|  |  |  |
| --- | --- | --- |
| Лабораторная работа  № 3-4  «Разработка компонентов  аналитических и ориентированных на добычу знаний программных средств поддержки принятия решений» | Выполнил | Бобунов А.В. |
| Группа | ИВТ-364 |
| Преподаватель | Алешкевич А.А. |
| Дата |  |
| Оценка |  |

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

**Задание:**

Вариант 1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | дикая местность | Polygon | 1 | Постоянно генерирует **волков** и **оленей** в случайной точке себя. |
| волк | Point | много | Пытается догнать и съесть ближайших **оленей**. Не может выйти за пределы **дикой местности**. |
| олень | Point | много | Движется хаотично. Не может выйти за пределы **дикой местности**. |

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. Стиль отображения, задается атрибутом CustomStyle. Для движения объектов в заданном направлении используется вектор.

Код модуля

using System;

using NetTopologySuite.Geometries;

using OSMLSGlobalLibrary;

using OSMLSGlobalLibrary.Map;

using OSMLSGlobalLibrary.Modules;

using System.Collections.Generic;

using System.Linq;

namespace TestModules

{

static class Rand

{

private static Random rand = new Random();

public static int GenerateInRange(int min, int max) => (int)Math.Round(min - 0.5 + rand.NextDouble() \* (max - min + 1));

public static Coordinate GenerateNext((int leftX, int rightX, int downY, int upY) map) => new Coordinate(GenerateInRange(map.leftX, map.rightX), GenerateInRange(map.downY, map.upY));

}

public class VolkModule : OSMLSModule

{

public (int leftX, int rightX, int downY, int upY) map

Polygon polygon;

protected override void Initialize()

{

// Создание координат полигона.

var leftX = 5030801;

var rightX = 5120947;

var downY = 6234004;

var upY = 6298146;

map = (leftX, rightX, downY, upY);

var polygonCoordinates = new Coordinate[] {

new Coordinate(leftX, downY),

new Coordinate(leftX, upY),

new Coordinate(rightX, upY),

new Coordinate(rightX, downY),

new Coordinate(leftX, downY),

};

// Создание полигона по ранее созданным координатам.

polygon = new Polygon(new LinearRing(polygonCoordinates));

MapObjects.Add(polygon);

var countDeer = 40;

for (var i = 0; i < countDeer; i++)

{

MapObjects.Add(new Deer(Rand.GenerateNext(map), 10));

}

var countWolf = 20;

for (var i = 0; i < countWolf; i++)

{

MapObjects.Add(new Wolf(Rand.GenerateNext(map), 15));

}

}

/// <summary>

/// Вызывается постоянно

/// </summary>

/// <param name="elapsedMilliseconds">TimeNow.ElapsedMilliseconds</param>

public override void Update(long elapsedMilliseconds)

{

var deers = MapObjects.GetAll<Deer>();

foreach (var deer in deers)

deer.MoveByMap(map);

var wolfs = MapObjects.GetAll<Wolf>();

foreach (var wolf in wolfs)

{

var nearestDeer = deers.Aggregate((deer1, deer2) => wolf.distance(deer1) < wolf.distance(deer2) ? deer1 : deer2);

wolf.Move(new Coordinate(nearestDeer.X, nearestDeer.Y));

if (wolf.CanEat(nearestDeer))

{

MapObjects.Remove(nearestDeer);

deers.Remove(nearestDeer);

MapObjects.Add(new Deer(Rand.GenerateNext(map), 10));

}

}

}

}

public static class PointExtension

{

public static double distance(this Point p1, Point p2) => Math.Sqrt(Math.Pow(p1.X - p2.X, 2) + Math.Pow(p1.Y - p2.Y, 2));

public static double distance(this Point p1, Coordinate p2) => Math.Sqrt(Math.Pow(p1.X - p2.X, 2) + Math.Pow(p1.Y - p2.Y, 2));

public static void Move(this Point p, Coordinate direction, double speed)

{

double MinimumDirection(double s, double d) =>

Math.Min(speed, Math.Abs(s - d)) \* Math.Sign(d - s);

p.X += MinimumDirection(p.X, direction.X);

p.Y += MinimumDirection(p.Y, direction.Y);

}

}

/// <summary>

/// Олень.

/// </summary>

[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(224, 12, 30, 0.9)'

}),

stroke: new ol.style.Stroke({

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

width: 1

}),

})

});

")]

class Deer : Point

{

public double Speed { get; }

private Coordinate destinationPoint = null;

public Deer(Coordinate coordinate, double speed) : base(coordinate)

{

Speed = speed;

}

public void MoveByMap((int leftX, int rightX, int downY, int upY) map)

{

if (destinationPoint == null || this.distance(destinationPoint) < Speed)

{

destinationPoint = Rand.GenerateNext(map);

}

Move(destinationPoint);

}

public void Move(Coordinate direction)

{

this.Move(direction, Speed);

}

}

[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(0, 65, 106, 0.9)'

}),

stroke: new ol.style.Stroke({

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

width: 1

}),

})

});

")]

class Wolf : Point

{

public double Speed { get; }

public Wolf(Coordinate coordinate, double speed) : base(coordinate)

{

Speed = speed;

}

public void Move(Coordinate direction)

{

this.Move(direction, Speed);

}

public bool CanEat(Deer deer)

{

return this.distance(deer) < Speed;

}

}

}

**Вывод:** в результате проделанной работы был произведен анализ методов взаимодействия объектов между собой с использованием библиотек.