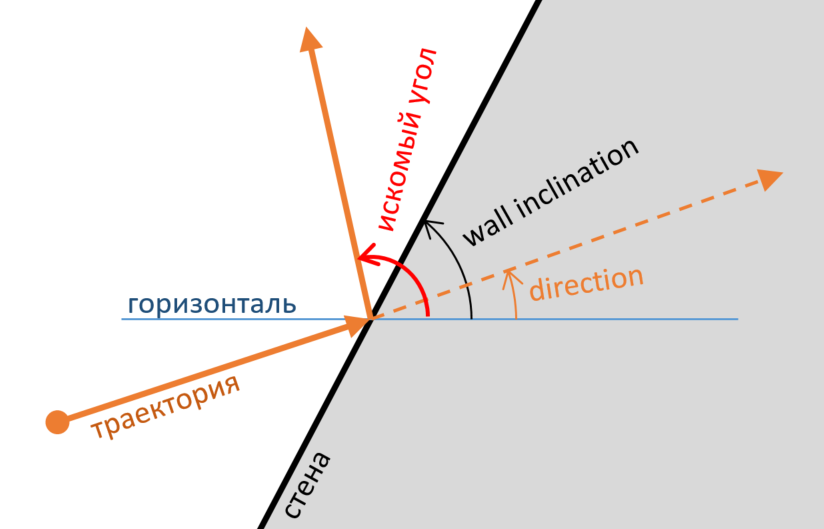
**Practice "Billiards"**

Download the archive with the Billard project.

Implement a method to calculate the angle of the ball bouncing off the wall. Consider that the angle of incidence is equal to the angle of reflection, that is, you can neglect all physical effects associated with the twisting of balls, friction of the ball against the wall, etc.

The meaning of all angles used in the problem is illustrated in the diagram:



You can check the correctness of your implementation by running the project.

You can study the design of the project - it will be useful, but for this task it is not at all necessary. Moreover, be prepared for the fact that the project actively uses topics not yet covered.

// Paste the final content of the BilliardsTask.cs file here

**Contents of the Program.cs file**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Windows.Forms;

using TestingRoom;

namespace Billiards

{

public 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 BilliardTestCase(45, 90, 135);

yield return new BilliardTestCase(10, 90, 170);

yield return new BilliardTestCase(171, 90, 9);

yield return new BilliardTestCase(90, 90, 90);

yield return new BilliardTestCase(91, 90, 89);

yield return new BilliardTestCase(90, 0, 270);

yield return new BilliardTestCase(270, 0, 90);

yield return new BilliardTestCase(-95, 0, 95);

yield return new BilliardTestCase(10, 0, 350);

yield return new BilliardTestCase(40, 0, 320);

yield return new BilliardTestCase(0, 45, 90);

yield return new BilliardTestCase(45, 45, 45);

yield return new BilliardTestCase(44, 45, 46);

yield return new BilliardTestCase(-44, -45, -46);

yield return new BilliardTestCase(44, -45, -134);

yield return new BilliardTestCase(0, 10, 20);

yield return new BilliardTestCase(0, -10, -20);

}

}

public class BilliardTestCase : TestCase

{

private readonly double expectedFinalDirection;

private readonly double initialDirection;

private readonly double wallInclination;

private double angle;

public BilliardTestCase(double initialDirection, double wallInclination, double expectedFinalDirection)

: base("")

{

this.wallInclination = wallInclination \* Math.PI / 180;

this.initialDirection = initialDirection \* Math.PI / 180;

this.expectedFinalDirection = expectedFinalDirection \* Math.PI / 180;

}

protected override void InternalVisualize(TestCaseUI ui)

{

ui.Log("Wall inclination: " + ToGradus(wallInclination));

ui.Log("Direction: " + ToGradus(initialDirection));

ui.Line(-100 \* Math.Cos(wallInclination), 100 \* Math.Sin(wallInclination), 100 \* Math.Cos(wallInclination),

-100 \* Math.Sin(wallInclination), new Pen(Color.Black, 1));

ui.Line(-50 \* Math.Cos(initialDirection), 50 \* Math.Sin(initialDirection), 0, 0, new Pen(Color.Red, 3));

ui.Line(50 \* Math.Cos(angle), -50 \* Math.Sin(angle), 0, 0,

new Pen(Color.Red, 3) {DashStyle = DashStyle.Dash});

ui.Line(50 \* Math.Cos(expectedFinalDirection), -50 \* Math.Sin(expectedFinalDirection), 0, 0,

new Pen(Color.Green, 1) {DashStyle = DashStyle.Dash});

}

protected override bool InternalRun()

{

angle = BilliardsTask.BounceWall(initialDirection, wallInclination);

var diff = angle - expectedFinalDirection;

while (diff < -Math.PI) diff += 2 \* Math.PI;

while (diff > Math.PI) diff -= 2 \* Math.PI;

return Math.Abs(diff) < 0.001;

}

private static string ToGradus(double radians)

{

return radians \* 180 / Math.PI + "°";

}

}

}

**Contents of the BilliardsTask.cs file**

using System;

namespace Billiards

{

public static class BilliardsTask

{

/// <summary>

///

/// </summary>

/// <param name="directionRadians">Угол направелния движения шара</param>

/// <param name="wallInclinationRadians">Угол</param>

/// <returns></returns>

public static double BounceWall(double directionRadians, double wallInclinationRadians)

{

//TODO

return 0.0;

}

}

}

**Code:**

using System;

namespace Billiards

{

public static class BilliardsTask

{

/// <summary>

///

/// </summary>

/// <param name="directionRadians">Угол направелния движения шара</param>

/// <param name="wallInclinationRadians">Угол</param>

/// <returns></returns>

public static double BounceWall(double directionRadians, double wallInclinationRadians)

{

double a = (2 \* wallInclinationRadians) - directionRadians;

return a;

}

}

}