|  |  |  |
| --- | --- | --- |
| Лабораторная работа № 3-4  Разработка компонентов  аналитических и ориентированных на добычу знаний программных средств поддержки принятия решений | Студент | Сулицкий М.В |
| Группа | ИВТ-364 |
| Преподаватель |  |
| Дата выполнения |  |
| Оценка (баллы) |  |

Цель: реализовать взаимодействие между объектами, используя несложные взаимодействия реализуемые библиотеками NetTopologySuite и OpenLayers.

Задание:

|  |  |  |  |
| --- | --- | --- | --- |
| порт | Point | 5 | Создает **корабли**, посылая их в другой случайный **порт**. |
| корабль | Point | много | Движется до заданного **порта** с заданной скоростью. |

Все объекты, создаваемые в рамках задач, должны быть описаны отдельными классами и иметь отличный от стандартного стиль отображения, заданный атрибутом CustomStyle.

**Описание работы:**

В данной работе используется готовый проект, позволяющий создавать объекты на карте и управлять их поведением. Интерфейс представлен автоматически обновляемой html страницей, которая отображает карту и выводимые на нее элементы. Для отображения используется библиотека OSMLSGlobalLibrary. Все создаваемые объекты на карте создаются с помощью библиотеки NetTopologySuite. В данном случае все объекты представлены Point. Для движения объектов в заданном направлении используется вектор. Стиль отображения, задается атрибутом CustomStyle.

Код модуля

using System;

using System.Collections;

using System.Collections.Generic;

using System.Numerics;

using System.Reflection.Metadata;

using NetTopologySuite.Geometries;

using NetTopologySuite.Mathematics;

using OSMLSGlobalLibrary;

using OSMLSGlobalLibrary.Map;

using OSMLSGlobalLibrary.Modules;

namespace Ships

{

public class ShipAndPort : OSMLSModule

{

//Корабли

List<Ship> ships = new List<Ship>();

//Скорость

double shipSpeed = 5000;

//Число отправленных кораблей

int shipWork = 0;

//Массив индексов кораблей в пути

List<int> indexShipsWork = new List<int>();

//Порты

List<Port> ports = new List<Port>(5);

double[,] coords ={

{60.130917, 29.269412},

{24.343244, 61.622640},

{26.650891, -14.632386},

{35.615088, 27.851294},

{45.314053, 51.115447}};

//Время по которому порт отправляет корабль

int timeRun = 0;

protected override void Initialize()

{

//Добавление портов

for (int i = 0; i < 5; i++)

{

ports.Add(new Port(MathExtensions.LatLonToSpherMerc(coords[i, 0], coords[i, 1]), i));

MapObjects.Add(ports[i]);

}

/\* //Добавление кораблей

for (int i = 0; i < 1000; i++)

{

ships.Add(new Ship(new Coordinate(i, i), 0, new Coordinate()));

MapObjects.Add(ships[i]);

ships[i].updateWork(new Coordinate(0, 0), 0, new Coordinate());

}\*/

}

public override void Update(long elapsedMilliseconds)

{

//Отправка из портов

if (timeRun == 0)

{

foreach (Port x in ports)

{

if (ships.Count < 10000)

{

//Выбор порта

Random random = new Random();

Coordinate coordinateDestination = new Coordinate(ports[(ports.IndexOf(x) + random.Next(1, ports.Count)) % ports.Count].Coordinate);

//Назначение корабля

ships.Add(new Ship(new Coordinate(ships.Count, 0), 0, new Coordinate()));

MapObjects.Add(ships[ships.Count - 1]);

ships[ships.Count - 1].updateWork(x.Coordinate, shipSpeed, coordinateDestination);

var allPoints = MapObjects.GetAll<Point>();

//Учет корабля

indexShipsWork.Add(ships.Count - 1);

}

}

timeRun = 100;

}

else

{

timeRun--;

}

//Изменяем положение двигающихся кораблей

for (int i = 0; i < indexShipsWork.Count; i++)

{

if (ships[indexShipsWork[i]].work() == 1)

{

indexShipsWork.RemoveAt(i);

MapObjects.Remove(ships[indexShipsWork[i]]);

i--;

}

}

}

}

#region

[CustomStyle(

@"new ol.style.Style({

image: new ol.style.Circle({

opacity: 1.0,

scale: 1.0,

radius: 3,

fill: new ol.style.Fill({

color: 'rgba(255, 0, 255, 0.4)'

}),

stroke: new ol.style.Stroke({

color: 'rgba(0, 0, 0, 0.4)',

width: 1

}),

})

});

")]

//Классы

//Корабль

class Ship : Point

{

//Скорость

public double Speed { get; set; }

//Пункт назначения

public Coordinate CoordinateDestination { get; set; }

public Ship(Coordinate coordinate, double speed, Coordinate coodinateDestination) : base(coordinate)

{

Speed = speed;

CoordinateDestination = coodinateDestination;

}

//Перемещение из одного порта в другой

public int work()

{

//Вычисляем текущий вектор

Vector2D vector = new Vector2D(Coordinate, CoordinateDestination);

//Вычисляем вектор перемещения

Vector2D vectorTravel = vector.Normalize().Multiply(Speed);

//Действия корабля

if (vector.Length() < vectorTravel.Length())//Корабль прибыл

{

X = CoordinateDestination.X;

Y = CoordinateDestination.Y;

}

else if (CoordinateDestination.Equals(Coordinate))//Корабль пропадает с радара

{

//Убираем с радара

return 1;

}

else //Продолжает движение

{

//Вычисляем координату перемещения (начало вектора после перемещения)

X = X + vectorTravel.X;

Y = Y + vectorTravel.Y;

}

return 0;

}

//Обновление данных движения

public void updateWork(Coordinate coordinate, double speed, Coordinate coodinateDestination)

{

X = coordinate.X;

Y = coordinate.Y;

Speed = speed;

CoordinateDestination = coodinateDestination;

}

}

[CustomStyle(

@"new ol.style.Style({

image: new ol.style.Circle({

opacity: 1.0,

scale: 1.0,

radius: 10,

fill: new ol.style.Fill({

color: 'rgba(44, 27, 194, 1)'

}),

stroke: new ol.style.Stroke({

color: 'rgba(0, 0, 0, 0.4)',

width: 1

}),

})

});

")]

class Port : Point

{

public int id { get; }

Coordinate coordinate { get; }

public Port(Coordinate Coordinate, int Id) : base(Coordinate)

{

id = Id;

coordinate = Coordinate;

}

}

#endregion

}

Вывод: в данной работе был произведен анализ основных методов взаимодействия объектов между собой с использованием различных библиотек.