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:  Бахвалова З.А. |
| 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-20-3 |  |  |  | Пипия К.Т. |
|  | шифр группы |  | подпись |  | Фамилия И.О |
| Нормоконтроль |  |  |  |  | Бахвалова З.А. |
|  |  |  | 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 | Пипия Кобе Тенгизиевичу | |
|  | (фамилия, инициалы) | |
| Тема работы: | Разработка системы управления частицами на языке C# | | |
| 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] // aqua.tealeaf.su : чаинка, 2021. URL: http://aqua.tealeaf.su/particle-system.html (дата обращения: 01.12.2021). | | | |
|  | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Графическая часть на | 23 | | листах. | |
| Дата выдачи задания |  | 23 / 11 / 2021 г. | | |
| Задание получил студент | | | |  | |  | Пипия К.Т. |
|  | | | | подпись | |  | Фамилия И.О. |

|  |  |
| --- | --- |
| Дата представления работы руководителю | 27 / 12 / 2021 г. |
| Руководитель курсовой работы |  |  | Бахвалова З.А. |
|  | подпись | Фамилия И.О. |

Содержание

[Введение 4](#_Toc91470503)

[1 Индивидуальный вариант 5](#_Toc91470504)

[2 Внешний вид главного окна 6](#_Toc91470505)

[3 Код эмиттера 7](#_Toc91470506)

[4 Код специальных точек 10](#_Toc91470507)

[5 Код частиц 13](#_Toc91470508)

[6 Код формы 14](#_Toc91470509)

[7 Описание работы интерфейса 17](#_Toc91470510)

[Заключение 22](#_Toc91470511)

[Список использованной литературы 23](#_Toc91470512)

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

* Реализовать раскрашивание, например, снега падающего с неба
* Добавить trackbar, с помощью которого можно перемещать точки перекрашивания

# 2 Внешний вид главного окна

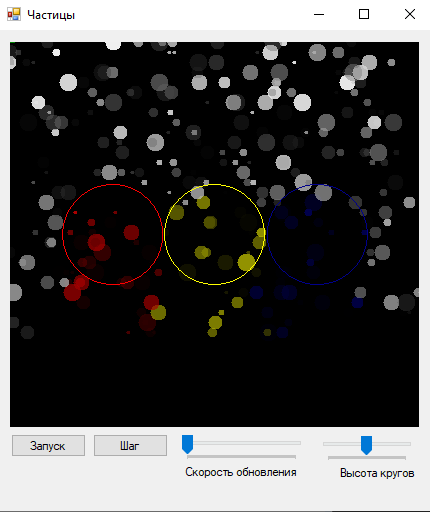


Рисунок 2.1

# 3 Код эмиттера

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

namespace ParticleSystem

{

public class Emitter

{

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 ParticlesCount = 500;

public bool a = true;

public Color ColorFrom = Color.White;

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

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

public int MousePositionX;

public int MousePositionY;

public float GravitationX = 0;

public float GravitationY = 1;

public int ParticlesPerTick = 1;

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

public virtual ParticleColorful CreateParticle()

{

var particle = new ParticleColorful();

particle.FromColor = ColorFrom;

particle.ToColor = ColorTo;

return particle;

}

public virtual void ResetParticle(ParticleColorful particle)

{

particle.FromColor = Color.White;

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);

}

public void UpdateIm()

{

foreach (var particle in particles)

{

foreach (var point in impactPoints)

{

point.ImpactParticle(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);

}

}

}

public class TopEmitter : Emitter

{

public int Width; // длина экрана

public override void ResetParticle(ParticleColorful particle)

{

base.ResetParticle(particle);

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

particle.Y = 0;

particle.SpeedY = 10;

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

}

}

}

# 4 Код специальных точек

using System;

using System.Collections.Generic;

using System.Drawing;

namespace ParticleSystem

{

public abstract class IImpactPoint

{

public float X;

public float Y;

public abstract void ImpactParticle(ParticleColorful particle);

public virtual void Render(Graphics g)

{

g.FillEllipse(

new SolidBrush(Color.Red),

X - 5,

Y - 5,

10,

10

);

}

}

public class GravityPoint : IImpactPoint

{

public int Power = 100;

public override void ImpactParticle(ParticleColorful particle)

{

float gX = X - particle.X;

float gY = Y - particle.Y;

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

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

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.DrawEllipse(

new Pen(Color.Red),

X - Power / 2,

Y - Power / 2,

Power,

Power

);

var stringFormat = new StringFormat();

stringFormat.Alignment = StringAlignment.Center; stringFormat.LineAlignment = StringAlignment.Center;

var text = $"Я гравитон\nc силой {Power}";

var font = new Font("Verdana", 10);

var size = g.MeasureString(text, font);

g.FillRectangle(

new SolidBrush(Color.Red),

X - size.Width / 2,

Y - size.Height / 2,

size.Width,

size.Height

);

g.DrawString(

text,

font,

new SolidBrush(Color.White),

X,

Y,

stringFormat

);

}

}

public class AntiGravityPoint : IImpactPoint

{

public int Power = 100;

public override void ImpactParticle(ParticleColorful 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; // тут минусики вместо плюсов

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

}

}

public class ColorPoint : IImpactPoint

{

public Color color;

public override void Render(Graphics g)

{

g.DrawEllipse(

new Pen(color),

X - 50,

Y - 50,

100,

100

);

}

public override void ImpactParticle(ParticleColorful particle)

{

float gX = X - particle.X;

float gY = Y - particle.Y;

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

if (r<50)

{

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

particle.FromColor = color;

}

else

{

}

}

}

public class InfoPoint : IImpactPoint

{

public Action<ParticleColorful> OnParticle;

public override void Render(Graphics g)

{

g.DrawEllipse(

new Pen(Color.Yellow),

X - 0.5f,

Y - 0.5f,

1,

1

);

}

public override void ImpactParticle(ParticleColorful particle)

{

float gX = X - particle.X;

float gY = Y - particle.Y;

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

if (r < particle.Radius\*2)

{

OnParticle(particle);

}

else

{

}

}

}

}

# 5 Код частиц

using System;

using System.Drawing;

namespace ParticleSystem

{

public class Particle

{

public int Radius;

public float X;

public float Y;

public float SpeedX;

public float SpeedY;

public float Life;

public static Random rand = new Random();

public Particle()

{

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);

Radius = 2 + rand.Next(10);

Life = 20 + rand.Next(100);

}

public void ShowInfo(Graphics g)

{

var stringFormat = new StringFormat();

stringFormat.Alignment = StringAlignment.Center; stringFormat.LineAlignment = StringAlignment.Center;

var text = $"X: {X}\nY: {Y}";

var font = new Font("Verdana", 10);

var size = g.MeasureString(text, font);

g.FillRectangle(

new SolidBrush(Color.Red),

X - size.Width / 2,

Y - size.Height / 2,

size.Width,

size.Height

);

g.DrawString(

text,

font,

new SolidBrush(Color.White),

X,

Y,

stringFormat

);

}

public virtual void Draw(Graphics g)

{

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

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

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

var b = new SolidBrush(color);

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

g.DrawLine(new Pen(Color.Black, 2), 0, 0, 5, 0);

b.Dispose();

}

}

public class ParticleColorful : Particle

{

public Color FromColor;

public Color ToColor;

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);

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

var b = new SolidBrush(color);

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

g.DrawLine(new Pen(Color.Green, 2), 0, 0, 5, 0);

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 ParticleSystem

{

public partial class Form1 : Form

{

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

Emitter emitter;

/\*

GravityPoint point1; // добавил поле под первую точку

GravityPoint point2;

\*/

InfoPoint point1;

bool a = true;

public Form1()

{

InitializeComponent();

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

emitter = new TopEmitter

{

Direction = 0,

Spreading = 10,

SpeedMin = 10,

SpeedMax = 10,

ColorFrom = Color.White,

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

ParticlesPerTick = 10,

Width = picDisplay.Width,

GravitationY = 0f

};

point1 = new InfoPoint

{

X = picDisplay.Width,

Y = picDisplay.Height,

};

point1.OnParticle += (o) =>

{

o.FromColor = Color.Red;

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

{

g.FillEllipse(new SolidBrush(Color.Red), 100, 100, 6, 6);

}

};

emitter.impactPoints.Add(new ColorPoint

{

X = (float)(picDisplay.Width \* 0.25),

Y = picDisplay.Height / 2,

color = Color.Red

});

emitter.impactPoints.Add(new ColorPoint

{

X = (float)(picDisplay.Width/2),

Y = picDisplay.Height / 2,

color = Color.Yellow

});

emitter.impactPoints.Add(new ColorPoint

{

X = (float)(picDisplay.Width \* 0.75f),

Y = picDisplay.Height / 2,

color = Color.DarkBlue

});

emitter.impactPoints.Add(point1);

}

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

{

emitter.UpdateIm();

if (a)

{

emitter.UpdateState();

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

{

g.Clear(Color.Black);

emitter.Render(g);

}

picDisplay.Invalidate();

}

}

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

{

point1.X = e.X;

point1.Y = e.Y;

}

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

{

timer1.Interval = tbSpeed.Value;

}

private void button1\_Click(object sender, EventArgs e)

{

a = !a;

//timer1.Enabled = !timer1.Enabled;

}

private void button2\_Click(object sender, EventArgs e)

{

emitter.UpdateState();

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

{

g.Clear(Color.Black);

emitter.Render(g);

}

picDisplay.Invalidate();

}

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

{

foreach (var p in emitter.impactPoints)

{

if (p is ColorPoint)

{

(p as ColorPoint).Y = trackBar1.Value;

}

}

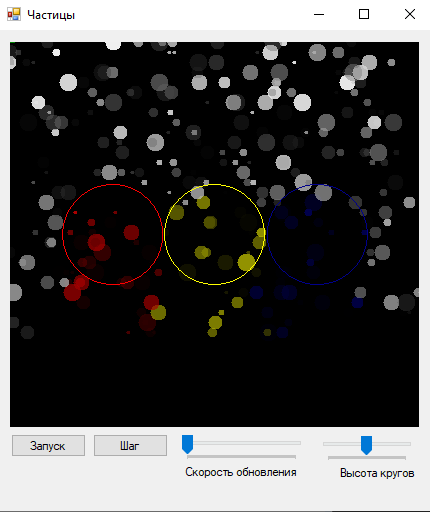
}

}

}

# 7 Описание работы интерфейса

Меняя trackback **Скорость обновления** мы меняем скорость обновления изображения на форме

  
Рисунок 7.1

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

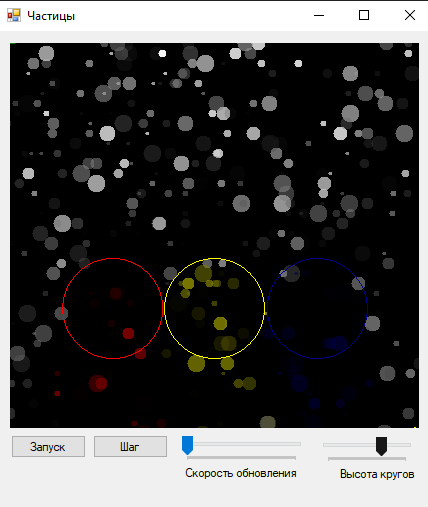


Рисунок 7.2

Нажимая на кнокпу Запуск мы останавливаем обновление изображения. Нажимая вновь обновление продолжается.

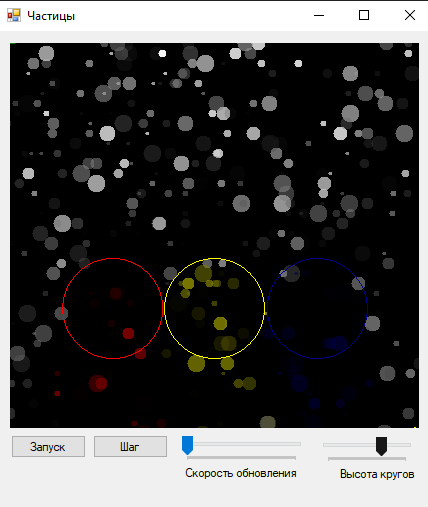


Рисунок 7.3

Частицы попадающие в область действия специальной точки меняют свой цвет в соответствии с цветом этой точки

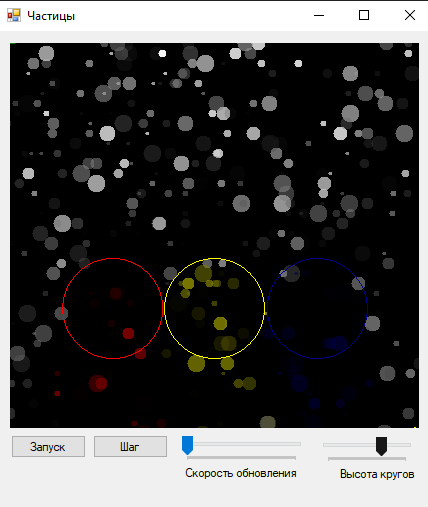


Рисунок 7.4

Нажимая на кнокпу Шаг мы обновляем изображение на один кадр.

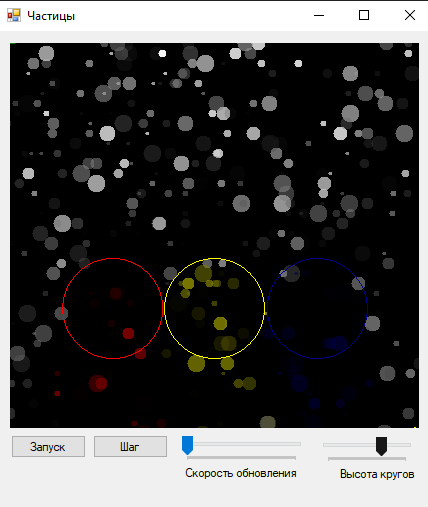


Рисунок 7.5

# Заключение

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

* Ползунок «Скорость обновления», для управления скоростью обновления изображения
* Ползунок «Высота кругов», для управления положения специальных точек
* Кнопка запуск, которая запускает обновление изображения
* Кнопка Шаг, которая обновляет изображение
* Специальная точка «Гравитон[Цветовой круг]» которая меняет цвет частицы, которая в неё попала

Приложение было всесторонне протестировано и отлажено. Все поставленные в рамках курсовой работы задачи были выполнены.

В рамках выполнения данной работы был получен обширный спектр сведений об объектно-ориентированном языке программирования C#, разработки графических приложений и эмуляции сложных динамических систем.

# Список использованной литературы

1. Троелсен, Эндрю Язык программирования C# 5.0 и платформа .NET 4.5 / Эндрю Троелсен. - М.: Вильямс, 2015. - 486 c.
2. Вагнер, Билл С# Эффективное программирование / Билл Вагнер. - М.: ЛОРИ, 2013. - 320 c.
3. Ишкова, Э. А. Самоучитель С#. Начала программирования / Э.А. Ишкова. - М.: Наука и техника, 2013. - 496 c.
4. Биллинг В.А. Основы программирования на C#. - Т.: Интернет-университет информационных технологий, Бином, 2012. - 488 с.
5. Евдокимов П.В. C# на примерах. - М.: Наука и техника,2016. - 304 с.
6. Нейгел К., Ивьен Б., Глинн Д., Уотсон К., Скиннер М. C# 4.0 и платформа .NET 4, 2011.
7. Система частиц, часть 1. [Электронный ресурс] // aqua.tealeaf.su : чаинка, 2021. URL: http://aqua.tealeaf.su/particle-system.html (дата обращения: 01.12.2021).