**Практика «Angry Birds»**

Скачайте [архив с проектом AngryBirds](https://ulearn.me/Exercise/StudentZip?courseId=BasicProgramming&slideId=9fdf2523-85d0-4180-8ef5-88ab8d6b21ff). Это простой симулятор системы прицеливания. В файле AngryBirdsTask реализуйте функцию расчета угла прицеливания, в зависимости от начальной скорости снаряда и дальности до цели. Если решения не существует, метод должен возвращать double.NaN.

Проверьте корректность своего решения, запустив проект.

Вы можете изучить устройство проекта — это будет полезно, но для выполнения этого задания это совсем не обязательно. Более того, будьте готовы к тому, что в проекте активно используются ещё не пройденные темы.

Детали:

1. Сопротивлением воздуха можно пренебречь
2. Ускорение свободного падения g = 9.8 м/с2
3. Иногда может существовать два решения: навесной траекторией и прямой наводкой. Вам нужно решение прямой наводкой, то есть с минимальным временем подлёта снаряда.
4. Освежить свои знания по физике всегда можно в википедии прочитав статью про [равноускоренное движение](http://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B2%D0%BD%D0%BE%D1%83%D1%81%D0%BA%D0%BE%D1%80%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B2%D0%B8%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5)

// Вставьте сюда финальное содержимое файла AngryBirdsTask.cs

**Содержимое файла Program.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

using TestingRoom;

namespace AngryBirds

{

internal static class Program

{

[STAThread]

private static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new TestRoom(CreateTestCases()));

}

private static IEnumerable<TestCase> CreateTestCases()

{

yield return new ArtilleryTestCase(100, 1000);

yield return new ArtilleryTestCase(10, 0);

yield return new ArtilleryTestCase(99.1, 1000);

yield return new ArtilleryTestCase(450, 20000);

yield return new ArtilleryTestCase(450, 1000);

yield return new ArtilleryTestCase(450, 200);

yield return new ArtilleryTestCase(10, 1);

yield return new ArtilleryTestCase(9, 1);

yield return new ArtilleryTestCase(8, 1);

yield return new ArtilleryTestCase(7, 1);

yield return new ArtilleryTestCase(6, 1);

yield return new ArtilleryTestCase(5, 1);

yield return new ArtilleryTestCase(4, 1);

yield return new ArtilleryTestCase(3.5, 1);

yield return new ArtilleryTestCase(3.2, 1);

yield return new ArtilleryTestCase(3.15, 1);

yield return new ArtilleryTestCase(3.14, 1);

yield return new ArtilleryTestCase(1, 1000, hasSolution:false);

}

}

public class ArtilleryTestCase : TestCase

{

private readonly double distance;

private readonly bool hasSolution;

private readonly IList<Tuple<double, double>> trajectory = new List<Tuple<double, double>>();

private readonly double v;

private double angle;

private double time;

public ArtilleryTestCase(double v, double distance, bool hasSolution = true)

: base("Artillery")

{

this.v = v;

this.distance = distance;

this.hasSolution = hasSolution;

}

protected override void InternalVisualize(TestCaseUI ui)

{

ui.Log("D = " + distance);

ui.Log("V = " + v);

// Горизонт

ui.Line(-100, 0, 100, 0, new Pen(Color.Black, 3));

// Цель

ui.Circle(50, 0, 2, new Pen(Color.Blue, 1));

if (LastException == null)

{

//Траектория

foreach (var dot in trajectory.Where((p, i) => i % 10 == 0))

ui.Dot(-50 + dot.Item1 \* 100 / distance, -dot.Item2 \* 100 / distance, Color.Red);

ui.Circle(-50, 0, 1, new Pen(Color.Black, 5));

if (trajectory.Any())

{

// Пушка

ui.Line(-50, 0, -50 + 10 \* Math.Cos(angle), -10 \* Math.Sin(angle), new Pen(Color.Black, 3));

ui.Log("Угол прицеливания: " + 180 \* angle / Math.PI + "°");

ui.Log("Высота над целью = " + trajectory.Last().Item2);

ui.Log("Время снаряда в полете = " + time);

}

}

}

protected override bool InternalRun()

{

time = 0;

trajectory.Clear();

angle = AngryBirdsTask.FindSightAngle(v, distance);

if (double.IsInfinity(angle)) return false;

if (double.IsNaN(angle)) return !hasSolution;

double x = 0;

double y = 0;

trajectory.Add(Tuple.Create(x, y));

var vx = v \* Math.Cos(angle);

var dt = distance / v / 1000;

var g = 9.8;

var vy = v \* Math.Sin(angle);

if (vx < 0.00001) return false;

while (x < distance)

{

time += dt;

vy -= g \* dt;

x += vx \* dt;

y += vy \* dt;

trajectory.Add(Tuple.Create(x, y));

}

return Math.Abs(y) <= distance / 100;

}

}

}

**Содержимое файла AngryBirdsTask.cs**

using System;

namespace AngryBirds

{

public static class AngryBirdsTask

{

// Ниже — это XML документация, её использует ваша среда разработки,

// чтобы показывать подсказки по использованию методов.

// Но писать её естественно не обязательно.

/// <param name="v">Начальная скорость</param>

/// <param name="distance">Расстояние до цели</param>

/// <returns>Угол прицеливания в радианах от 0 до Pi/2</returns>

public static double FindSightAngle(double v, double distance)

{

return Math.PI / 4;

}

}

}

**Программа:**

using System;

namespace AngryBirds

{

public static class AngryBirdsTask

{

// Ниже — это XML документация, её использует ваша среда разработки,

// чтобы показывать подсказки по использованию методов.

// Но писать её естественно не обязательно.

/// <param name="v">Начальная скорость</param>

/// <param name="distance">Расстояние до цели</param>

/// <returns>Угол прицеливания в радианах от 0 до Pi/2</returns>

public static double FindSightAngle(double v, double distance)

{

double g = 9.8;

double a = 0.5 \* Math.Asin(distance \* g / Math.Pow(v, 2));

return a;

}

}

}