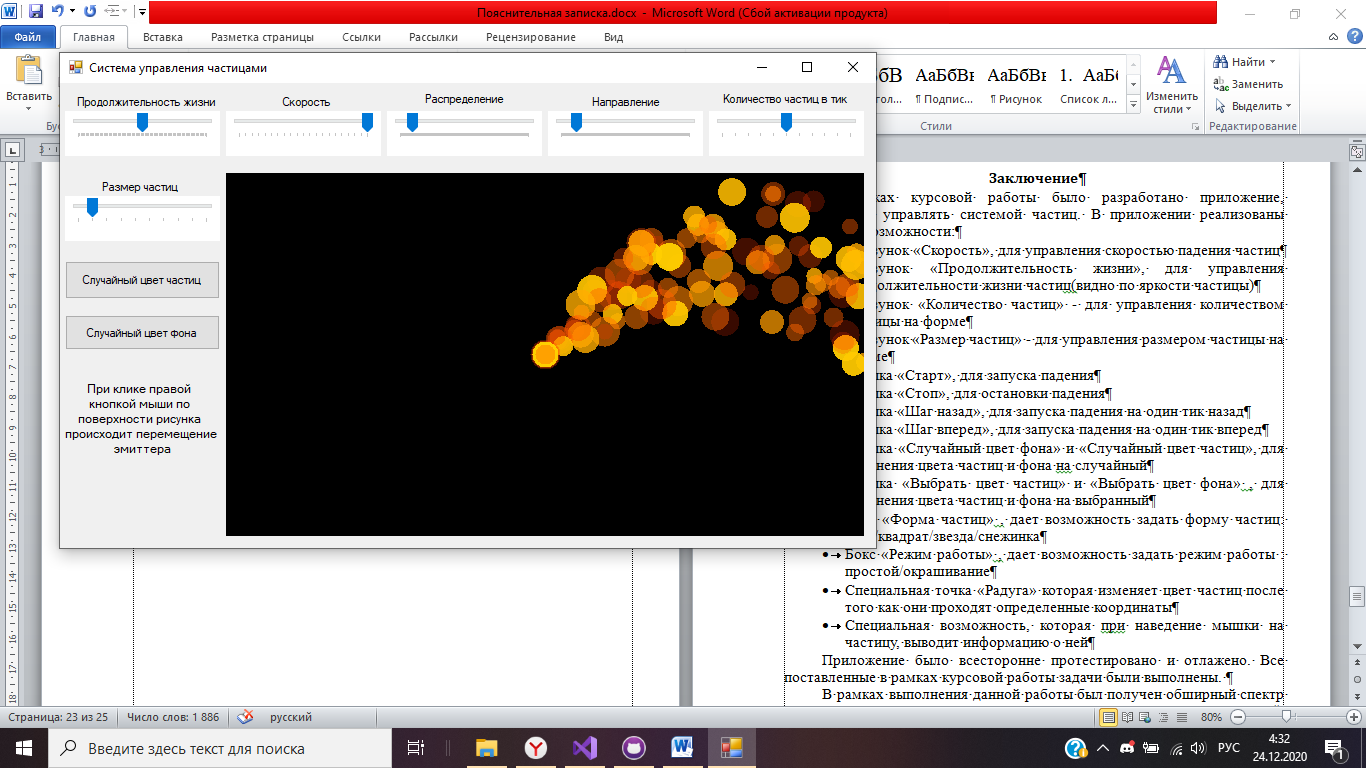


Рисунок 7.6 — Направление с одним значением

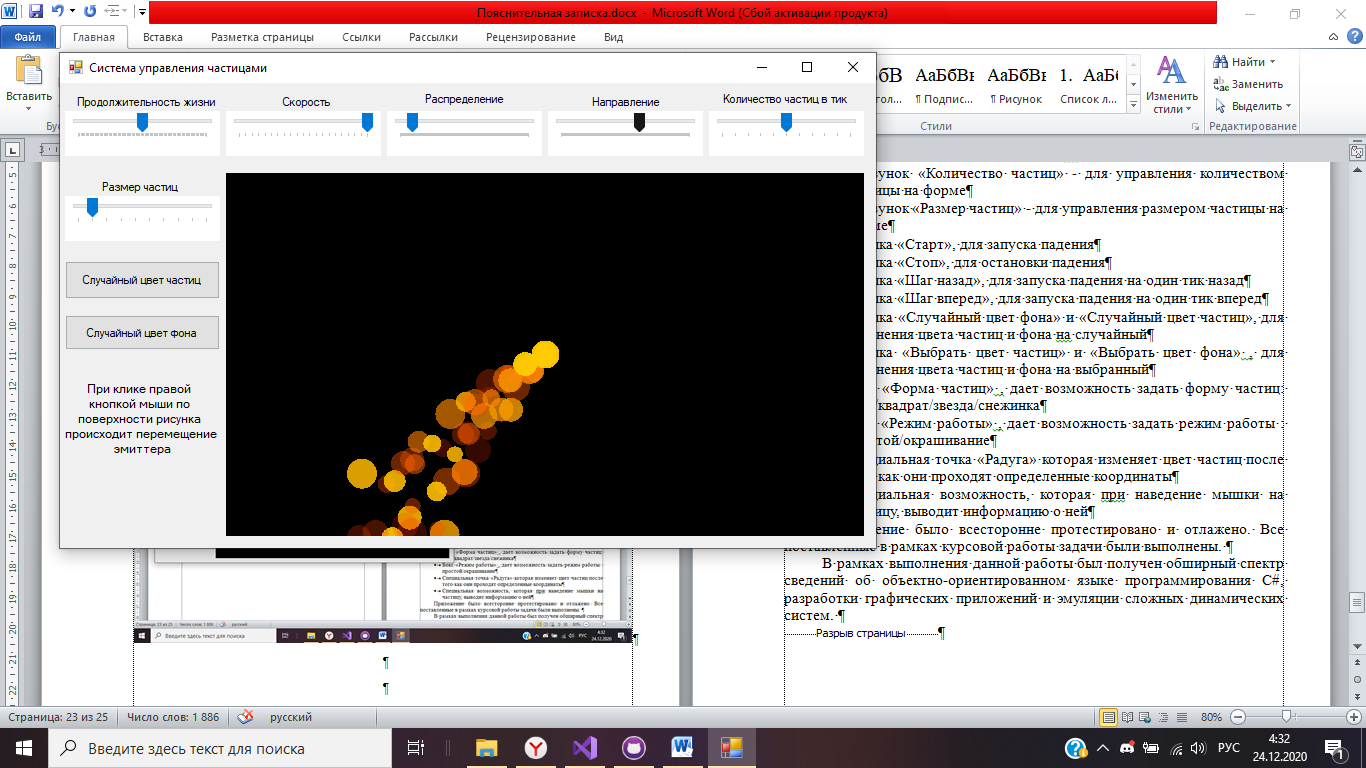


Рисунок 7.7 — Направление с другим значением

Двигая ползунок trackBar3 **Распределение** изменяется распределение в градусах вылетающих частиц .

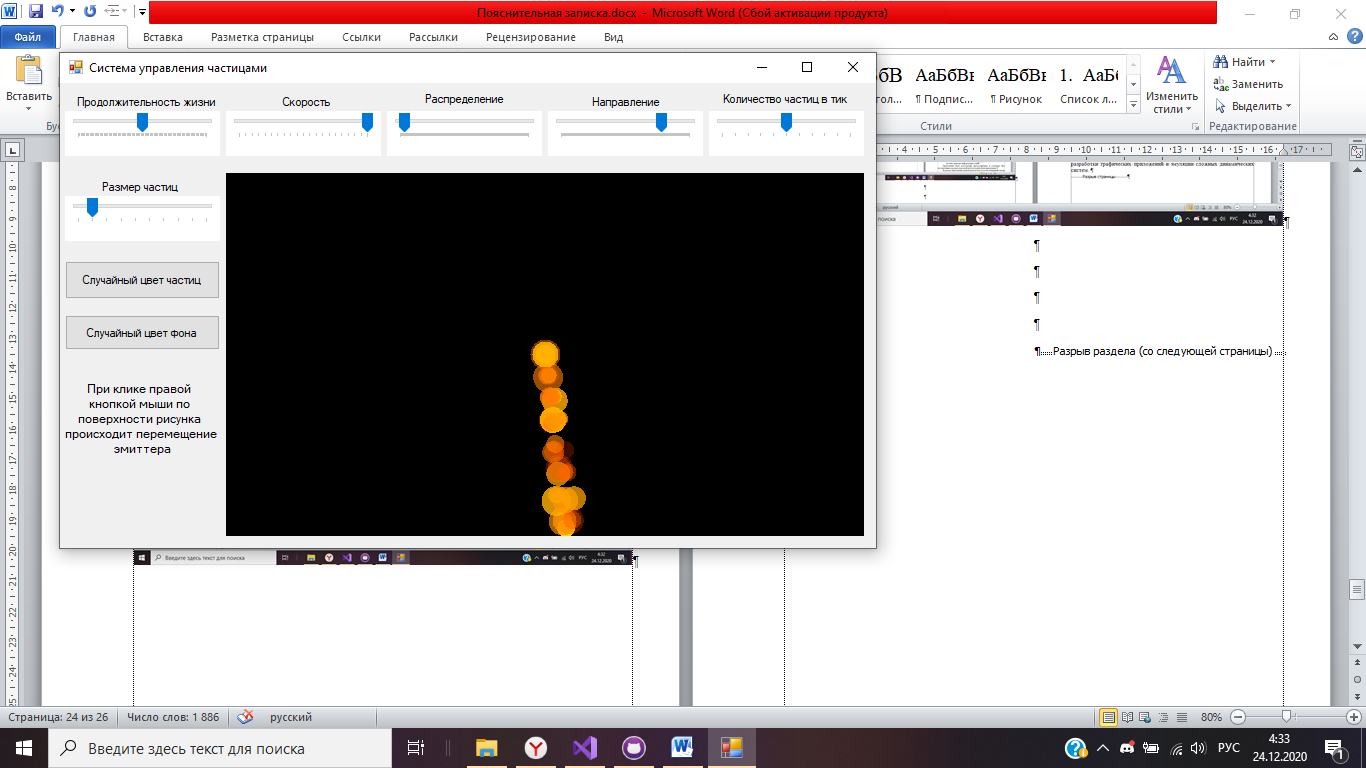


Рисунок 7.8 — Распределением с маленьким градусом

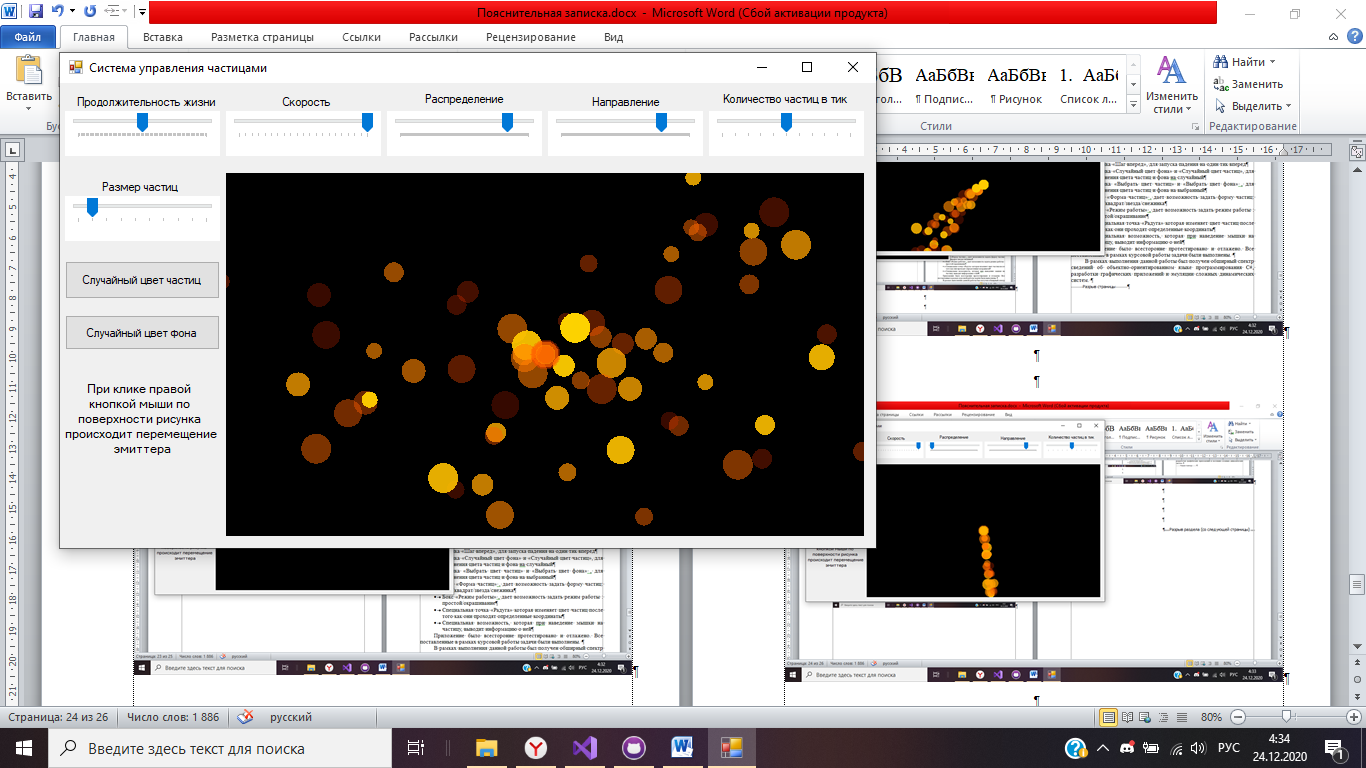


Рисунок 7.9 — Распределением с большим градусом

Двигая ползунок trackBar5 **Скорость** изменяется скорость вылетающих частиц.Если скорость маленькая,то частицы просто падают вниз.

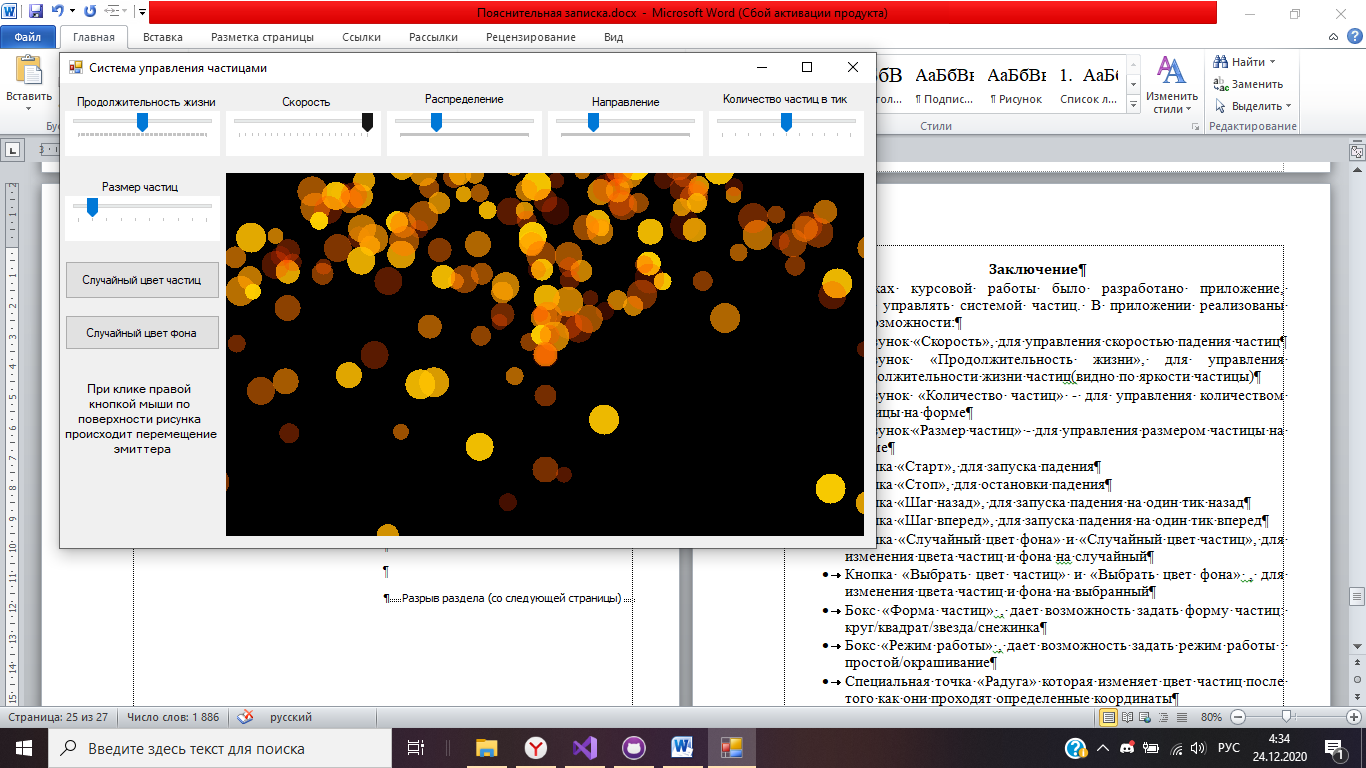


Рисунок 7.10 — Большое значение скорости

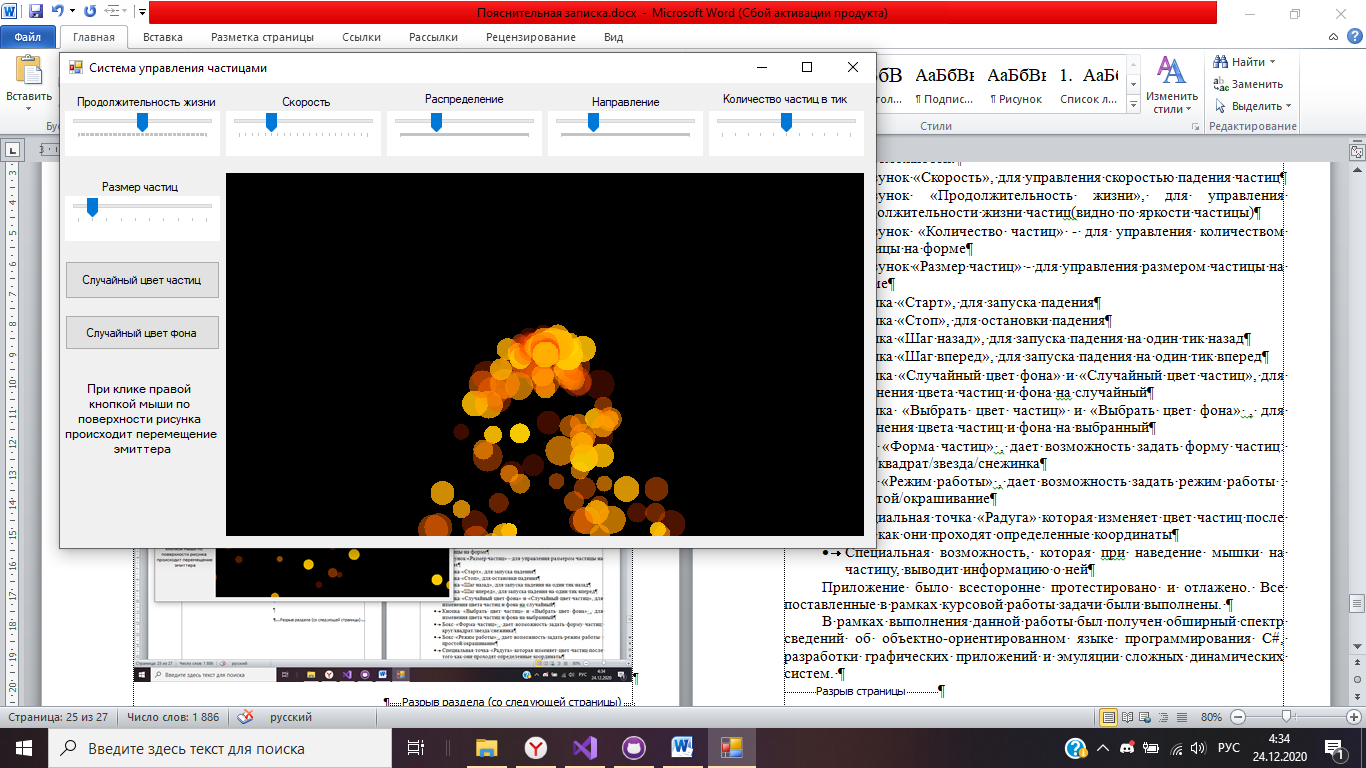


Рисунок 7.11— Маленькое значение скорости

Двигая ползунок trackBar4 **Продолжительность жизни** изменяется продолжительность жизни частиц.Наглдяно показывается в виде прозрачности частицы.

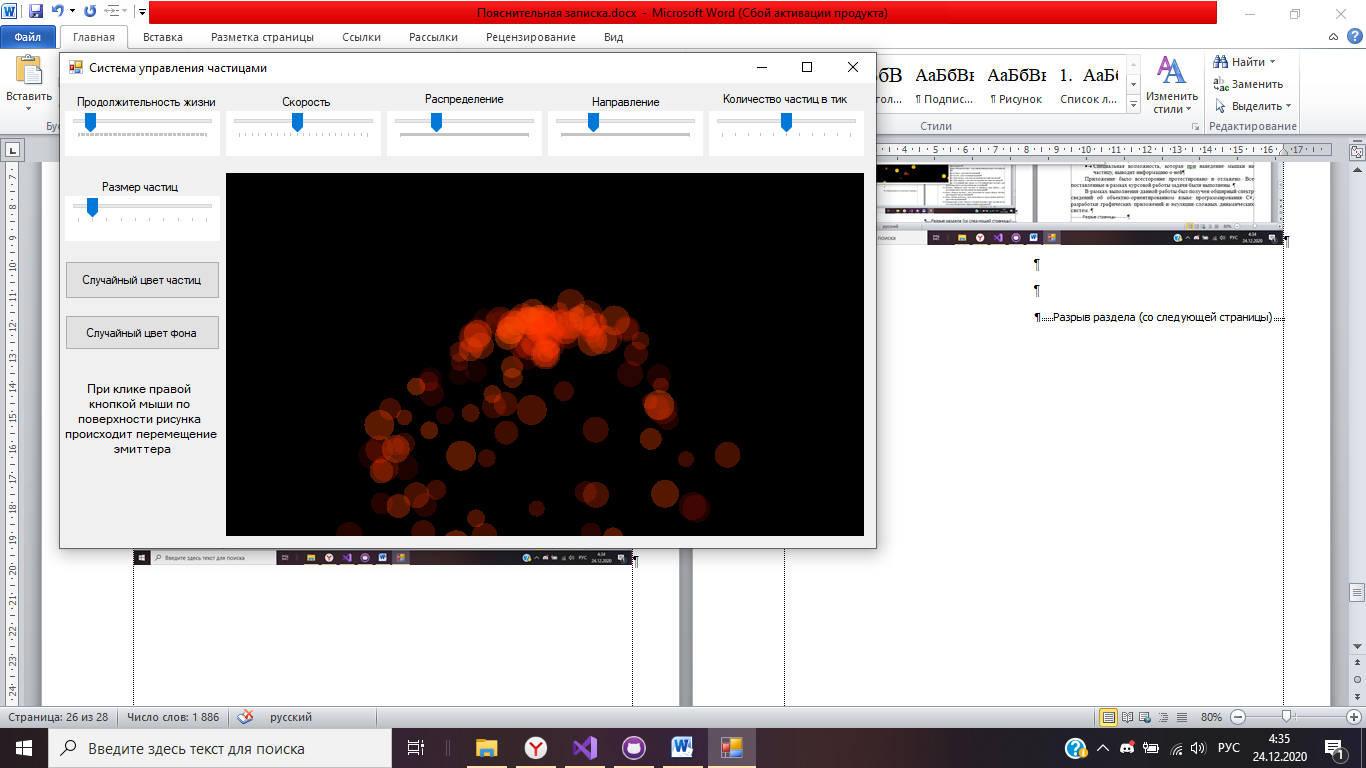


Рисунок 7.12 — Маленькая продолжительность жизни

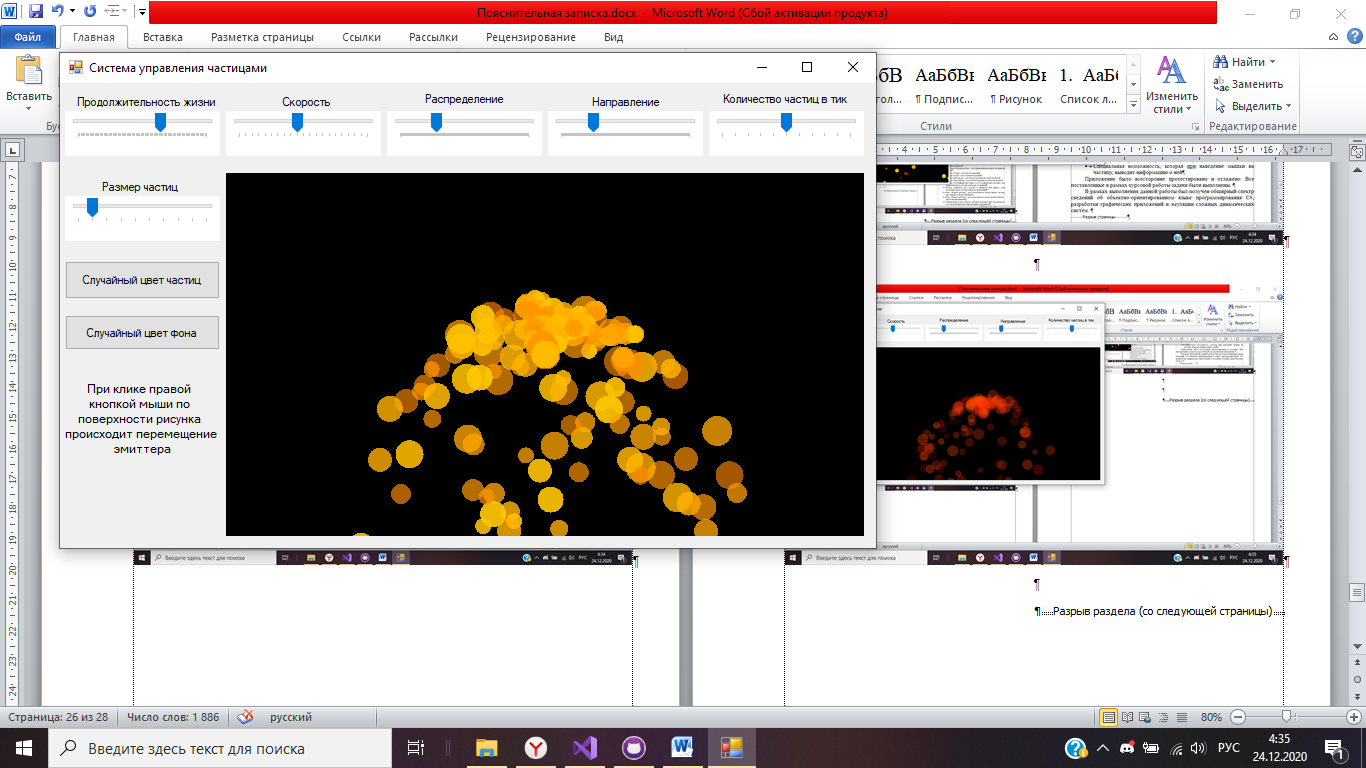


Рисунок 7.13 — Большая продолжительность жизни

Нажимая на picDisplay левой кнопкой мыши положение источника частиц(емиттера) изменяется

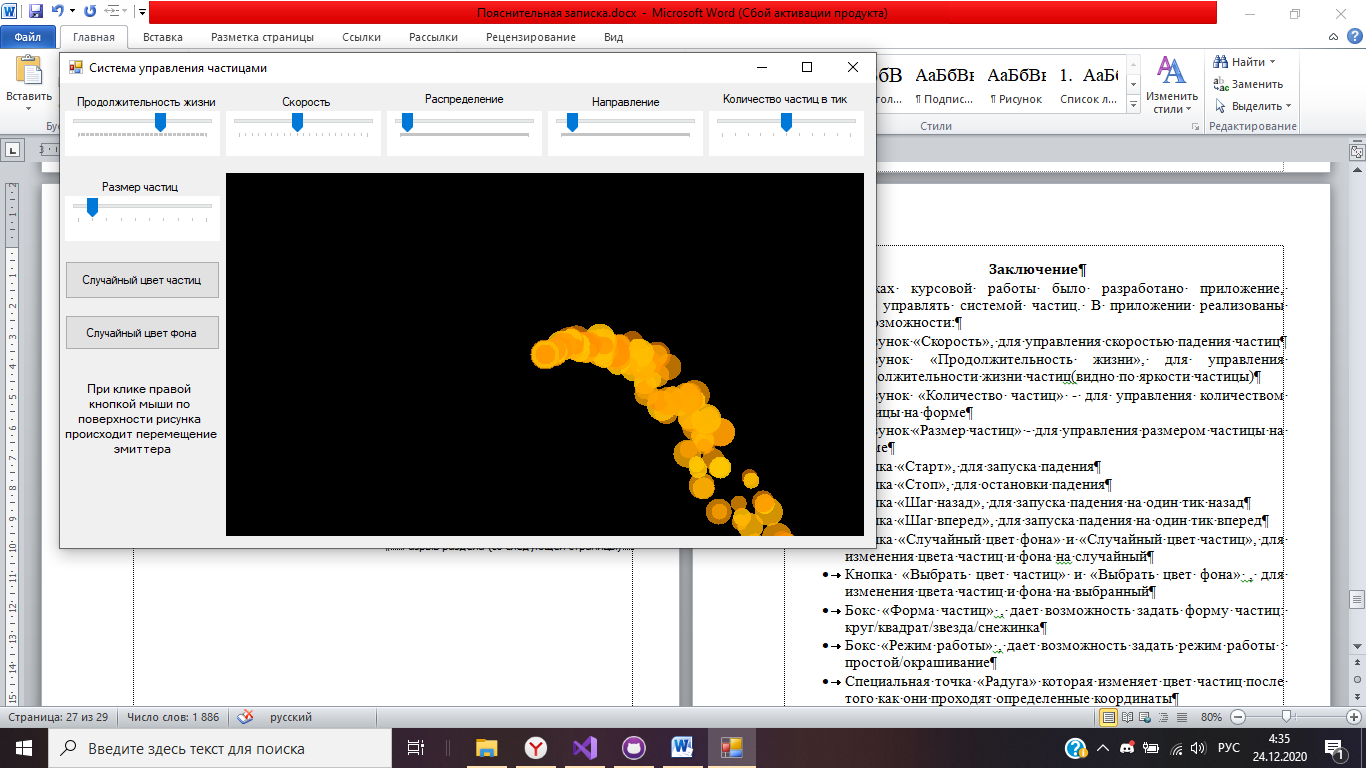


Рисунок 7.14 — До смены положения источника частиц

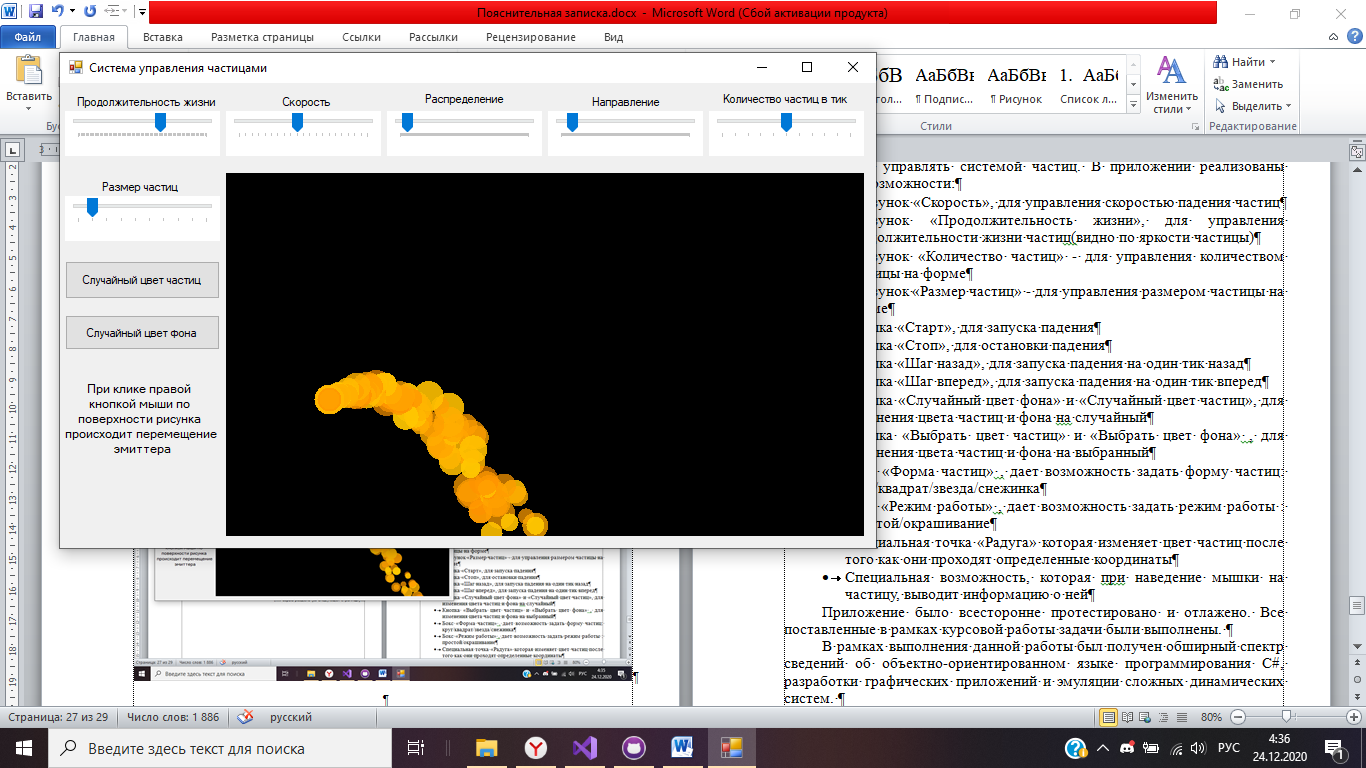


Рисунок 7.15 — После смены положения источника частиц

# Заключение

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

* Ползунок «Скорость», для управления скоростью вылетания частиц из источника
* Ползунок «Продолжительность жизни», для управления продолжительности жизни частиц(видно по яркости частицы)
* Ползунок «Количество частиц» - для управления количеством вылетающих частиц
* Ползунок «Размер частиц» - для управления размером вылетающих из источника частиц
* Кнопка «Случайный цвет фона» и «Случайный цвет частиц», для изменения цвета частиц и фона на случайный
* Ползунок «Распределение» - для управления распределения частиц в градусах
* Ползунок «Направление» - для управления направления вылетающих из источника частиц
* Перемещение емиттера(источника) по клику мыши

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).