Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных

систем и технологий

**КУРСОВАЯ РАБОТА**

по дисциплине «**Разработка приложений в визуальных средах**»

на тему: «**Программная модель гармонических колебаний маятника**» 1

Выполнил

студент гр.10701219 Харлап С.А

Принял

доц. Гурский Н.Н.

Минск 2021

Белорусский национальный технический университет

Кафедра программного обеспечения информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе**

**по дисциплине «**Разработка приложений в визуальных средах»

Тема: «Программная модель гармонических колебаний маятника»

**Исполнитель**: Харлап С. А.

**Студент 2 курса 10701219 группы**

**Руководитель**: доц. Гурский Н.Н.

Минск 2021

Оглавление

[ВВЕДЕНИЕ 4](#_Toc71561467)

[1. МАТЕМАТИЧЕСКАЯ ФОРМУЛИРОВКА ЗАДАЧИ 5](#_Toc71561468)

[2. ОПИСАНИЕ ПРОГРАММЫ 6](#_Toc71561469)

[2.1 Структурная схема программы 6](#_Toc71561470)

[2.2 Описание разработанного класса 6](#_Toc71561471)

[2.3 Основные возможности программы 7](#_Toc71561472)

[3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 8](#_Toc71561473)

[ЗАКЛЮЧЕНИЕ 11](#_Toc71561474)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 12](#_Toc71561475)

# ВВЕДЕНИЕ

Компьютерная модель - компьютерная программа, работающая на отдельном компьютере, суперкомпьютере или множестве взаимодействующих компьютеров (вычислительных узлов), реализующая абстрактную, то есть информационную модель некоторой системы. Компьютерные модели стали обычным инструментом численно-математического моделирования и применяются в физике, астрофизике, механике, химии, биологии, экономике, социологии, метеорологии, других науках и прикладных задачах в различных областях радиоэлектроники, машиностроения, автомобилестроения и проч. Компьютерные модели используются для получения новых знаний о моделируемом объекте или для приближенной оценки поведения систем, слишком сложных для логико-аналитического исследования.

Компьютерное моделирование является одним из эффективных методов изучения сложных систем. Компьютерные модели проще и удобнее исследовать в силу их возможности проводить так называемые "вычислительные эксперименты", которые на самом деле экспериментами не являются, так как информация берется не из физического эксперимента, физической реальности, а из модельного представления о ней, проводят в тех случаях, когда реальные эксперименты затруднены из-за финансовых или физических препятствий, или могут дать непредсказуемо опасный результат. В случае корректной логики и корректной формализации на этапе создания компьютерных моделей имеется возможность выявить основные факторы, определяющие количественные свойства изучаемого объекта-оригинала (или целого класса объектов), в частности, исследовать отклик моделируемой физической системы на изменения ее параметров и начальных условий.

Построение компьютерной модели базируется на абстрагировании от конкретной природы явлений или изучаемого объекта-оригинала и состоит из двух этапов — сначала создание качественно-логической, а затем и количественно-математической модели. Компьютерное же моделирование заключается в проведении серии "вычислительных экспериментов" на компьютере, целью которых является анализ на внутреннюю непротиворечивость модели и получение количественных данных о процессе функционирования модели.

Затем исследователь производит интерпретацию, то есть объяснение этих количественных результатов и их содержательное, то есть неформальное сопоставление с реальным поведением изучаемого объекта, а также частое и многократное последующее уточнение модели и т. д.

Целью курсовой работы является разработка приложения отображения гармонических колебаний маятника. Для работы был выбран язык программирования C# (VunForms), который даёт возможность быстрого создания приложений с графическим интерфейсом, а также позволяет реализовать концепции ООП.

# МАТЕМАТИЧЕСКАЯ ФОРМУЛИРОВКА ЗАДАЧИ

Дана система из двух маятников: маятника 1 и обращенного маятника 2 (Рисунок 1.1)

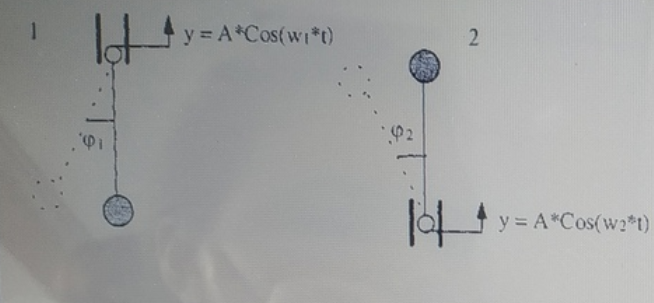


Рисунок 1.1 – Система из двух маятников

Точка подвеса первого маятника гармонически колеблется по вертикали около среднего положения по закону (1), а второго маятника по закону *(2)*

|  |  |
| --- | --- |
|  | (1) |
|  | *(2)* |

Где А – амплитуда колебаний; w1 и w2 – частоты колебаний. Пусть А = 2м. w1 = 3c-, w2=3c-. При этом каждые 10 с значение A изменяется на величину ± 0.5 поочередно.

Длина каждого из маятников равно l = 40м.

|  |  |
| --- | --- |
|  | *(3)* |
|  | *(4)* |

Первый маятник колеблется по закону *(4)*

Второй если то маятник находится в состоянии покоя, иначе по закону *(4)*

# ОПИСАНИЕ ПРОГРАММЫ

# 2.1 Структурная схема программы

Структурно программа состоит из главной формы и разработанных классов и приведена на Рисунок 2.1.

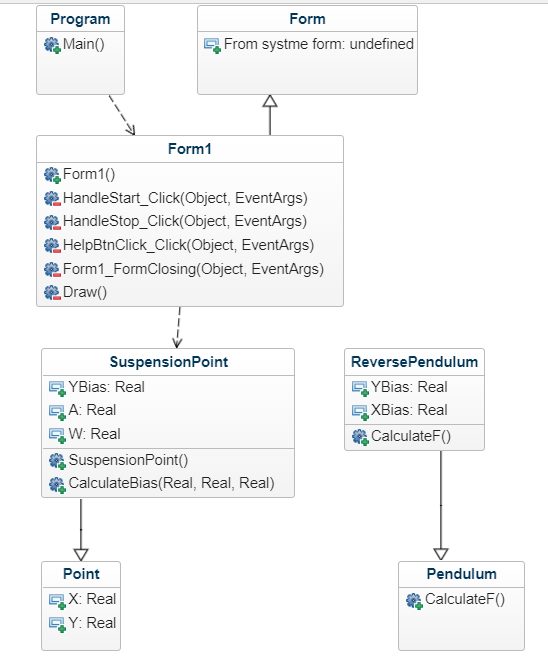
****

Рисунок 2.1 Структурная схема связей модулей программы.

# 2.2 Описание разработанного класса

В результате анализа предметной области для её описания и моделирования был разработаны.

За выполнение операции таймера на background используется класс EventLoop.

public Action<double> HandleActions – свойство отвечающее за сохранение делегата обрабатывающего события

Класс Point и наследуемы от него SuspensionPoint используемые для математического представления точки подвеса и содержащие логику позволяющую рассчитывать положение точки подвеса в зависимости от времени.

Методы и пораметры

Аналогично для расчета положения маятника в зависимости от времени существуют соответственно Pendulum class and ReversePendulum class

# 2.3 Основные возможности программы

Для взаимодействия с программой пользователю предоставлена главное окно (Form1). Функционал, который предоставляет форма:

* Вызов помощи окна помощи.
* Запуск программы
* Остановка программы

При использовании кнопки «Помощь» вызывается help-файл с руководством пользователя.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для запуска приложения необходимо вызвать файл PendulumProject.exe. После запуска появится рабочее меню приложения *Рисунок 3.1*).

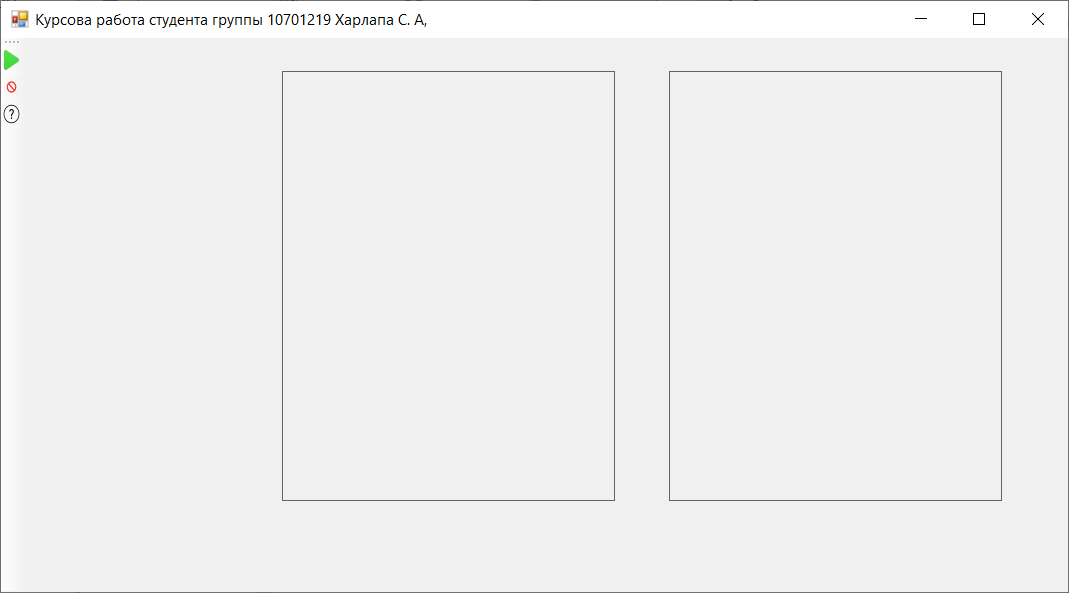


Рисунок 3.1 Главное окно программы

В левом углу мы можем видеть меню управления (*Рисунок 3.2*).

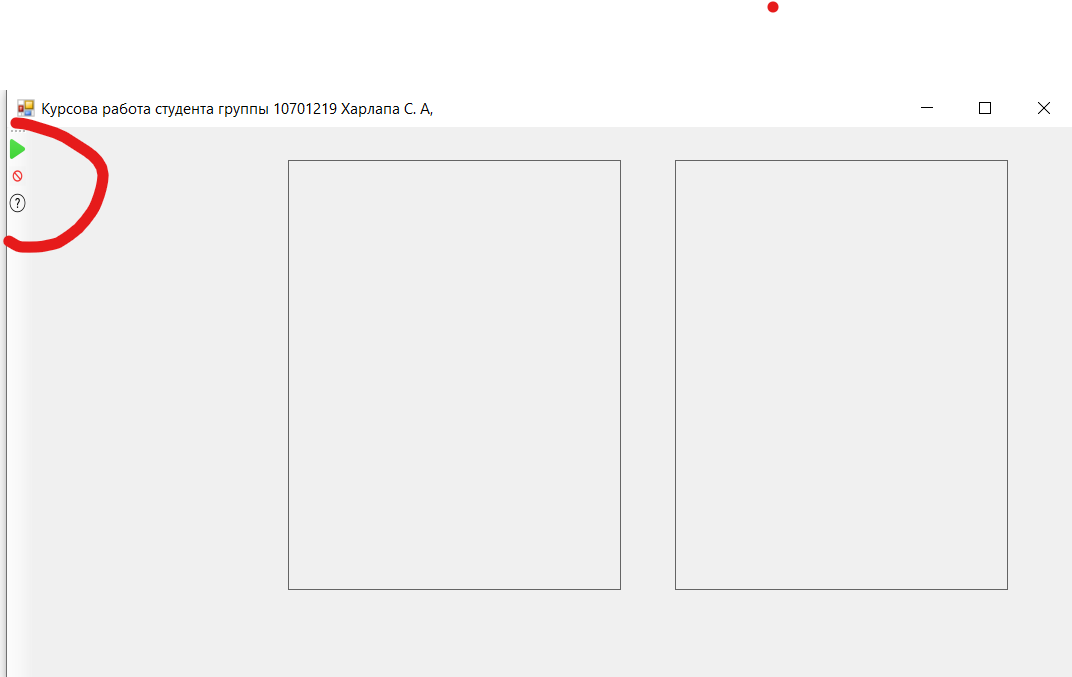


Рисунок 3.2 Меню прогрмммы

Для запуска программы необходимо нажать на кнопку старта, из меню управления (Рисунок 3.3)



Рисунок 3.3 Кнопка старт

Для остановки компьютерного моделирования необходимо нажать на кнопку стоп (*Рисунок 3.4*)



Рисунок 3002E4 Конопка стоп

Для вызова руководства пользователя необходимо нажать на кнопку help

(*Рисунок 3.5*)



Рисунок 3.5 Кнопка help

После нажатия на кнопку старт начинается моделирование (*Рисунок 3.6*)

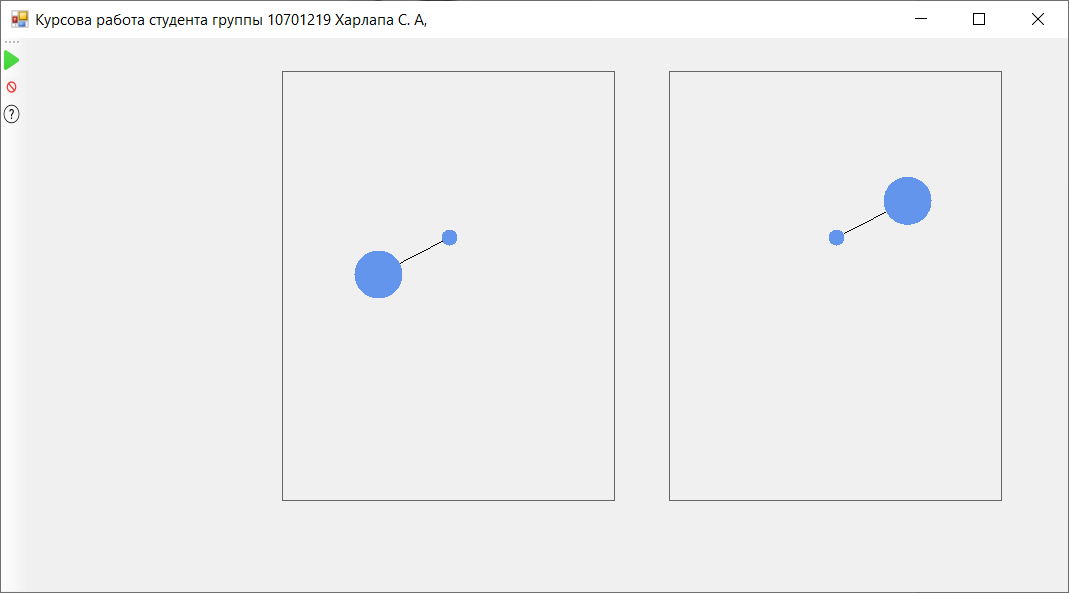


Рисунок 3.6 Пример моделирования

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы был приобретен опыт при разработке объектно-ориентированных программ, закреплен опыт, полученный при прохождении курса «Разработка приложений в визуальных средах».

Программа имеет удобный для пользователя интерфейс, различные формы вывода информации. Программа может использоваться для получения изображений из подобных треугольников. В дальнейшем можно доработать программу, чтобы она моделировала не только заранее заданные уравнения но и вводимые пользователем.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Гринчишин Я.Т. и др. Алгоритмы и программы на Бейсике: Учеб. Пособие для студентов пед. Ин-тов по физ.-мат. Спец./Я. Т. Гринчишин, В. И. Ефимов, А. Н. Ломакович. –М.: Просвещение. 1988.-160с.
2. CLRviaC#. Программирование на платформе Microsoft .NETFramework 4.5 на языке C#. 4-е изд. – СПб.; Питер, 2013 – 896с.
3. С# для чайников.: Пер. с англ. – СПб. : ООО «Диалектика», 2019. – 608 с.

**ПРИЛОЖЕНИЕ A**

**Файл Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace PendulumProject

{

public partial class Form1 : Form

{

private double speedCorrector = 1;

private EventLoop eventLoop;

private double suspensionPointRadius = 2;

private bool isStarted = false;

public Form1()

{

InitializeComponent();

}

private void HandleStart\_Click(object sender, EventArgs e)

{

if (isStarted)

{

this.ShowError("The program is already started!");

return;

}

this.isStarted = true;

this.eventLoop = new EventLoop((float)0.1);

var startTime = DateTime.Now;

var sp = new SuspensionPoint();

sp.Y = this.pendulumPictureBox.Width / 2;

sp.X = this.pendulumPictureBox.Width / 2;

var pendulum = new Pendulum();

var rePendulum = new ReversePendulum();

this.eventLoop.HandleActions += (double time) =>

{

var dt = (DateTime.Now - startTime).TotalSeconds \* this.speedCorrector;

sp.RecalculateYBias(dt);

pendulum.CalculateF(dt);

rePendulum.CalculateF(dt);

this.Draw(this.pendulumPictureBox, sp, pendulum);

this.Draw(this.rePendulumPictureBox, sp, rePendulum);

};

new Task(() =>

{

this.eventLoop.Start().Wait();

}).Start();

}

private void HandleStop\_Click(object sender, EventArgs e)

{

if (!this.isStarted)

{

this.ShowError("There isn't something to stop");

return;

}

this.eventLoop?.Stop();

this.isStarted = false;

}

private void Draw(PictureBox convars,SuspensionPoint sp, Pendulum pendulum)

{

//suspension point cercle radius

this.suspensionPointRadius = convars.ClientSize.Height / 50;

//float h = this.pendulumPictureBox.ClientSize.Height - r;

using (Graphics g = convars.CreateGraphics())

{

g.Clear(SystemColors.Control);

RectangleF rectangle = this.CreateREctangle(

sp.X,

sp.Y,

this.suspensionPointRadius

);

var x = (float)(sp.X + pendulum.XBias \* 2);

var y = (float)(sp.Y + pendulum.YBias \* 2);

g.DrawLine(

Pens.Black,

(float)sp.X,

(float)sp.Y,

x,

y

);

RectangleF rectangle2 = this.CreateREctangle(

x,

y,

this.suspensionPointRadius \* 3

);

g.FillEllipse(Brushes.CornflowerBlue, rectangle);

g.FillEllipse(Brushes.CornflowerBlue, rectangle2);

}

}

private RectangleF CreateREctangle(double x, double y, double r)

{

var rect = RectangleF.FromLTRB(

(float)(x - r),

(float)(y - r),

(float)(x + r),

(float)(y + r));

return rect;

}

private void HelpBtnClick\_Click(object sender, EventArgs e)

{

Process process = new Process();

ProcessStartInfo startInfo = new ProcessStartInfo();

process.StartInfo = startInfo;

startInfo.FileName = @"D:\Project\Pendulum\help.pdf";

process.Start();

}

private void ShowError(string errorMessage)

{

MessageBox.Show(errorMessage, "Error",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

if(this.isStarted) this.HandleStop\_Click(sender, e);

}

}

}

**Файл EventLoop.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace PendulumProject

{

public class EventLoop

{

private bool isRunning = false;

public Action<double> HandleActions { get; set; }

public float PeriodSecunds { get; set; }

public EventLoop(

float periodSecunds = 1,

Action<double> handleActions = null

)

{

this.HandleActions += handleActions;

this.PeriodSecunds = periodSecunds;

}

public async Task Start()

{

this.isRunning = true;

var time = 0;

while (this.isRunning)

{

this.HandleActions?.Invoke(time);

Thread.Sleep((int)Math.Truncate(this.PeriodSecunds \* 1000));

}

}

public void Stop()

{

this.isRunning = false;

}

}

}

**Файл Pendulum.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PendulumProject

{

public class Pendulum

{

private const double GRAVITY = 10;

public double XBias { get => Math.Sin(this.F) \* this.Lenth; }

public double YBias { get => Math.Cos(this.F) \* this.Lenth; }

public double StartAmplitude { get; set; } = 1.7;

public double A { get; set; } = 2;

public double W { get; set; }

public double Lenth { get; set; } = 40;

public double F { get; protected set; } = 0.1;

public Pendulum()

{

this.W = Math.Sqrt(GRAVITY / this.Lenth);

}

public virtual double CalculateF(double time)

{

//this.F = (GRAVITY/this.Lenth + this.A\*this.W\*this.W/this.Lenth\*Math.Cos(this.W\*time))\*this.F;

this.F = this.A \* Math.Cos(this.W \* time + this.StartAmplitude);

return this.F;

}

}

}

**Файл ReversePendulum.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PendulumProject

{

public class ReversePendulum : Pendulum

{

public override double CalculateF(double time)

{

if (this.A \* this.W < Math.Sqrt(2 \* 10 \* this.Lenth))

{

this.F = base.CalculateF(time);

this.F += 3.14;

}

else

{

this.F = 3.14;

}

return this.F;

}

}

}

**Файл Point.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PendulumProject

{

public class Point

{

public virtual double X { get; set; }

public virtual double Y { get; set; }

}

}

**Файл Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace PendulumProject

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

**Файл SuspensionPoint.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PendulumProject

{

public class SuspensionPoint : Point

{

private readonly Func<double, double, double, double> biasCalculator;

public override double Y

{

get => base.Y + this.YBias;

set => base.Y = value;

}

public double YBias { get; private set; } = 0;

public double A { get; set; }

public double W { get; set; }

public SuspensionPoint(

double A = 2,

double W = 3,

Func<double, double, double, double> biasCalculator = null

)

{

if(biasCalculator == null)

{

biasCalculator = this.CalculateBias;

}

this.A = A;

this.W = W;

this.biasCalculator = biasCalculator;

}

public double RecalculateYBias(double time)

{

this.YBias =

this.biasCalculator?.Invoke(time, this.A, this.W)

?? this.YBias;

return this.YBias;

}

protected virtual double CalculateBias(double t, double a, double w)

{

double times = t / 10;

if (times >= 1)

{

if (Math.Truncate(a) % 2 == 0)

{

a -= 0.5;

}

else

{

a += 0.5;

}

}

return a \* Math.Cos(w \* t);

}

}

}