Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт/Факультет – Информационные системы и технологии   
 наименование

**Название работы** – Обработка событий

Отчет по лабораторной работе № 5

по дисциплине Технология программирования   
 наименование учебной дисциплины

Выполнилcтудент ИСТб-20-3 Д.А.Пинегин

номер группы подпись И. О. Фамилия

дата

Принял Доцент З.А. Бахвалова

должность подпись И.О. Фамилия

дата

Иркутск – 2021

# Лабораторная работа №5

### Постановка задачи

Создать приложение Обработка Событий

1. Реализовать новый объект, который будет исчезать при пересечении с игроком и появляться на новом месте
2. Реализовать вывод очков. Увеличивать количество очков при пересечении с объектом, добавленным в предыдущем пункте. Дополнительно добавить на поле несколько зеленых кругов.
3. Добавить счетчик к зеленому объекту. Если игрок не успел добраться до объекта, то переместить его. Сам счетчик хранить в зеленом кружке, событие конца отсчета должен генерировать зеленый кружок

**Словесное описание алгоритма**

1. Создаем графическое приложение, добавляем Picturebox, richTextBox, timer и label
2. Далее создаем четыре класса: BaseObject, Player, Marker, Krug
3. Заполняем классы кодом с условием, что разность координат X и Y равна местоположению маркера
4. Скорость игрока зависит от X и Y, к которым прибавляются вектора скорости X и Y
5. Координаты маркеры равны векторам X и Y нажатию клавиши мыши
6. Создаем маркер движения для игрока нажатием клавиши мыши

### Таблица спецификаций

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | Диапазон |
| Спецификации класса BaseObject | | | |
| X | Координата X | public float | [0; 800] |
| Y | Координата Y | public float | [0; 400] |
| Angle | Угол поворота | public float | [0; 360] |

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | Параметры |
| Методы класса BaseObject | | | |
| GetTransform | Определение координат объекта | public Matrix | — |
| Render | Отрисовка объекта | public virtual void | Graphics graphic |
| GetGraphicsPath | Графический путь объекта | public virtual GraphicsPath | — |
| Overlaps | Метод определения пересечений двух объектов | public virtual bool | BaseObject obj, Graphics graphic |
| Overlap | Вызов делегата | public virtual void | BaseObject obj |

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | Диапазон |
| Спецификация методов класса BaseObject | | | |
| graphic | Графика | Graphics | — |
| path1 | Графический путь 1 | GraphicsPath | — |
| path2 | Графический путь 2 | GraphicsPath | — |
| region | Область соприкосновения точек | Region | — |
| obj | Объект, с которым происходит соприкосновение | BaseObject | — |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя | | Назначение | | Тип | | Диапазон |
| Спецификация класса Player | | | | | | |
| X | Вектор скорости X | | private float | | [0; 800] | |
| Y | Вектор скорости Y | | private float | | [0; 400] | |
| OnMarkerOverlap | Делегат пересечения игрока с маркером | | public Action<Marker>? | | — | |
| OnKrugOverlap | Делегат пересечения игрока с кругом | | public Action<Krug | | — | |

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | Параметры |
| Методы класса Player | | | |
| Render | Отрисовка объекта | public override void | Graphics graphic |
| GetGraphicsPath | Графический путь объекта | public override GraphicsPath | — |
| Overlap | Вызов делегата | public override void | BaseObject obj |

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | Диапазон |
| Спецификации методов класса Player | | | |
| graphic | Графика | Graphics | — |
| path | Графический путь | GraphicsPath | — |
| obj | Объект, с которым происходит соприкосновение | BaseObject | — |

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | Параметры |
| Методы класса Marker | | | |
| Render | Отрисовка объекта | public override void | Graphics graphic |
| GetGraphicsPath | Графический путь объекта | public override GraphicsPath | — |

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | Диапазон |
| Спецификации методов класса Marker | | | |
| path | Графический путь | GraphicsPath | — |

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | Параметры |
| Методы класса Krug | | | |
| Render | Отрисовка объекта | public override void | Graphics graphic |
| GetGraphicsPath | Графический путь объекта | public override GraphicsPath | — |

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | Диапазон |
| Спецификации методов класса Krug | | | |
| path | Графический путь | GraphicsPath | — |

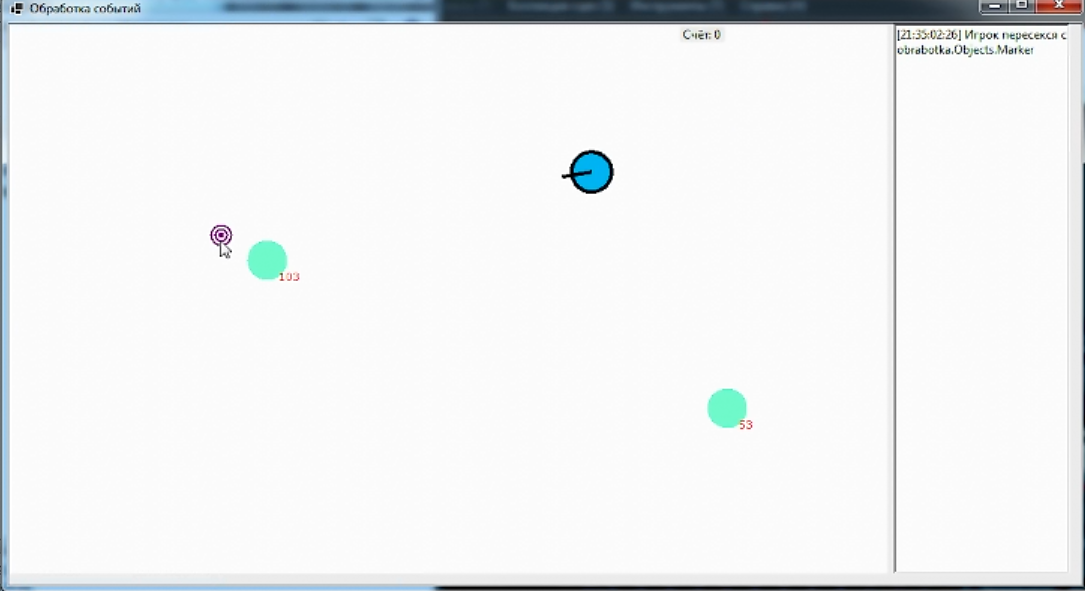
|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | Диапазон |
| Спецификации класса Form1 | | | |
| objects | Список объектов | private readonly List<BaseObject> | — |
| player | Итог | private readonly Player | — |
| marker | Маркер | private Marker | — |
| score | Счёт игрока | private ushort | [0; 65535] |

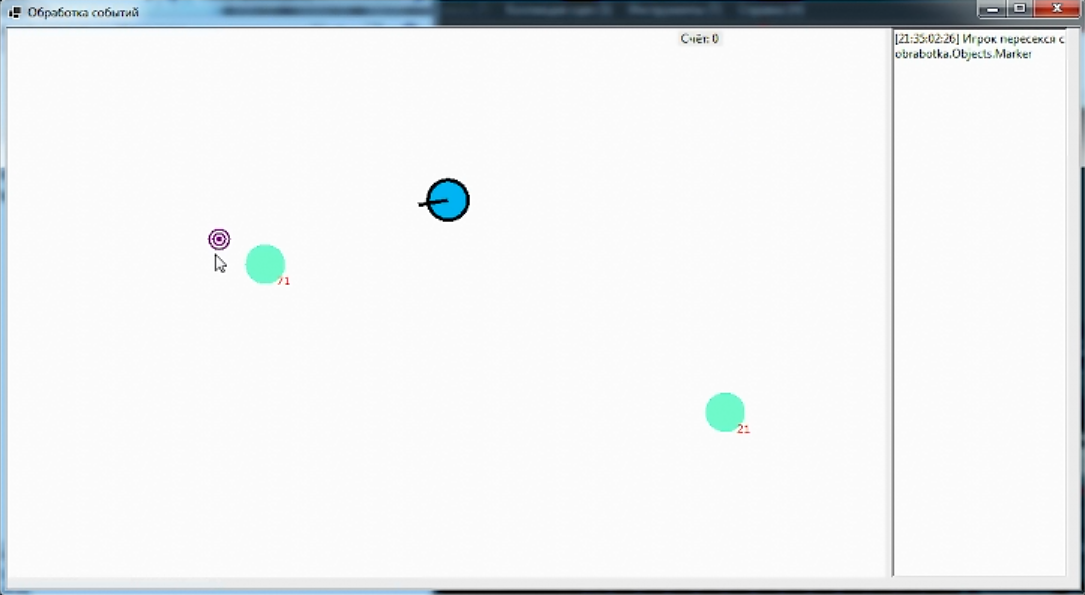
|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | Параметры |
| Методы класса Form1 | | | |
| PictureBox\_Paint | Событие отрисовки PictureBox | public void | object sender, PaintEventArgs e |
| PictureBox\_MouseClick | Событие нажатия на PictureBox | public void | object sender, MouseEventArgs e |
| Timer\_Tick | Событие обновление таймера | public void | BaseObject obj |
| UpdatePlayer | Обновление координат игрока | private void | — |
| UpdateMarker | Обновление координат маркера | private void | MouseEventArgs e |
| GenerateCircle | Случайная генерация координат круга | private static GreenСircle | GreenСircle сircle |

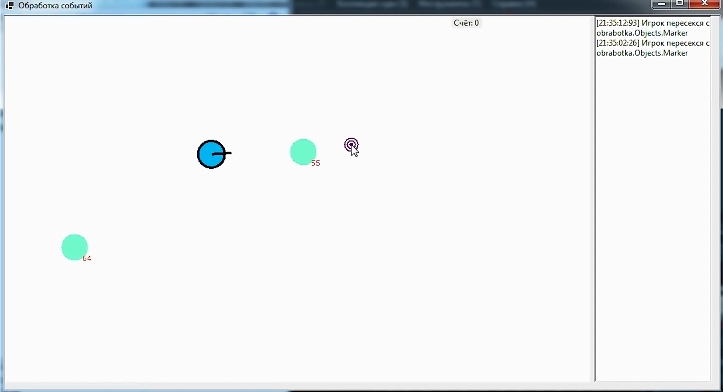
|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Назначение | Тип | Диапазон |
| Таблица спецификаций методов Form1 | | | |
| graphic | Графика | Graphics | — |
| dX | Новая координата X | float | [0;800] |
| dY | Новая координата Y | float | [0;400] |
| length | Расстояние до маркера | float | [0;894.4272] |
| random | Объект класса Random | Random | — |

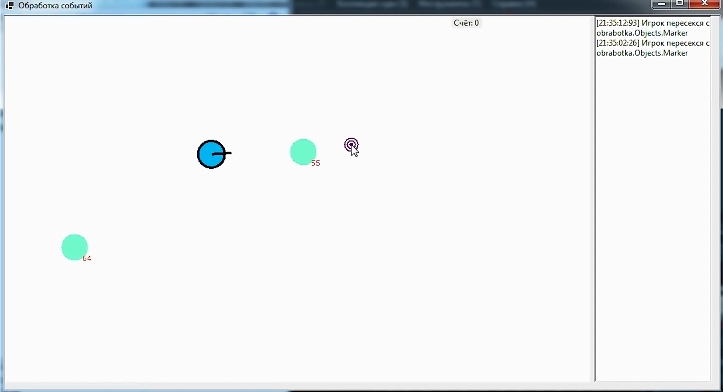
### Таблица тестов

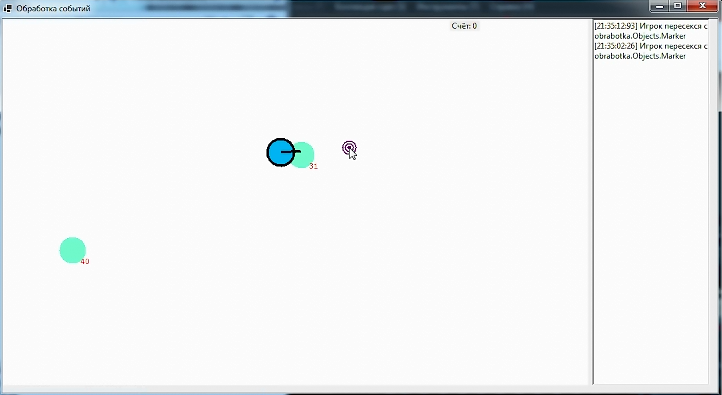
|  |  |
| --- | --- |
| Номер теста | Назначение |
| 1 | Проверка перемещения игрока к маркеру |
| 2 | Проверка подбора круга и увеличение очков |
| 3 | Проверка на создания больше одного круга |
| 4 | Проверка таймера исчезновения |

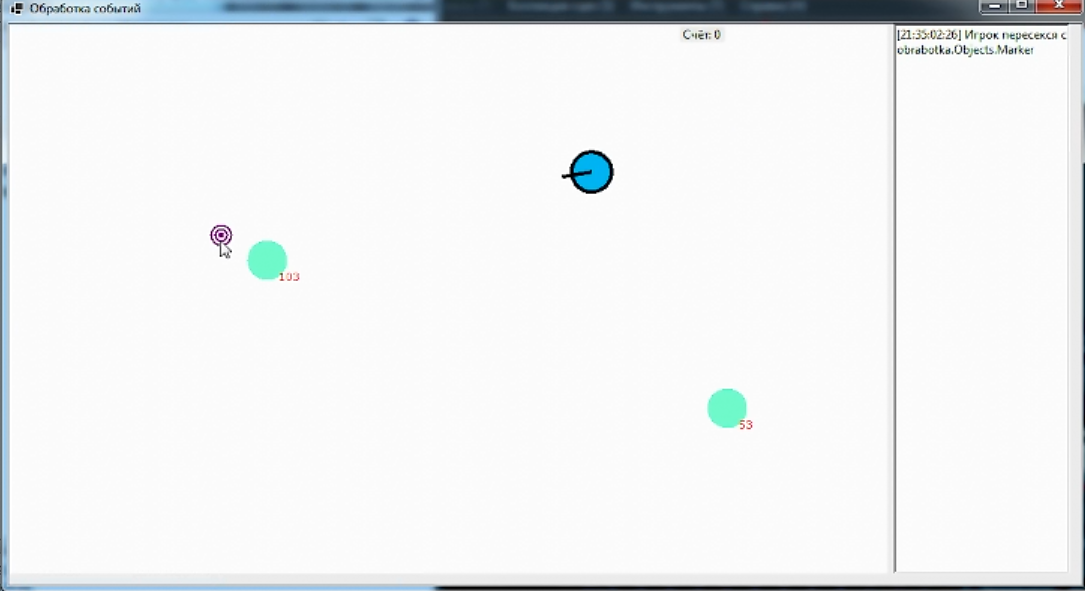
Тест №1

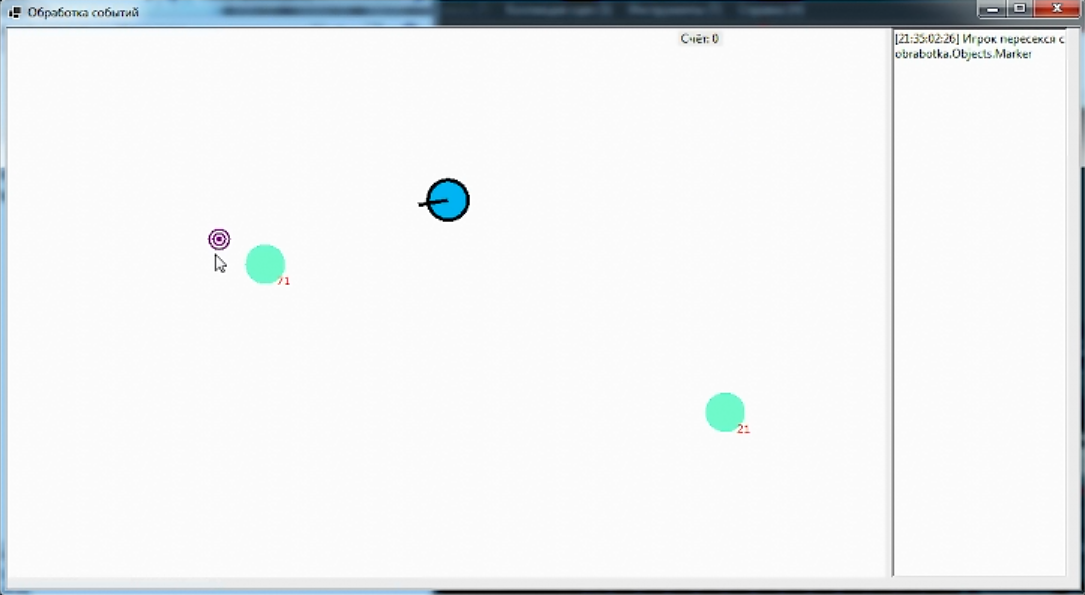


Тест №2

Тест №3



Тест №4



### Код программы

**Класс Form1.cs**

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;

using obrabotka.Objects;

namespace obrabotka

{

public partial class Form1 : Form

{

List<BaseObject> objects = new List<BaseObject>();

Player player;

Marker marker;

Krug krug;

Krug krug1;

int ochko = 0;

Random r = new Random();

public Form1()

{

InitializeComponent();

player = new Player(pbMain.Width / 2, pbMain.Height / 2, 0);

// добавляю реакцию на пересечение

player.onOverlap += (p, obj) =>

{

txtLog.Text = $"[{DateTime.Now:HH:mm:ss:ff}] Игрок пересекся с {obj}\n" + txtLog.Text;

};

player.onMarkerOverlap += (m) =>

{

objects.Remove(m);

marker = null;

};

player.onKrugOverlap += (m) =>

{

GenerateCircle(m);

ochko++;

Score.Text = $"Счёт: "+ochko;

};

marker = new Marker(pbMain.Width / 2+50, pbMain.Height / 2+50, 0);

krug = new Krug(0, 0,0);

GenerateCircle(krug);

krug1 = new Krug(0, 0, 0);

GenerateCircle(krug1);

objects.Add(krug);

objects.Add(krug1);

objects.Add(marker);

objects.Add(player);

}

private void GenerateCircle(Krug сircle)

{

Random random = new Random();

сircle.X = random.Next() % 780 + 40;

сircle.Y = random.Next() % 380 + 40;

}

private void pbMain\_Paint(object sender, PaintEventArgs e)

{

var g = e.Graphics;

g.Clear(Color.White);

peredvig();

// пересчитываем пересечения

foreach (var obj in objects.ToList())

{

if (obj != player && player.Overlaps(obj, g))

{

player.Overlap(obj);

obj.Overlap(player);

}

if (obj != krug && krug.Overlaps(obj, g))

{

krug.Overlap(obj);

obj.Overlap(krug);

}

}

// рендерим объекты

foreach (var obj in objects)

{

g.Transform = obj.GetTransform();

obj.Render(g);

}

}

private void peredvig()

{

if (marker != null)

{

float dx = marker.X - player.X;

float dy = marker.Y - player.Y;

float lenght = MathF.Sqrt(dx \* dx + dy \* dy);

dx /= lenght;

dy /= lenght;

player.vX += dx \* 0.5f;

player.vY += dy \* 0.5f;

player.Angle = 90 - MathF.Atan2(player.vX, player.vY) \* 180 / MathF.PI;

}

player.vX += -player.vX \* 0.1f;

player.vY += -player.vY \* 0.1f;

// пересчет позиция игрока с помощью вектора скорости

player.X += player.vX;

player.Y += player.vY;

}

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

{

foreach (var obj in objects.ToList())

{

if (obj is Krug krug)

{

krug.time--;

if (krug.time <= 0)

{

GenerateCircle(krug);

krug.time = 130 + r.Next() % 70;

}

}

}

pbMain.Invalidate();

}

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

{

if (marker == null)

{

marker = new Marker(0, 0, 0);

objects.Add(marker);

}

marker.X = e.X;

marker.Y = e.Y;

}

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

{

}

}

}

**Player.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Text;

namespace obrabotka.Objects

{

class Player : BaseObject

{

public float vX, vY;

public Player(float x,float y, float angle) : base(x,y,angle)

{

}

public Action<Marker> onMarkerOverlap;

public Action<Krug> onKrugOverlap;

public override void Render(Graphics g)

{

g.FillEllipse(new SolidBrush(Color.DeepSkyBlue), -20, -20, 40, 40);

g.DrawEllipse(new Pen(Color.Black,4), -20, -20, 40, 40);

g.DrawLine(new Pen(Color.Black, 4), 0, 0, 30, 0);

}

public override GraphicsPath GetGraphicsPath()

{

var path = base.GetGraphicsPath();

path.AddEllipse(-10, -10, 20, 20);

return path;

}

public override void Overlap(BaseObject obj)

{

base.Overlap(obj);

if(obj is Marker)

{

onMarkerOverlap(obj as Marker);

}

else if(obj is Krug krug)

{

onKrugOverlap.Invoke(krug);

}

}

}

}

**Krug.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Text;

namespace obrabotka.Objects

{

class Krug : BaseObject

{

public int time = 0;

public Action<Krug> onKrugOverlap;

public Krug(float x, float y, float angle) : base(x, y, angle)

{

Random ran = new Random();

time = 130 + ran.Next() % 70;

}

public override void Render(Graphics g)

{

g.FillEllipse(new SolidBrush(Color.Aquamarine), -20, -20, 40, 40);

g.DrawString($"{time}", new Font("Verdana", 8), new SolidBrush(Color.Red), 10, 10);

}

public override GraphicsPath GetGraphicsPath()

{

var path = base.GetGraphicsPath();

path.AddEllipse(-10, -10, 20, 20);

return path;

}

public override void Overlap(BaseObject obj)

{

base.Overlap(obj);

if (obj is Krug)

{

onKrugOverlap(obj as Krug);

}

}

}

}

**BaseObjects.cs**

using System;

using System.Collections.Generic;

using System.Text;

using System.Drawing.Drawing2D;

using System.Drawing;

namespace obrabotka.Objects

{

class BaseObject

{

public float X;

public float Y;

public float Angle;

public BaseObject(float x,float y,float angle)

{

X = x;

Y = y;

Angle = angle;

}

public Action<BaseObject, BaseObject> onOverlap;

public Matrix GetTransform()

{

var matrix = new Matrix();

matrix.Translate(X,Y);

matrix.Rotate(Angle);

return matrix;

}

public virtual void Render(Graphics g)

{

}

public virtual GraphicsPath GetGraphicsPath()

{

return new GraphicsPath();

}

public virtual bool Overlaps(BaseObject obj, Graphics g)

{

// берем информацию о форме

var path1 = this.GetGraphicsPath();

var path2 = obj.GetGraphicsPath();

// применяем к объектам матрицы трансформации

path1.Transform(this.GetTransform());

path2.Transform(obj.GetTransform());

// используем класс Region, который позволяет определить

// пересечение объектов в данном графическом контексте

var region = new Region(path1);

region.Intersect(path2); // пересекаем формы

return !region.IsEmpty(g); // если полученная форма не пуста то значит было пересечение

}

public virtual void Overlap(BaseObject obj)

{

if(this.onOverlap!= null)

{

this.onOverlap(this, obj);

}

}

}

}