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 | ИС\*\*\* 19-1 |  |  |  | Иванов И.И. |
|  | 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 | Иванову Ивану Ивановичу | |
|  | (фамилия, инициалы) | |
| Тема работы: | Технология моделирования системы частиц | | |
| 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] // 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 | | | |  | |  | Иванов И.И. |
|  | | | | 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 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/распределения, а также количества частиц в тик для эмиттера,

1. Дополнительно реализовать гравитон который притягивает частицы оказавшиеся в радиусе действия частиц

# 2 Внешний вид главного окна

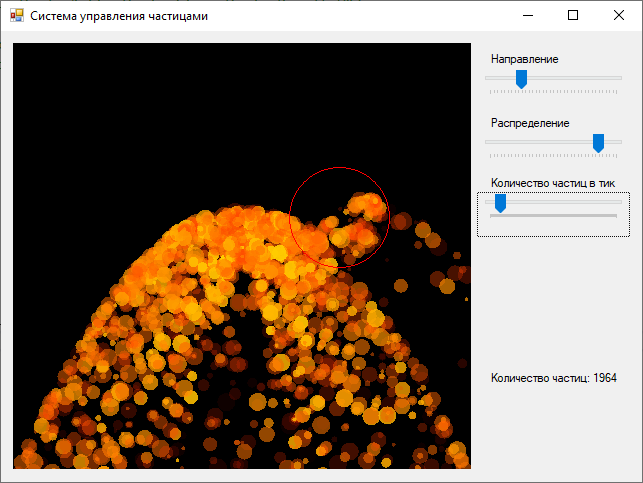


Рисунок 2.1

# 3 Код эмиттера

*обязательно добавьте комментарии к коду если он отличается от того что был в методичке*

public class Emitter

{

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

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

public int MousePositionX;

public int MousePositionY;

public float GravitationX = 0;

public float GravitationY = 1;

public int X;

public int Y;

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 int ParticlesPerTick = 1;

public Color ColorFrom = Color.White;

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

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;

if (particle is ParticleColorful)

{

(particle as ParticleColorful).FromColor = ColorFrom;

(particle as ParticleColorful).ToColor = ColorTo;

}

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);

}

public virtual Particle CreateParticle()

{

var particle = new ParticleColorful();

particle.FromColor = ColorFrom;

particle.ToColor = ColorTo;

return particle;

}

public void UpdateState()

{

int particlesToCreate = ParticlesPerTick;

foreach (var particle in particles)

{

if (particle.Life <= 0)

{

if (particlesToCreate > 0)

{

particlesToCreate -= 1;

ResetParticle(particle);

}

}

else

{

particle.Life -= 1;

foreach (var point in impactPoints)

{

point.ImpactParticle(particle);

}

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 void Render(Graphics g)

{

foreach (var particle in particles)

{

particle.Draw(g);

}

foreach (var point in impactPoints)

{

point.Render(g);

}

}

}

# 4 Код специальных точек

*обязательно добавьте комментарии к коду если он отличается от того что был в методичке*

public abstract class IImpactPoint

{

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

);

}

}

/\*

\* класс Гравитон -- притягивает частицы оказавшиеся в радиусе его действия,

\* радиус действия равен половине значения Power

\*/

public class GravityPoint : IImpactPoint

{

public int Power = 100; // сила притяжения

public override void ImpactParticle(Particle particle)

{

float gX = X - particle.X;

float gY = Y - particle.Y;

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

// проверяю что частица оказалась в радиусе притяжения

if (r + particle.Radius < Power / 2)

{

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

// пересчитываю скорость частицы

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

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

}

}

public override void Render(Graphics g)

{

// буду рисовать оружность с диаметром равным Power

g.DrawEllipse(

new Pen(Color.Red),

X - Power / 2,

Y - Power / 2,

Power,

Power

);

}

}

# 5 Код частиц

*обязательно добавьте комментарии к коду если он отличается от того что был в методичке*

public 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; // скорость перемещения по оси Y

public float Life; // запас здоровья частицы

public static Random rand = new Random();

// метод генерации частицы

public Particle()

{

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

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

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

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

Radius = 2 + rand.Next(10);

Life = 20 + rand.Next(100);

}

virtual public void Draw(Graphics g)

{

// рассчитываем коэффициент прозрачности по шкале от 0 до 1.0

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

// рассчитываем значение альфа канала в шкале от 0 до 255

// по аналогии с RGB, он используется для задания прозрачности

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

// создаем цвет из уже существующего, но привязываем к нему еще и значение альфа канала

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

var b = new SolidBrush(color);

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

b.Dispose();

}

}

public class ParticleColorful : Particle

{

public Color FromColor;

public Color ToColor;

// 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))

);

}

public override void Draw(Graphics g)

{

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

// так как k уменшается 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

*обязательно добавьте комментарии к коду если он отличается от того что был в методичке*

public partial class Form1 : Form

{

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

Emitter emitter;

GravityPoint point;

public Form1()

{

InitializeComponent();

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

emitter = new Emitter

{

Direction = 0,

Spreading = 10,

SpeedMin = 10,

SpeedMax = 10,

ColorFrom = Color.Gold,

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

ParticlesPerTick = 10,

X = picDisplay.Width / 2,

Y = picDisplay.Height / 2,

};

point = new GravityPoint

{

X = picDisplay.Width / 2 - 100,

Y = picDisplay.Height / 2,

};

emitter.impactPoints.Add(point);

emitters.Add(emitter);

}

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

{

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

{

g.Clear(Color.Black);

foreach (var emitter in emitters)

{

emitter.UpdateState(); // тут обновляем эмиттер

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

/\* тут какая-та магия, а у вас свой код \*/

}

}

picDisplay.Invalidate();

}

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

{

point.X = e.X;

point.Y = e.Y;

}

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

{

/\* а тут плохо видно, не могу разглядеть что -\_- \*/

}

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

{

/\* ой-ой -\_- \*/

}

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

{

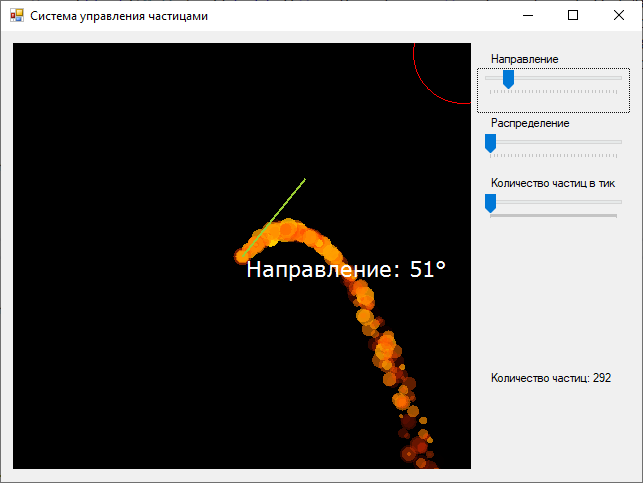
/\* надо новые очки О-О \*/

}

}

# 7 Описание работы интерфейса

Меняя trackback **Направление** мы меняем направление в которое выстреливают новые частицы:

  
Рисунок 7.1

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

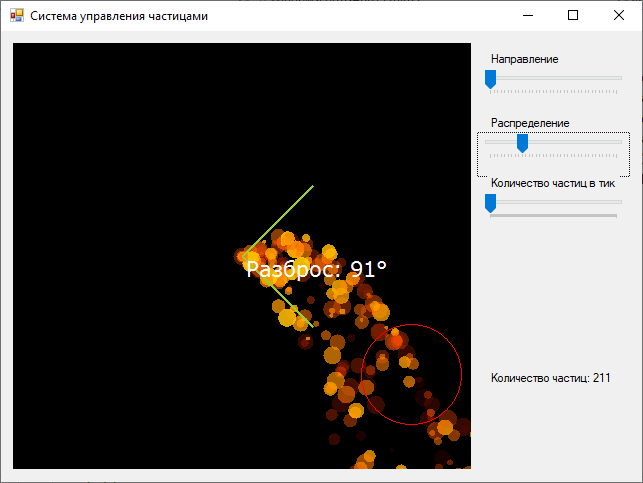


Рисунок 7.2

Меняя trackbar **Количество частиц в тик,** мы управляем количеством активных частиц в области генерации, при резком изменении частиц в боьлшую сторону и с последюущем:

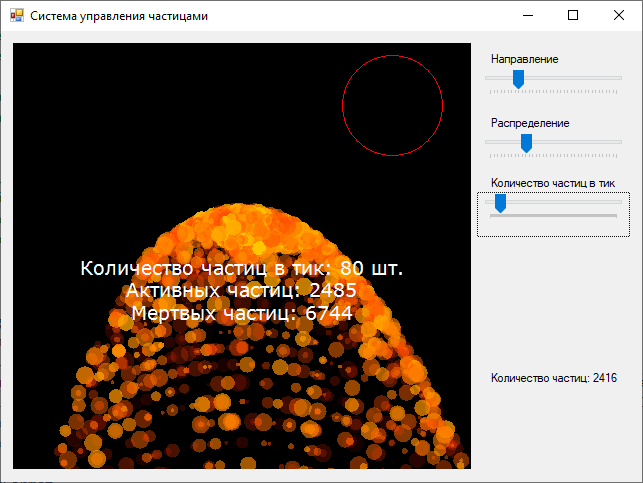


Рисунок 7.3

Частицы попадающие в область действия специальной точки притягиваются к ней с силой равной удвоеному радиусу:

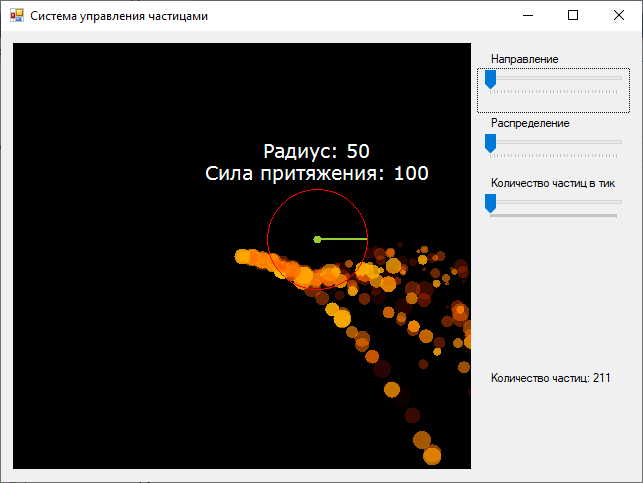


Рисунок 7.4

# Заключение

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

* Ползунок «Направление», для управления направлением вылета частиц
* Ползунок «Распределение», для управления разбросом вылета частиц около заданного направления
* Ползунок «Количество частиц в тик» - для управления количеством частицы на форме,
* Специальная точка «Гравитон[Придумай имя своей точке, ведь все равно это никто не читает]» которая притягивает частицы оказывающиеся в заданной области

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).