

Содержание

[Введение 4](#_Toc58190883)

[1. Индивидуальный вариант 5](#_Toc58190884)

[2. Внешний вид главного окна 6](#_Toc58190885)

[3. Код эмиттера 7](#_Toc58190886)

[4. Код специальных точек 10](#_Toc58190887)

[5. Код частиц 12](#_Toc58190888)

[6. Код формы 12](#_Toc58190889)

[7 Описание работы интерфейса 18](#_Toc58190890)

[Заключение 20](#_Toc58190891)

[Список использованной литературы 21](#_Toc58190892)

# 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.

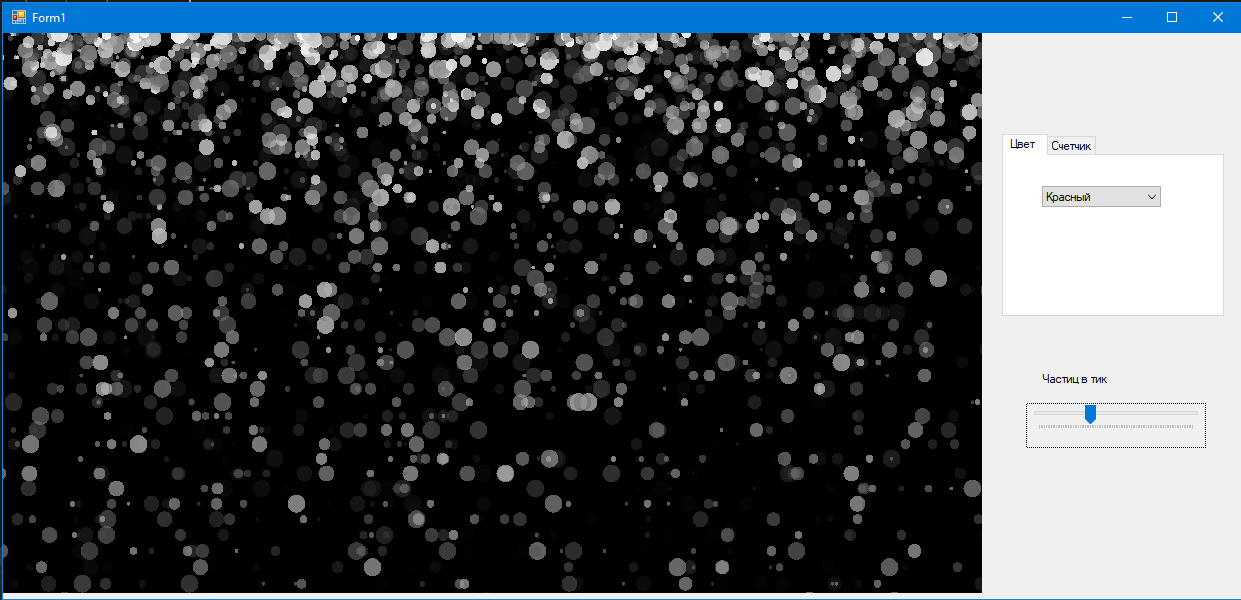
Для реализации используется язык C#. C# – это язык программирования, предназначенный для разработки самых разнообразных приложений. Язык C# прост, строго типизирован и объектно-ориентирован. Благодаря множеству нововведений C# обеспечивает возможность быстрой разработки приложений, но при этом сохраняет выразительность и элегантность, присущую си подобным языкам.

# 2 Индивидуальный вариант

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

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

# 3 Внешний вид главного окна

Рисунок 1.1 – Внешний вид интерфейса

# 3 Код эмиттера

public class Emitter

{

public float GravitationX = 0;

public float GravitationY = 1;

public int MousePositionX = 0;

public int MousePositionY = 0;

public int ParticlesCount = 0;

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

public List<Particle> particles = new List<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.X += particle.SpeedX;

particle.Y += particle.SpeedY;

foreach (var point in impactPoints)

{

point.ImpactParticle(particle);

}

particle.SpeedX += GravitationX;

particle.SpeedY += GravitationY;

}

}

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 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);

var p = particle as ParticleColorful;

p.FromColor = ColorFrom;

}

public virtual Particle CreateParticle()

{

var particle = new ParticleColorful();

particle.FromColor = ColorFrom;

particle.ToColor = ColorTo;

return particle;

}

}

public class TopEmitter : Emitter

{

public int Width;

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

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

);

}

}

public class CountPoint : IImpactPoint

{

public int Power = 100;

public int count = 0;

public override void ImpactParticle(Particle particle)

{

if (count < 15000)

{

float gX = X - particle.X;

float gY = Y - particle.Y;

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

//Если частица внутри точки

if (r + particle.Radius < Power / 1.7 && particle.Life > 0)

{

//Убить частицу

particle.Life = 0;

//Увеличить кол-во на 1

count++;

}

}

else

{

count = 0;

}

}

public override void Render(Graphics g)

{

var color = Color.FromArgb(count / 60, Color.Red);

//Нарисовать точку

g.FillEllipse(

new SolidBrush(color),

X - Power / 2,

Y - Power / 2,

Power,

Power

);

g.DrawEllipse(

new Pen(Color.Red, 3),

X - Power / 2,

Y - Power / 2,

Power,

Power

);

//Вывести информацию

var stringFormat = new StringFormat();

stringFormat.Alignment = StringAlignment.Center;

stringFormat.LineAlignment = StringAlignment.Center;

g.DrawString(

count.ToString(),

new Font("Verdana", 10),

new SolidBrush(Color.White),

X,

Y,

stringFormat

);

}

}

public class ColorPoint : IImpactPoint

{

public int Power = 100;

public Color color;

public override void ImpactParticle(Particle particle)

{

float gX = X - particle.X;

float gY = Y - particle.Y;

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

//Если частица внутри точки

if (r + particle.Radius < Power / 1.7 && particle.Life > 0)

{

//Перекрасить частицу

(particle as ParticleColorful).FromColor = color;

}

}

public override void Render(Graphics g)

{

//Нарисовать точку

g.DrawEllipse(

new Pen(color, 3),

X - Power / 2,

Y - Power / 2,

Power,

Power

);

}

}**5 Код частиц**

public class Particle

{

public int Radius;

public float X;

public float Y;

public float SpeedX;

public float SpeedY;

public float Life;

public bool isTP;

public static Random rand = new Random();

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);

isTP = true;

}

public virtual void Draw(Graphics g)

{

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

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

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

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;

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);

b.Dispose();

}

}

# 6 Код формы

public partial class Form1 : Form

{

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

Emitter emitter;

public Form1()

{

InitializeComponent();

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

emitter = new TopEmitter

{

Width = picDisplay.Width,

GravitationY = 0.25f

};

emitters.Add(this.emitter);

comboBox1.SelectedIndex = 0;

tbTick.Value = emitter.ParticlesPerTick;

picDisplay.MouseWheel += picDisplay\_MouseWheel;

tickInfo.Text = emitter.ParticlesPerTick.ToString();

}

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

{

if (tabControl1.SelectedIndex == 0)

{

foreach (var p in emitter.impactPoints)

{

if (p is ColorPoint)

{

var x = (p as ColorPoint).X - e.X;

var y = (p as ColorPoint).Y - e.Y;

double r = Math.Sqrt(x \* x + y \* y);

if (r <= (p as ColorPoint).Power / 2)

{

if(e.Delta<0 && (p as ColorPoint).Power > 30)

{

(p as ColorPoint).Power -= 10;

}

if (e.Delta > 0 && (p as ColorPoint).Power < 300)

{

(p as ColorPoint).Power += 10;

}

}

}

}

}

if (tabControl1.SelectedIndex == 1)

{

foreach (var p in emitter.impactPoints)

{

if (p is CountPoint)

{

var x = (p as CountPoint).X - e.X;

var y = (p as CountPoint).Y - e.Y;

double r = Math.Sqrt(x \* x + y \* y);

if (r <= (p as CountPoint).Power / 2)

{

if (e.Delta < 0 && (p as CountPoint).Power > 30)

{

(p as CountPoint).Power -= 10;

}

if (e.Delta > 0 && (p as CountPoint).Power < 300)

{

(p as CountPoint).Power += 10;

}

}

}

}

}

}

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

{

emitter.UpdateState();

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

{

g.Clear(Color.Black);

emitter.Render(g);

}

picDisplay.Invalidate();

foreach(var p in emitter.particles)

{

if(p.X > picDisplay.Width || p.Y > picDisplay.Height)

{

p.Life = 0;

}

}

}

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

{

}

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

{

if(e.Button == MouseButtons.Left)

{

if(tabControl1.SelectedIndex == 0)

{

int x = e.X;

int y = e.Y;

Color color = Color.Red;

switch (comboBox1.SelectedIndex)

{

case 0:

color = Color.Red;

break;

case 1:

color = Color.Navy;

break;

case 2:

color = Color.Yellow;

break;

case 3:

color = Color.Purple;

break;

case 4:

color = Color.Green;

break;

}

emitter.impactPoints.Add(new ColorPoint

{

X = x,

Y = y,

color = color

}) ;

}

if (tabControl1.SelectedIndex == 1)

{

int x = e.X;

int y = e.Y;

emitter.impactPoints.Add(new CountPoint

{

X = x,

Y = y

});

}

}

if(e.Button == MouseButtons.Right)

{

if(tabControl1.SelectedIndex == 0)

{

foreach (var p in emitter.impactPoints)

{

if (p is ColorPoint)

{

var x = (p as ColorPoint).X - e.X;

var y = (p as ColorPoint).Y - e.Y;

double r = Math.Sqrt(x \* x + y \* y);

if (r <= (p as ColorPoint).Power / 2)

{

emitter.impactPoints.Remove((p as ColorPoint));

break;

}

}

}

}

if (tabControl1.SelectedIndex == 1)

{

foreach (var p in emitter.impactPoints)

{

if (p is CountPoint)

{

var x = (p as CountPoint).X - e.X;

var y = (p as CountPoint).Y - e.Y;

double r = Math.Sqrt(x \* x + y \* y);

if (r <= (p as CountPoint).Power / 2)

{

emitter.impactPoints.Remove((p as CountPoint));

break;

}

}

}

}

}

}

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

{

emitter.ParticlesPerTick = tbTick.Value;

tickInfo.Text = emitter.ParticlesPerTick.ToString();

}

}**7 Описание работы интерфейса**

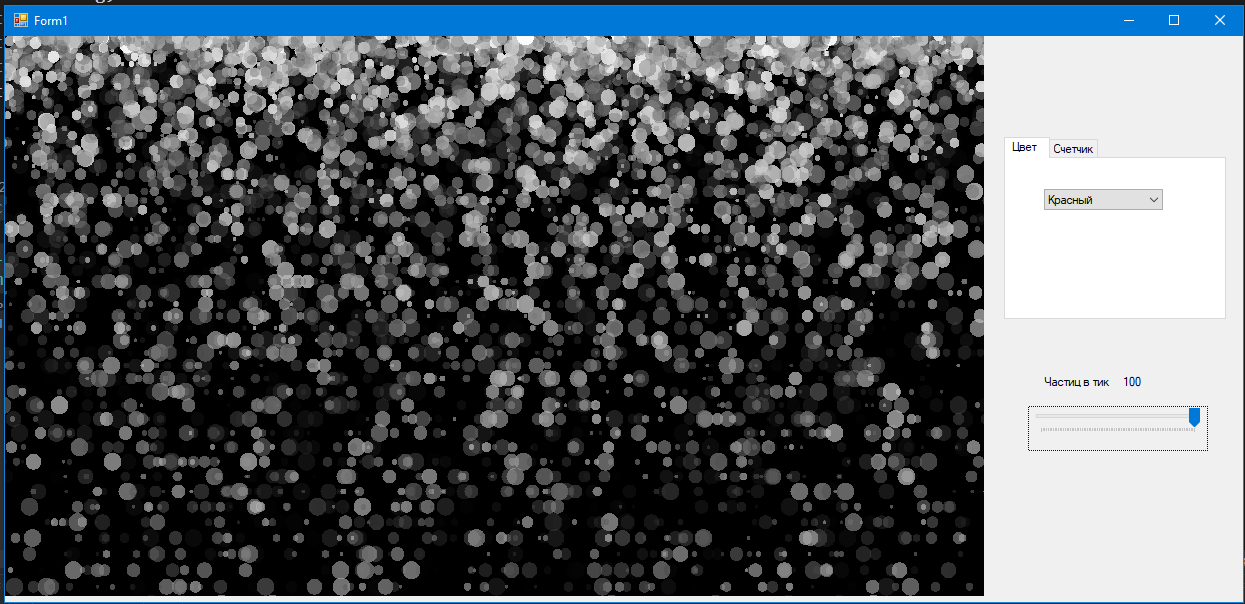
Trackbar **Частиц в тик** отвечaет зa изменение количества частиц в тик

Рисунок 2.1 – Изменение количества частиц в тик Эммитера

Частицы попадающие в область действия специальной точки умирают, но подсчитываются

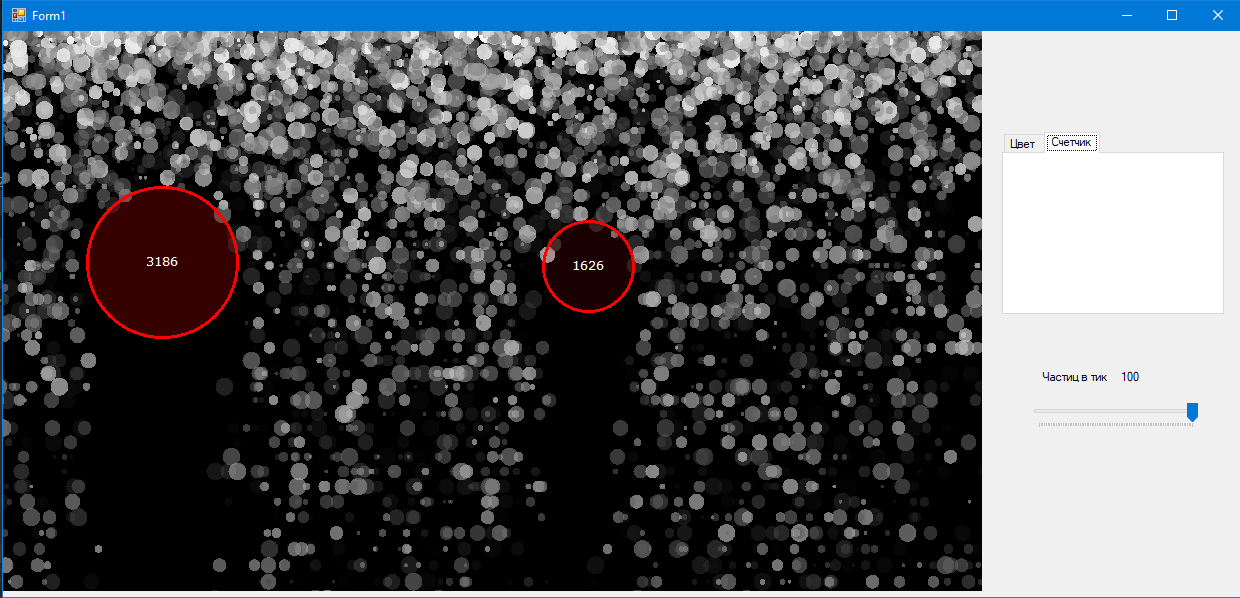


Рисунок 2.2 – Точка-счетчик частиц

Частицы попадающие в область действия специальной точки меняют цвет

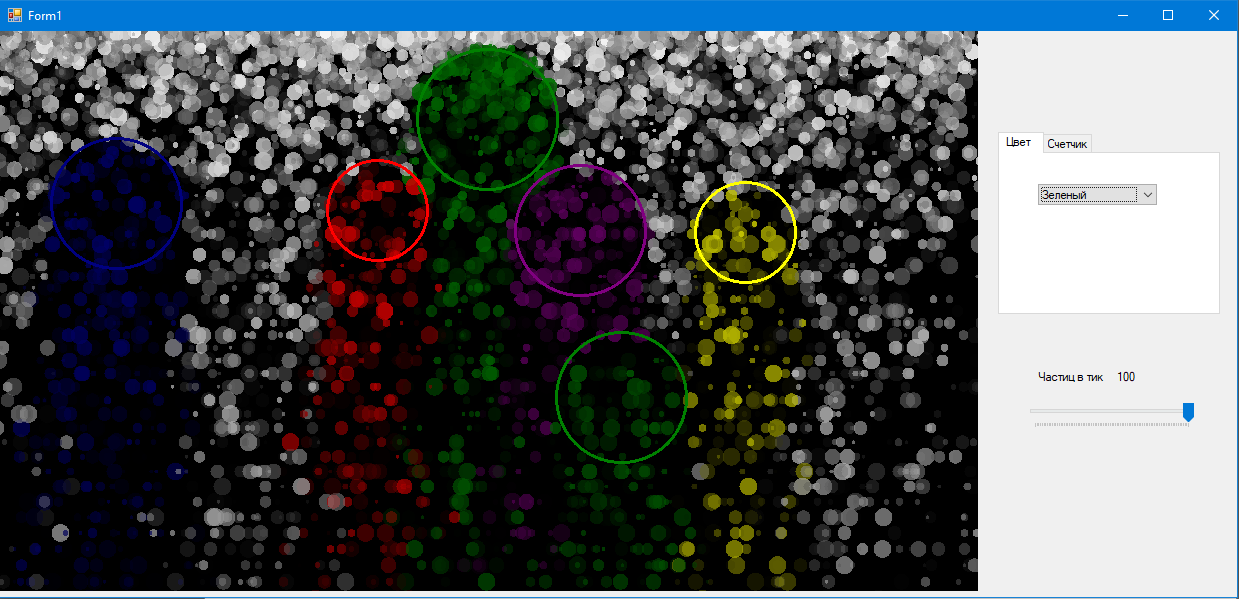


Рисунок 2.3 – Перекрашиваение частиц с помощью специальных точек

# Заключение

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

* Ползунок «Количество частиц в тик» - для управления количеством частицы на форме,
* Ползунок «Время жизни» - для управления временем жизни частицы на форме,
* Специальная точка «Счетчик», которая подчитывает количество частиц, которые попадают в радиус ее действия
* Специальная точка «Раскраска», которая меняет цвет частицы, когда та попадает в ее радиус

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).