Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт/Факультет - ИИТиАД

**Название работы – Обработка событий**

Отчет по лабораторной работе № 5

по дисциплине

Технология программирования

Выполнил cтудент ИСТб-20-2 М.А. Пигарев

номер группы подпись И. О. Фамилия

21.09.21

дата

Принял Доцент З.А. Бахвалова

21.09.21 Должность подпись И. О. Фамилия

дата

Иркутск –2021

Оглавление

[Постановка задачи 3](#_Toc87952047)

[Class BaseObject 3](#_Toc87952048)

[Class Form1 4](#_Toc87952049)

[Class Marker 6](#_Toc87952050)

[Class MyRectangle 7](#_Toc87952051)

[Class Player 7](#_Toc87952052)

[Class Red 8](#_Toc87952053)

[Таблица тестов 9](#_Toc87952054)

[Код программы 9](#_Toc87952055)

[Результаты тестов 15](#_Toc87952056)

[Ссылка на Git 18](#_Toc87952057)

### Постановка задачи

Добавить красный увеличивающийся в размерах круг, при пересечении с которым количество очков уменьшается на 1, а круг сбрасывает свой размер и меняет позицию. Размер хранить в красном кружке

### Class BaseObject

Поля class BaseObject

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Назначение | Тип | Диапазон |
| X | Координата объекта по оси X | float | - |
| Y | Координата объекта по оси Y | float | - |
| Angle | Угол поворота объекта | float | - |

Методы class BaseObject

|  |  |  |
| --- | --- | --- |
| Название | Назначение | Тип |
| BaseObject(float x, float y, float angle) | Конструктор с заданными параметрами | void |
| Render | Ничего не делает. Нужен для того, чтобы вызывать этот метод у наследников, когда идёт перебор объектов | void |
| GetTransform | Матрица трансформации. Позволяет двигать объекты по координатам X и Y | matrix |
| GetGraphicsPath | Возвращает форму объекта | void |
| Overlaps | Отслеживает пересечение объектов | bool |
| Overlap | Модуль, который вызывает событие пересечения игрока с другим объектом | void |

Словесное описание метода GetTransform

1. Создаём переменную типа Matrix
2. Записываем в неё координаты
3. Записываем в неё угол
4. Возвращаем матрицу в код

Таблица спецификаций метода GetTransform

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Назначение | Тип | Диапазон |
| matrix | Хранимая матрица трансформаций | var | - |

Словесное описание метода Overlaps

1. Берем информацию о форме
2. Применяем к объектам матрицы трансформации
3. Используем класс Region, который позволяет определить пересечение объектов в данном графическом контексте
4. Пересекаем формы
5. Если полученная форма не пуста, то значит было пересечение
6. Возвращаем значение true или false в код, в зависимости от пересечения

Таблица спецификаций метода Overlaps

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Назначение | Тип | Диапазон |
| Path1 | Информация о первой форме | var | - |
| Path2 | Информация о второй форме | var | - |
| region | Хранит первую форму | var | - |

Словесное описание метода GetGraphicsPath

Метод, с который описывает форму объекта. Возвращает экземпляр класса GraphicsPath.

Словесное описание метода Overlap

Если к полю есть привязанные функции, то вызываем их.

### Class Form1

Поля class Form1

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Назначение | Тип | Диапазон |
| objects | Список. В нём хранятся элементы, которые будут отрисованы на форме | BaseObject | - |
| player | Фигурка игрока | Player | - |
| marker | Маркер, появляется на pictureBox-е, если на неё нажать мышкой. К нему притягивается фигурка игрока | Marker | - |
| circle\_1 | Первый синий круг (за пересечение с ним счёт увеличивается) | MyRectangle | - |
| circle\_2 | Второй синий круг (за пересечение с ним счёт увеличивается) | MyRectangle | - |
| RedZone | Объект красной зоны (за пересечение с ним счёт уменьшается) | Red | - |
| score | Счёт игрока | Int | - |

Методы class Form1

|  |  |  |
| --- | --- | --- |
| Название | Назначение | Тип |
| Form1 | Инициализируются объекты. Также здесь прописаны реакции на события | void |
| pbMain\_Paint | Выполняет отрисовку объектов на экране | void |
| timer1\_Tick | Таймер. Вызывает метод pbMain\_Paint | void |
| pbMain\_MouseClick | Создание маркера по клику мыши | void |
| updatePlayer | Ускорение игрока к маркеру, также пересчитывает позицию игрока | void |

Словесное описание метода Form1

1. Запускаем форму
2. Создаём генератор случайных чисел
3. Создаём два объекта синих круга и один объект красной зоны. Все координаты генерируются случайным образом
4. Реакция на пересечение игрока с каким-то объектом: выводим информацию о пересечении на экран, также указываем время пересечения
5. Реакция на пересечение игрока с маркером: удаляем маркер из списка объектов, “обнуляем” маркер
6. Реакция на пересечение игрока с синим кругом: удаляем круг из списка объектов, “обнуляем” круг, задаём все параметры круга равные 0, добавляем круг в список объектов, генерируем случайные координаты для круга, увеличиваем счётчик очков на один
7. Реакция на пересечение игрока с красной зоной: удаляем красную зону из списка объектов, “обнуляем” зону, задаём все параметры зоны равные 0, добавляем зону в список объектов, генерируем случайные координаты для зоны, уменьшаем счётчик очков на один
8. Добавляем красную зону, оба круга и игрока в список объектов

Таблица спецификаций метода Form1

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Назначение | Тип | Диапазон |
| rand | Генератор случайных чисел. Нужен для того, чтобы генерировать случайные координаты | Random | integer |

Словесное описание метода pbMain\_Paint

1. “Вытаскиваем” объект графики и присваиваем его переменной g
2. Заполняем весь задний фон белым цветом
3. На label выводим информацию о текущем счёте
4. Вызываем функцию, которая выполняет пересчёт позиции игрока
5. В цикле перебираем каждый элемент списка, если он не равен игроку и пересекается с игроком, то вызываем событие пересечения игрока с объектом
6. В цикле перебираем каждый элемент списка и отрисовываем их на форме

Таблица спецификаций метода pbMain\_Paint

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Назначение | Тип | Диапазон |
| g | Нужен для того, чтобы “вытащить” объект графики из события Paint | var | - |

Словесное описание метода timer1\_Tick

Вызывает метод, для отрисовки объектов на форме, каждые 30 мс

Словесное описание метода pbMain\_MouseClick

Если маркер ещё не создан, то создаём новый маркер и кладём его в список объектов, после чего задаём ему координаты, которые были получены от клика мыши по форме. Если маркер уже был создан, тогда задаём ему координаты, которые были получены от клика мыши по форме

Словесное описание метода updatePlayer

Если маркер существует, то создаём переменные: dx равную разности между координатами маркера и игрока по оси Х, dy равную разности между координатами маркера и игрока по оси Y, length равную корню квадратному из суммы квадратов dx и dy. Делим переменные dx и dy на переменную length и присваиваем получившиеся значения переменным dx и dy. К переменным ускорения (vX и vY) добавляем dx и dy. После рассчитываем угол, на который необходимо повернуть игрока. Прибавляем к значениям координат игрока переменные vX и vY.

Таблица спецификаций метода updatePlayer

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Назначение | Тип | Диапазон |
| dx | Разница между координатами маркера и игрока по оси X | float | - |
| dy | Разница между координатами маркера и игрока по оси Y | float | - |
| length | Расстояние между маркером и игроком (находится по теореме Пифагора) | float | - |

### Class Marker

Поля class Marker

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Назначение | Тип | Диапазон |
| X | Координата маркера по оси X | float | - |
| Y | Координата маркера по оси Y | float | - |
| Angle | Угол поворота объекта маркера | float | - |

Методы class Marker

|  |  |  |
| --- | --- | --- |
| Название | Назначение | Тип |
| Marker(float x, float y, float angle) | Конструктор с заданными параметрами | void |
| Render | Рисует маркер на форме | void |
| GetGraphicsPath | Возвращает форму, в которой находится размер маркера | GraphicsPath |

Словесное описание метода Render

Рисуем два круга, один больше другого. Выполняем заливку области по меньшему кругу.

Словесное описание метода GetGraphicsPath

Вызываем метод родителя GetGraphicsPath. Получаем пустую форму, заполняем её формой заливки и возвращаем в код.

Таблица спецификаций метода GetGraphicsPath

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Назначение | Тип | Диапазон |
| path | Форма объекта | var | - |

### Class MyRectangle

Поля class MyRectangle

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Назначение | Тип | Диапазон |
| X | Координата круга по оси X | float | - |
| Y | Координата круга по оси Y | float | - |
| Angle | Угол поворота объекта круга | float | - |

Методы class MyRectangle

|  |  |  |
| --- | --- | --- |
| Название | Назначение | Тип |
| MyRectangle (float x, float y, float angle) | Конструктор с заданными параметрами | void |
| Render | Рисует круг на форме | void |
| GetGraphicsPath | Возвращает форму, в которой находится размер круга | GraphicsPath |

Словесное описание метода Render

Рисуем круг заливкой.

Словесное описание метода GetGraphicsPath

Вызываем метод родителя GetGraphicsPath. Получаем пустую форму, заполняем её формой заливки круга и возвращаем в код.

Таблица спецификаций метода GetGraphicsPath

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Назначение | Тип | Диапазон |
| path | Форма объекта | var | - |

### Class Player

Поля class Player

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Назначение | Тип | Диапазон |
| X | Координата круга по оси X | float | - |
| Y | Координата круга по оси Y | float | - |
| Angle | Угол поворота объекта круга | float | - |
| vX | Вектор скорости по оси X | float | - |
| vY | Вектор скорости по оси Y | float | - |

Методы class Player

|  |  |  |
| --- | --- | --- |
| Название | Назначение | Тип |
| Player(float x, float y, float angle) | Конструктор с заданными параметрами | void |
| Render | Рисует круг на форме | void |
| GetGraphicsPath | Возвращает форму, в которой находится размер круга | GraphicsPath |
| Overlap | Вызывает событие пересечения игрока с каким-то объектом | void |

Словесное описание метода Render

Рисуем круг контур и потом выполняем заливку. После рисуем линию, которая будет находится частично на круге и частично выходить вне круга (выполняет функцию указателя движения игрока).

Словесное описание метода GetGraphicsPath

Вызываем метод родителя GetGraphicsPath. Получаем пустую форму, заполняем её формой заливки круга и возвращаем в код.

Таблица спецификаций метода GetGraphicsPath

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Назначение | Тип | Диапазон |
| path | Форма объекта | var | - |

Словесное описание метода Overlap

Если игрок пересёкся с определённым объектом, то вызываем соответствующее событие.

Таблица спецификаций метода Overlap

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Назначение | Тип | Диапазон |
| obj | Объект, с которым пересекается игрок | BaseObject | - |

### Class Red

Поля class Red

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Назначение | Тип | Диапазон |
| X | Координата круга по оси X | float | >=3 |
| Y | Координата круга по оси Y | float | >=3 |
| Angle | Угол поворота объекта круга | float | - |

Методы class Red

|  |  |  |
| --- | --- | --- |
| Название | Назначение | Тип |
| Red(float x, float y, float angle) | Конструктор с заданными параметрами | void |
| Render | Рисует круг на форме | void |
| GetGraphicsPath | Возвращает форму, в которой находится размер круга | GraphicsPath |

Словесное описание метода Render

Рисуем круг заливкой и увеличиваем координаты X и Y на единицу.

Словесное описание метода GetGraphicsPath

Вызываем метод родителя GetGraphicsPath. Получаем пустую форму, заполняем её формой заливки круга и возвращаем в код.

Таблица спецификаций метода GetGraphicsPath

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Назначение | Тип | Диапазон |
| path | Форма объекта | var | - |

### Таблица тестов

| Номер теста | Что проверяем | Входные данные | Выходные данные |
| --- | --- | --- | --- |
| 1 | Работу программы | - | Запустилась форма |
| 2 | Работу программы | - | Пересёк синий круг, счёт увеличился (есть уведомление) |
| 3 | Работу программы | - | Пересёк маркер (есть уведомление) |
| 4 | Работу программы | - | Пересёк красный круг, счёт уменьшился (есть уведомление) |
| 5 | Работу программы | - | Пересёк синий круг и маркер (есть уведомление) |

### Код программы

Код Form1:

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 \_5\_laba.Objects;

namespace \_5\_laba

{

public partial class Form1 : Form

{

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

Player player;

Marker marker;

MyRectangle circle\_1;

MyRectangle circle\_2;

Red RedZone;

int score = 0;

// Random rand = new Random();

public Form1()

{

InitializeComponent();

Random rand = new Random();

circle\_1 = new MyRectangle(rand.Next(30, pbMain.Width - 30), rand.Next(30, pbMain.Height - 30), 0);

circle\_2 = new MyRectangle(rand.Next(30, pbMain.Width - 30), rand.Next(30, pbMain.Height - 30), 0);

RedZone = new Red(rand.Next(30, pbMain.Width - 30), rand.Next(30, pbMain.Height - 30), 0);

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}] Игрок пересекся с {obj}\n" + txtLog.Text;

};

// добавил реакцию на пересечение с маркером

player.OnMarkerOverlap += (m) =>

{

objects.Remove(m);

marker = null;

};

//-------

player.OnMyRectangleOverlap += (m) =>

{

objects.Remove(m);

circle\_1 = null;

circle\_1 = new MyRectangle(0, 0, 0);

objects.Add(circle\_1);

circle\_1.X = rand.Next(30, pbMain.Width - 30);

circle\_1.Y = rand.Next(30, pbMain.Height - 30);

score++;

};

player.OnRedOverlap += (m) =>

{

objects.Remove(m);

RedZone = null;

RedZone = new Red(0, 0, 0);

objects.Add(RedZone);

RedZone.X = rand.Next(30, pbMain.Width - 30);

RedZone.Y = rand.Next(30, pbMain.Height - 30);

score--;

};

objects.Add(RedZone);

objects.Add(circle\_1);

objects.Add(circle\_2);

objects.Add(player);

}

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

{

var g = e.Graphics; // вытащили объект графики из события

g.Clear(Color.White);

label1.Text = "Счёт: " + score;

updatePlayer();

// пересчитываем пересечения

foreach (var obj in objects.ToList())

{

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

{

player.Overlap(obj);

obj.Overlap(player);

}

}

foreach (var obj in objects.ToList())

{

g.Transform = obj.GetTransform();

obj.Render(g);

}

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

{

pbMain.Invalidate();

}

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

{

// добавил создание маркера по клику если он еще не создан

if (marker == null)

{

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

objects.Add(marker); //не забыть пололжить в objects

}

marker.X = e.X;

marker.Y = e.Y;

}

//ускорение игрока

private void updatePlayer()

{

if (marker != null)

{

float dx = marker.X - player.X;

float dy = marker.Y - player.Y;

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

dx /= length; //разделить переменную на значение и ответ присвоить этой же переменной.

dy /= length;

// используем вектор dx, dy

// как вектор ускорения, точнее даже вектор притяжения

// который притягивает игрока к маркеру

player.vX += dx \* 1f;

player.vY += dy \* 1f;

// расчитываем угол поворота игрока

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

}

// тормозящий момент,

// нужен чтобы, когда игрок достигнет маркера произошло постепенное замедление

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

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

// пересчет позиции игрока с помощью вектора скорости

player.X += player.vX;

player.Y += player.vY;

}

}

}

Код класса Marker:

using System;

using System.Collections.Generic;

using System.Text;

using System.Drawing;

using System.Drawing.Drawing2D;

namespace \_5\_laba.Objects

{

class Marker : BaseObject

{

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

{

}

public override void Render(Graphics g)

{

g.FillEllipse(new SolidBrush(Color.Gold), -3, -3, 6, 6);

g.DrawEllipse(new Pen(Color.Gold, 2), -6, -6, 12, 12);

g.DrawEllipse(new Pen(Color.Gold, 2), -10, -10, 20, 20);

}

public override GraphicsPath GetGraphicsPath()

{

var path = base.GetGraphicsPath();

path.AddEllipse(-3, -3, 6, 6); //заполняем пустую форму маркером

return path;

}

}

}

Код класса BaseObject:

using System;

using System.Collections.Generic;

using System.Text;

using System.Drawing;

using System.Drawing.Drawing2D;

namespace \_5\_laba.Objects

{

class BaseObject

{

public float X;

public float Y;

public float Angle;

public Action<BaseObject, BaseObject> OnOverlap;

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

{

X = x;

Y = y;

Angle = angle;

}

public virtual void Render(Graphics g)

{

}

public Matrix GetTransform()

{

var matrix = new Matrix();

matrix.Translate(X, Y);

matrix.Rotate(Angle);

return matrix;

}

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) //если к полю есть привязанные функции

{

this.OnOverlap(this, obj); //то вызываем их

}

}

}

}

Код класса MyRectangle:

using System;

using System.Collections.Generic;

using System.Text;

using System.Drawing;

using System.Drawing.Drawing2D;

namespace \_5\_laba.Objects

{

class MyRectangle : BaseObject

{

// base(x, y, angle) -- вызывает конструктор родительского класса

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

{

// count = 100;

}

// переопределяем Render

public override void Render(Graphics g)

{

g.FillEllipse(new SolidBrush(Color.Blue), 0, 0, 30, 30);

}

//---

public override GraphicsPath GetGraphicsPath()

{

var path = base.GetGraphicsPath();

path.AddEllipse(0, 0, 30, 30);

return path;

}

}

}

Код класса Player:

using System;

using System.Collections.Generic;

using System.Text;

using System.Drawing;

using System.Drawing.Drawing2D;

namespace \_5\_laba.Objects

{

class Player : BaseObject

{

public Action<Marker> OnMarkerOverlap;

public Action<MyRectangle> OnMyRectangleOverlap;

public Action<Red> OnRedOverlap;

public float vX, vY; //поля под вектор скорости

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

{

}

public override void Render (Graphics g)

{

g.FillEllipse(new SolidBrush(Color.Red),-15, -15, 30, 30);

g.DrawEllipse(new Pen(Color.Black,2), -15, -15, 30, 30);

g.DrawLine(new Pen(Color.Black, 2), 0, 0, 25, 0);

}

public override GraphicsPath GetGraphicsPath()

{

var path = base.GetGraphicsPath();

path.AddEllipse(-15, -15, 30, 30);

return path;

}

public override void Overlap(BaseObject obj)

{

base.Overlap(obj);

if(obj is Marker)

{

OnMarkerOverlap(obj as Marker);

}

//---

if (obj is MyRectangle)

{

OnMyRectangleOverlap(obj as MyRectangle);

}

if (obj is Red)

{

OnRedOverlap(obj as Red);

}

//--

}

}

}

Код класса Red:

using System;

using System.Collections.Generic;

using System.Text;

using System.Drawing;

using System.Drawing.Drawing2D;

namespace \_5\_laba.Objects

{

class Red : BaseObject

{

int x = 3;

int y = 3;

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

{

}

// переопределяем Render

public override void Render(Graphics g)

{

g.FillEllipse(new SolidBrush(Color.Tomato), 0-x/2, 0-y/2, x, y);

x++;

y++;

}

//---

public override GraphicsPath GetGraphicsPath()

{

var path = base.GetGraphicsPath();

path.AddEllipse(0-x/2, 0-y/2, x, y);

return path;

}

}

}

## Результаты тестов

|  |  |
| --- | --- |
| №Теста | Скриншот результата |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |

## Ссылка на Git

https://github.com/Bert-10/5\_laba