МИНИСТЕРСТВО ОБРАЗОВАНИЯ НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное бюджетное образовательное учреждение  
 высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Физико-технический факультет  
Кафедра теоретической физики и компьютерных технологии**

**КУРСОВОЙ ПРОЕКТ**

**ВИЗУАЛИЗАЦИЯ ДВИЖЕНИЯ ФИЗИЧЕСКИХ ОБЪЕКТОВ**

Работу выполнил\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.А.Просветов

Курс 2

Направление 09.03.02 Информационные системы и технологии  
Научный руководитель   
канд. физ. - мат. наук, доцент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.В.Лежнев  
Нормокотролер, ст. преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Г.Д.Цой

Краснодар 2018 **РЕФЕРАТ** Курсовой проект 43 с., 8 рис.,9 источников, 1 прил.

Предметом исследования данной курсовой работы являются способы визуализации колебательных процессов. Основная цель курсовой: представить картинку, описывающую колебания, позволяющую увидеть их в динамике.

Для достижения этой цели решаются следующие задачи:

- анализируются способы визуализации и типы моделей, которые позволяют визуализировать процесс;

- подбирается тип модели;

- согласно типу выбранной модели, изучается теоретическая база;

- на основе теоретической базы строится математическая модель визуализации процесса;

- на основе математической модели пишется и отлаживается программа.

В результате выполнения курсового проекта была разработана программа, описывающая колебания натянутой струны, и позволяющая увидеть это в динамике.

**СОДЕРЖАНИЕ**  
Введение 4

1Стадии и этапы разработки программы 5

2Технология визуализации физических объектов 6

2.1 Методы и средства визуализации 6  
2.2 Колебания струны 7

3 Разработка технического задания 13

3.1 Назначение программы 13

3.2 Требования к программе или программному изделию 13

3.3 Требования к составу и параметрам технических средств 13

4 Разработка технического проекта 14

4.1 Структура и формы входных и выходных данных 14

4.2 Алгоритм решения задачи и структура программы 15

5 Создание программы визуализации колебаний струны как твёрдого тела 18

5.1 Динамическое изображение 18

5.1.1 Натяжение струны 18

5.1.2 Отпускаем натянутую струну 22

5.2 График зависимости амплитуды колебания натянутой струны 25

Заключение 29

Список использованных источников 30

ПриложениеАЛистинг кода файлаForm1.cs.....................................................31

**ВВЕДЕНИЕ**

Предметом исследования данной курсовой работы являются способы визуализации колебательных процессов. Основная цель курсовой: представить картинку, описывающую колебания, позволяющую увидеть их в динамике.  
Для достижения этой цели решаются следующие задачи:

- анализируются способы визуализации и типы моделей, которые позволяют визуализировать процесс;

- подбирается тип модели;

- согласно типу выбранной модели, изучается теоретическая база;

- на основе теоретической базы строится математическая модель визуализации процесса;

- на основе математической модели пишется и отлаживается программа.

Визуализация колебательных процессов проводится на примересвободных колебаний струны, описанных академиком Г.С. Ландсбергом. Под струной понимается тонкая натянутая и закреплённая с двух сторон нить, которая может свободно изгибаться.

Для достижения поставленной цели была написана программа на базе платформы.NETFramework версии 4.6 с использованием среды разработки VisualStudio 2013. Т.к. языком программирования был выбран C#, соответственно, использовался компонент VisualC# среды разработки VisualStudio [1].

Вся работа велась согласно ГОСТ 19.102-77 ЕСПД. Стадии разработки и ГОСТ 19.201-78 ЕСПД. Техническое задание. Требования к содержанию и оформлению.

Программа, созданная в процессе работы над курсовой работой, может быть применена при изучении колебаний струны для наглядной демонстрации таких колебаний.

**1 Стадии и этапы разработки программы**

Согласно ГОСТ 19.102-77 «Единая система программной документации (ЕСПД). Стадии разработки» разработаны следующие этапы работы над программой(таблица1).  
  
Таблица 1 – Процесс разработки программы

|  |  |  |
| --- | --- | --- |
| Стадии  Разработки | Этапы работ | Содержание работ |
| 1 Техническое задание | Научно-исследовательские работы | Определение структуры входных и выходных данных. |
| Предварительный выбор методов решения задач |
| Обоснование принципиальной возможности решения поставленной задачи |
| Разработка технического задания | Определение стадий и этапов разработки программы и документации на нее |
| Выбор языков программирования |
| Определение необходимости проведения научно-исследовательских работ на последующих стадиях |
| 2 Технический проект | Разработка технического проекта | Разработка структуры входных и выходных данных |
| Разработка алгоритма решения задачи |
| Определение формы представления входных и выходных данных. |
| Разработка структуры программы |
| 3 Рабочий проект | Разработка программы | Программирование и отладка программы |
| Разработка программной документации | Разработка программных документов |

**2Технология визуализации технических объектов**

## **2.1 Методы и средства визуализации**

Визуализация – это общее название приёмов представления числовой информации или физического явления в виде, удобном для зрительного наблюдения и анализа. Примерами визуализации являются рисунки, таблицы, графики, диаграммы, структурные схемы, карты, содержащие в себе характеристики объектов. Для показа динамических процессов применяетсяанимация, т.е. идущие одно за другим статичные изображения, которые в конечном итоге создают представления непрерывной смены кадров [2].  
Любая визуализация выполняет определённые функции и служит для достижения определённых целей. Она способствует наиболее полному восприятию необходимой информации, поддержке высокого уровня информированности, повышает возможности контроля и самоконтроля.  
Есть многочисленные исследования, подтверждающие, что:  
 - 90% информации человек воспринимает через зрение;  
- 70% сенсорных рецепторов находятся в глазах;  
- около половины нейронов головного мозга человека задействованы в   
обработке визуальной информации;  
- на 19% меньше при работе с визуальными данными используется когнитивная функция мозга, отвечающая за обработку и анализ информации;  
- на 17% выше производительность человека, работающего с  
визуальной информацией;  
- на 4,5% лучше вспоминаются подробные детали визуальной информации;  
- 10% человек запоминает из услышанного, 20% — из прочитанного, и 80% — из увиденного и сделанного;  
- на 323% лучше человек выполняет инструкцию, если она содержит иллюстрации;  
- 60000 раз быстрее воспринимается визуальная информация по сравнению с текстовой;  
 - 10% человек запоминает из услышанного, 20% — из прочитанного, и 80% — из увиденного и сделанного;  
 - на 33% лучше человек выполняет инструкцию, если она содержит иллюстрации.

Цели визуализации — это реализация основной идеи информации, выявления отношений в информации, показа распределения данных.

Перед началом создания визуализации надо сначала определить её цель, определить тип данных, которые необходимо представить, и на основе вышесказанного определить вид и тип модели, которая будет выступать в качестве визуализации (рисунок 1).

Определение цели визуализации

Определение типа данных

Определение типа подходящей модели

Построение модели визуализации

Рисунок 1­­– Алгоритм создания модели визуализации

**2.2Колебания струны**

Пусть струна закреплена с двух сторон итуго натянута. Если вывести струну из положения равновесия, например,оттянуть посередине со   
  
  
смещением Δyочень малым по сравнению с длиной струны, то струна начнёт совершать свободные колебания.Будем предполагать, что все точки струны движутся перпендикулярно её положению равновесия и находятся в одной плоскости.

Тогда её частота определяется по формуле

*ν*= , (1)

где *ν* – частота колебаний струны;

F – сила, с которой действовали на струну перпендикулярно её оси;

l – длина струны;

S – площадь поперечного сечения струны;

π =3.14;

ρ – плотность материала, из которых струна изготовлена.

Длина струны и её площадь поперечного сечения зависят от поставленного эксперимента и в программе задаётся вручную.Плотность материала, из которого изготовлена струна также необходимо будет задать вручную. Значение плотности можно взять из справочников. В таблице 2 приведены некоторые значения плотности веществ, из которых может быть изготовлена струна.  
Таблица 2– Плотность твёрдых тел (при 20 0С)

|  |  |
| --- | --- |
| Вещество | Плотность,ρ, кг/м3 |
| Алюминий | 2700 |
| Латунь | 8400-8700 |
| Медь | 8930 |
| Никель | 8900 |
| Серебро | 10500 |
| Сталь | 7700-7900 |

В работе рассматриваются только малые деформации, когда смещение Δyмало по отношению к длине струны *l* (рисунок 2), поэтому для расчёта приложенной к струне силы применим закон Гука.

Δy

*l*

Рисунок 2 –Натянутая струна

Закон Гука для сдвиговых деформаций:

(2)  
  
гдеτ – напряжение сдвига;

G – модуль сдвига;

γ – деформация сдвига (мера искажения формы).

Модуль сдвига G – является постоянной величиной для материала, из которого изготовлена струна. В таблице 3 приведены значения модуля сдвига для некоторых веществ.  
  
Таблица 3 – Модули сдвига

|  |  |
| --- | --- |
| Вещество | Модуль сдвига,G, Гпа |
| Алюминий | 25-26 |
| Латунь | 34-36 |
| Медь | 48 |
| Никель | 79 |
| Серебро | 30,3 |
| Сталь | 80 |

Деформацию сдвига γ при малых деформациях можно посчитать по формуле:

(3)

Подставляя формулу (3) в закон Гука (2), получим

(4)

Силу,приложенную к струне, можно рассчитать по формуле:

, (5)  
  
  
гдеτ – напряжение сдвига;

S – площадь поперечного сечения струны.

После подстановки формулы (5) в (1) получаем формулу для расчёта частоты свободных колебаний струны:

*ν*= (6)

Зная, что площадь поперечного сечения струны вычисляется по формуле , получаем окончательную формулу для нахождения частоты колебаний струны:

(7)

Из всего вышеописанного следует, что для того, чтобы можно было визуализировать колебание струны, необходимо знать следующие параметры:  
-длину струны *l;*  
-диаметр поперечного сечения струны d;  
 - плотность материала, из которого изготовлена струна ρ;  
 - величину сдвига струны Δy;  
 - модуль сдвига G.

Два из перечисленных параметров являются табличными величинами – это плотность материала ρ и модуль сдвигаG, остальные параметры задаются экспериментально [3].

С течением времени амплитуда колебаний струны будет уменьшаться, т.е. струна будет совершать затухающие колебания. Значение амплитуды на момент времени и т.д. образует убывающую геометрическую прогрессию, знаменатель которой равен – коэффициент затухания, характеризующий скорость затухания, и является величиной, обратной времени, за которое амплитуда колебаний уменьшается в *е* раз. Коэффициент затухания зависит от структуры, параметров и свойств среды, длины и типов волн и др. Он обусловлен потерей энергии колеблющийся струны [4].

При не слишком значительном рассеянии механической энергии, а именно этот случай рассматривается в данной работе, циклическая частота примерно равна, а движение струны описывается уравнением:

, (8)

гдеА – амплитуда затухающих колебаний;

–зависит от выбора начального времени отсчёта;

– коэффициент затухания, зависит от условий эксперимента.

График такого колебания показан на рисунке 3.

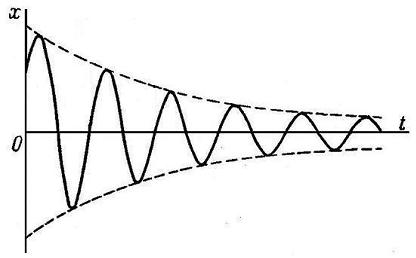


Рисунок 3 – Затухающие колебания струны

**3 Разработка технического задания** **3.1Назначение программы**

Функциональным назначением программы является предоставление пользователю возможностиувидеть в замедленном режиме процесс колебания струны и построить график зависимости амплитуды колебаний от времени.   
 Программа может использоваться для изучения колебаний натянутой струны в зависимости от материала, из которого изготовлена струна, её поперечного сечения и длины. При изменении хотя бы одного из перечисленных параметров, частота колебаний струны согласно формуле (7) изменяется, что и должно быть отображено на графике [5].

## **3.2 Требования к программе или программному изделию**

Программа должна обеспечивать взаимодействие с пользователем ([оператором](javascript:void(20435))) посредством [графического пользовательского интерфейса](http://tdocs.su/1808)*.*

Программа должна визуально показывать изменения частоты колебаний струны в зависимости от параметров самой струны. В качестве интегрированной среды разработки программы должна быть VisualStudio 2013 с использованием платформы .NETFramework версии 4.6. языком программирования выбран C# [2].

## **3.3 Требования к составу и параметрам технических средств**

Программа рассчитана на работу в операционных системах семейства Windows, начиная от Windows 7. Соответственно, технические требования к оборудованию соответствуют тем, которые предъявляют к операционные системы семейства Windows, начиная от Windows7.  
 **4 Разработка технического проекта** **4.1 Структура и формы входных и выходных данных**

Для визуализации колебания струны необходимо ввести с клавиатуры следующие входные данные:  
 *- G –* модуль сдвига, постоянная величина для материала, из которого  
выполнена струна. Её значение можно найти в справочниках, значения для некоторых веществ приведены в таблице 2. Измеряется в *Па*;  
 *- l* – длина струны, измеряется в *м*;  
 *- ρ* – плотность материала, из которого выполнена струна. Её значение можно найти в справочниках, значения для некоторых веществ приведены в таблице 1. Измеряется в*кг/м3*;  
 *- Δy* – начальная деформация струны, зависит от условий эксперимента, измеряется в *мм*;  
 *- δ*– коэффициент затухания, зависит от условий эксперимента ихарактеризующий скорость затухания колебаний струны, измеряется в с-1;  
 *- d*– диаметр поперечного сечения струны, измеряется в *мм* [6].

Интерфейс ввода данных, необходимых для начала работы программы, показан на рисунке 4.

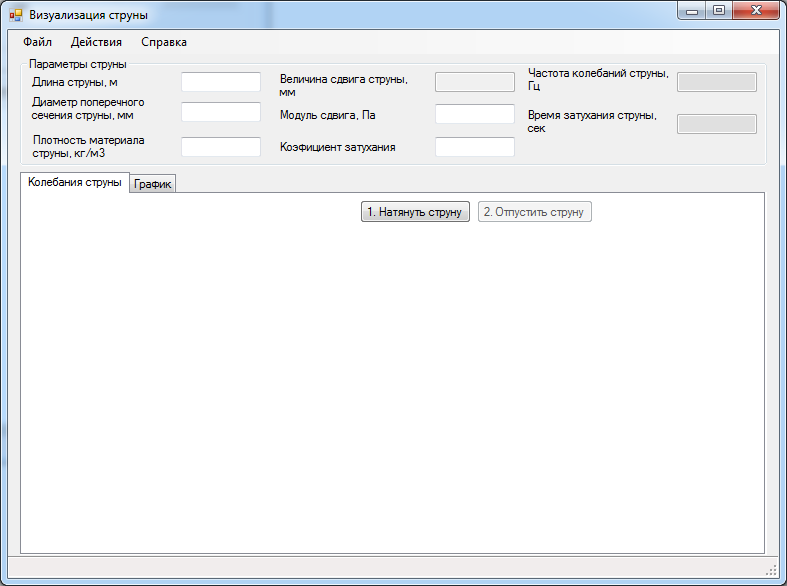


Рисунок 4 – Интерфейс ввода данных

## **4.2 Алгоритм решения задачи и структура программы**

После ввода данных в соответствующие поля, программа строит динамическую картину колебаний струны или график зависимости амплитуды колебаний от времени. Кроме динамической картины колебаний струны и графика на выводе можно увидеть расчётное время затухания колебаний и частоту колебаний в Гц.

Общая структура программы представлена схемой на рисунке 5.

Ввод исходных параметров

*G, l, ρ, d, Δy, δ*

Расчёт частоты колебаний струны,

формула (6)

Построение динамического изображения

Построение графика

Натянуть струну

Отпустить струну

Рисунок 5 –Общая структура программы визуализации   
колебаний струны

Более подробно блок-схема динамического изображения представлена на рисунках 6 (натянуть струну) и 7 (отпустить струну).При нажатии кнопки «2. Отпустить струну» происходит вызов процедуры .

Вызов процедуры

отрисовки системы

координат

Все параметрывнесены

Да

Нет

Нажатие кнопки

«1. Натянуть струну»

Вызов процедуры

отрисовки натянутой струны

Расчёт частоты колебаний струны

Сообщение о невнесённых начальных параметрах

Рисунок 6 –Динамическое изображение колебаний струны.  
 Натянуть струну

Метод CalculateAmhlitude ()

Задание начальных значений *A=Δy, t=0, такт=0, sumy=0,интервал таймера = 0.1 с.*

Запуск таймера.Таймер по заданному интервалу запускает метод

CalculateAmhlitude ()

Нажатие кнопки «2. Отпустить струну»

(вызов процедуры Vibration)

Расчёт текущего сдвига по оси *у* точки середины струны, мм

Расчёт амплитуды

А>0

sumy=0

Вывод результатов расчёта частоты и времени затухания колебаний

Нет

Да

Вызов таймера

sumy=sumy+pixy

Да

|sumy|>1

Нет

|pixy|>1

Да

SystemCoordinate()

Да

Рисуем струну

sumy=pixy-целое(pixy)

Нет

sumy=sumy+pixy

Нет

SystemCoordinate()

Рисуем струну

sumy=pixy-целое(pixy)

Рисунок 7 –Динамическое изображение колебаний струны

# **5 Создание программы визуализации колебаний струны как твёрдого тела** **5.1 Динамическое изображение**

Программа предполагает, что пользователь сначала натянет струну, для этого он нажимает кнопку «1. Натянуть струну». А затем отпускает струну, нажав на кнопку «2. Отпустить струну». После этого на экране появится анимация колебаний струны. Отображаются только те положения струны, которые наблюдаются через каждые 0,1 секунды. Этот интервал времени подобран эмпирически [7].

### **5.1.1 Натяжение струны**

Перед прорисовкой натянутой струны программа проверяет корректность заполнения всех исходных данных. В случае, если данные не введены, выводится об этом сообщение.

// IsNullOrWhiteSpace - указывает, является ли указанная строка значением null, пустой строкой или строкой, состоящей только из пробельных символов.

if(!String.IsNullOrWhiteSpace(textBox1.Text) && !String.IsNullOrWhiteSpace(textBox2.Text) && !String.IsNullOrWhiteSpace(textBox3.Text) && !String.IsNullOrWhiteSpace(textBox4.Text) && !String.IsNullOrWhiteSpace(textBox5.Text))

{

button1.Enabled = true;

SystemCoordinate();

TautString();

CalculationOscillationFrequency();

}

else

MessageBox.Show("Введеныневсезначенияпараметровструны. Пожалуйста, заполните все поля параметров струны.", "Не все поля параметров струны заполнены.", MessageBoxButtons.OK);

Для отрисовки системы координат создана процедураSystemCoordinate(), которая рисует деления, подписи к ним, единицы измерения и сами оси координат:

///<summary>

/// Метод рисования системы координат

///</summary>

internalvoidSystemCoordinate()

{

Graphicsgraph;

// Указываем pictureBox1 в качестве области рисования

graph = pictureBox1.CreateGraphics();

// Очистимкартинку

graph.Clear(Color.White);

// перо

PenblackPen = newPen(Color.Black, 2);

// Масштабированиепопиксельное

graph.PageUnit = GraphicsUnit.Pixel;

// ->Ось X

// Подпишемось X

Point point1 = newPoint(0, pictureBox1.Height - 18); Point point2 = newPoint(0, pictureBox1.Height - 22); Point point3 = newPoint(0, pictureBox1.Height - 21);

Double j = 0.0;

Fonttextfont = newFont("arial", 10);

SolidBrushbrushtext = newSolidBrush(Color.Black);

for (inti = 28; i<= pictureBox1.Width; i += 60)

{

point1.X = i; point2.X = i; point3.X = Convert.ToInt32(i \* InternalVariables.scale);

// Рисуемделение

graph.DrawLine(blackPen, point1, point2);

// Подписываемзначение

graph.DrawString(Math.Round(j, 2).ToString(), textfont, brushtext, point3);

j += InternalVariables.length/11.0;

}

point3.X += 15;

// Выведем единицу измерения оси Х (м)

graph.DrawString("м", textfont, brushtext, point3);

// Координатыоси X

point1.X = 0; point1.Y = pictureBox1.Height - 20;

point2.X = pictureBox1.Width; point2.Y = pictureBox1.Height - 20;

// Рисуемось X

graph.DrawLine(blackPen, point1, point2);

// <- ОсьX

// ->ОсьY

// Величинанатяженияструнывмм

Double tension = InternalVariables.deltaY \* 1000;

// Подпишемось Y

point1.X = 26; point2.X = 30; point3.X = 0;

j = 0.0;

for (inti = pictureBox1.Height - 20; i>= 0; i -= 100)

{

if (j != 0) // Цифра 0 уже подписана, накладывать на нее еще один 0 нет смысла.

{

point1.Y = i; point2.Y = i; point3.Y = Convert.ToInt32(i \* InternalVariables.scale);

// Рисуемделение

graph.DrawLine(blackPen, point1, point2);

// Подписываемзначение

graph.DrawString(Math.Round(j, 2).ToString(), textfont, brushtext, point3);

}

j += tension/3.0;

}

point3.X = 4; point3.Y -= 55;

// Выведем единицу измерения оси Y (мм)

graph.DrawString("мм", textfont, brushtext, point3);

// Координатыоси Y

point1.X = 28; point1.Y = 0;

point2.X = 28; point2.Y = pictureBox1.Height;

// Рисуемоcь Y

graph.DrawLine(blackPen, point1, point2);

// <- Ось Y

}

Для отрисовки натянутой струны создана процедураTautString():

///<summary>

/// Метод рисования натянутой струны

///</summary>

internalvoidTautString()

{

// Указываем pictureBox1 в качестве области рисования

Graphicsgraph = pictureBox1.CreateGraphics();

// Перо

PenredPen = newPen(Color.Red, 2);

// Масштабированиепопиксельное

graph.PageUnit = GraphicsUnit.Pixel;

// Начальнаяточкаструны

Point point1 = newPoint(28, InternalVariables.startEndPositionStringY);

// Расчитаем, сколько в пикселях надо натянуть струну

DoublepartLenght = 100;

// Точканатяженияструны

Point point2 = newPoint(28 + Convert.ToInt32(60 \* 11 / 2), InternalVariables.startEndPositionStringY + Convert.ToInt32(partLenght));

// Конечнаяточкаструны

Point point3 = newPoint(28 + 60 \* 11, InternalVariables.startEndPositionStringY);

// Рисуемструну

graph.DrawLine(redPen, point1, point2);

graph.DrawLine(redPen, point2, point3);

// перо

PenblackPen = newPen(Color.Black, 2);

blackPen.Width = 1; // Ширинапера

// Рисуем струну до натяжения

graph.DrawLine(blackPen, point1, point3);

}

Результат выполнения этой процедуры для стальной струны диаметром 1,5 мм, длиной 1 м представлен на рисунке 8

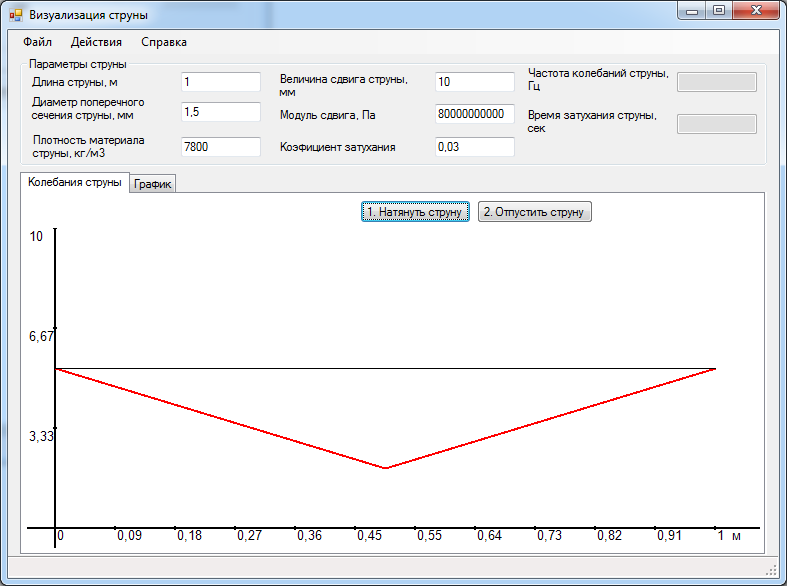


Рисунок 8 – Натяжение струны

### **5.1.2 Отпускаемнатянутуюструну**

Для того, чтобы заставить струну колебаться, нажмем на кнопку «2. Отпустить струну». После чего запускается алгоритмы визуализации колебаний струны Vibration(). В эту функцию входит перерисовка системы координат, вычисление и отрисовка текущего положения струны. Перерисовка системы координат, в свою очередь, включает команду очистки экрана (заливка элемента управления picturebox1белым цветом).

///<summary>

/// Метод рисования колебаний струны

///</summary>

internalvoidVibration()

{

// Начальнаяамплитуда

InternalVariables.A = InternalVariables.deltaY;

InternalVariables.t = 0.0;

InternalVariables.numberTick = 0;

InternalVariables.sumY = 0.0;

// Зададиминтервалтаймера

timer1.Interval = InternalVariables.intervalTime;

// Подписываемсянаобработчиксобытия Tick

timer1.Tick += newEventHandler(CalculationAmplitude);

// Запусктаймера

timer1.Start();

}

privatevoidCalculationAmplitude(object sender, EventArgs e)

{

// Увеличим счетчик количества тиков таймера на 1

InternalVariables.numberTick++;

// Время в сек, прошедшее после начала колебаний

InternalVariables.t = Convert.ToDouble(InternalVariables.numberTick \* InternalVariables.intervalTime) / 1000;

// Рассчитаем текущий сдвиг точки середины струны

Double Y = InternalVariables.A \* Math.Cos(2 \* Math.PI \* InternalVariables.frequency \* InternalVariables.t);

// Рассчитаем количество пикселей сдвига точки середины струны

DoublepixY = 100.0 \* Y / InternalVariables.deltaY;

// Рассчитаемтекущуюамплитудуструны

InternalVariables.A = InternalVariables.A \* Math.Exp(-InternalVariables.attenuation \* InternalVariables.t);

// Рисуем, если амплитуда не равна нулю

if (InternalVariables.A != 0.0)

{

if (InternalVariables.sumY == 0.0)

{

if (Math.Abs(pixY) > 1.0)

{

SystemCoordinate();

// Указываем pictureBox1 вкачествеобластирисования

Graphics graph = pictureBox1.CreateGraphics();

// Перо

PenredPen = newPen(Color.Red, 2);

// Масштабированиепопиксельное

graph.PageUnit = GraphicsUnit.Pixel;

// Начальнаяточкаструны

Point point1 = newPoint(28, InternalVariables.startEndPositionStringY);

// Точкасерединыструны

Point point2 = newPoint(28 + Convert.ToInt32(60 \* 11 / 2), InternalVariables.startEndPositionStringY + Convert.ToInt32(Math.Truncate(pixY)));

Point point3 = newPoint(28 + 60 \* 11, InternalVariables.startEndPositionStringY);

// Рисуемструну

graph.DrawLine(redPen, point1, point2);

graph.DrawLine(redPen, point2, point3);

InternalVariables.sumY = pixY - Math.Truncate(pixY);

}

else

InternalVariables.sumY = InternalVariables.sumY + pixY;

}

else

{

InternalVariables.sumY = InternalVariables.sumY + pixY;

if (Math.Abs(InternalVariables.sumY) > 1.0)

{

SystemCoordinate();

// Указываем pictureBox1 в качестве области рисования

Graphicsgraph = pictureBox1.CreateGraphics();

// Перо

PenredPen = newPen(Color.Red, 2);

// Масштабированиепопиксельное

graph.PageUnit = GraphicsUnit.Pixel;

// Начальнаяточкаструны

Point point1 = newPoint(28, InternalVariables.startEndPositionStringY);

// Точкасерединыструны

Point point2 = newPoint(28 + Convert.ToInt32(60 \* 11 / 2), InternalVariables.startEndPositionStringY + Convert.ToInt32(Math.Truncate(InternalVariables.sumY)));

// Конечнаяточкаструны

Point point3 = newPoint(28 + 60 \* 11, InternalVariables.startEndPositionStringY);

// Рисуемструну

graph.DrawLine(redPen, point1, point2);

graph.DrawLine(redPen, point2, point3);

InternalVariables.sumY = InternalVariables.sumY - Math.Truncate(InternalVariables.sumY);

}

}

}

else

{

// Выводрезультатавычислений

textBox6.Text = InternalVariables.frequency.ToString();

textBox8.Text = InternalVariables.t.ToString();

timer1.Stop(); // Остановка таймера

}

}

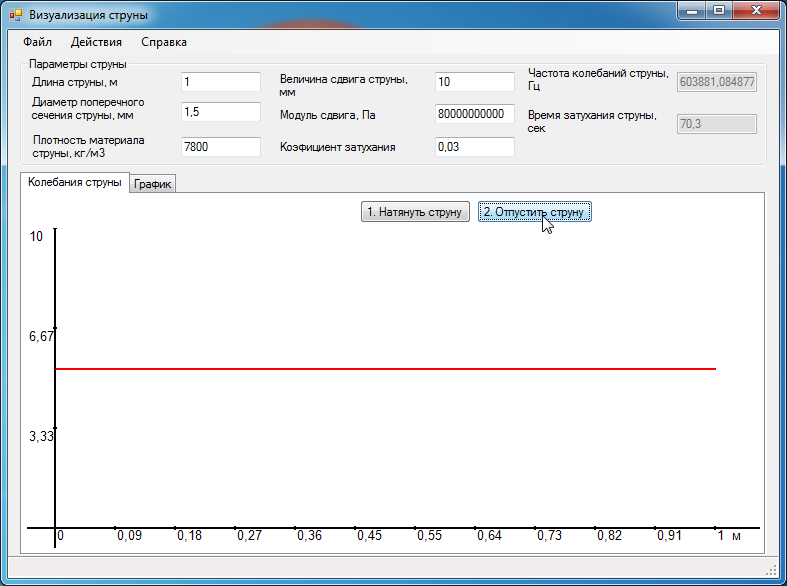
Перерисовка рисунка осуществляется через каждые 0,1 с. Некоторые положения колебаний струны показаны на рис.8. После полного затухания колебаний струны в полях «Частота колебаний струны, Гц» и «Время затухания струны, сек» будут выведены соответствующие параметры (рисунок9)  
  


Рисунок 9 – Некоторые положения колебаний струны

**5.2 Графикзависимости амплитуды колебания натянутой струны**

Построить график зависимости амплитуды колебаний струны от времени можно будет только после выполнения программы показа динамической картинки, т.е. после выполнения всех действий на вкладке «Колебания струны». Это объясняется тем, что для построения графика необходимо знать частоту колебаний и время затухания колебаний. Оба эти параметра вычисляются при построении динамического изображения колебаний струны [8].

Для создания графика сначала отрисуем систему координат. Для этого используется процедура SystemCoordinate2().

///<summary>

///Метод

///</summary>

internalvoid SystemCoordinate2()

{

// Указываем pictureBox2s в качестве области рисования

using (Graphics graph2 = Graphics.FromImage(pictureBox2.Image))

{

// Очистимкартинку

graph2.Clear(Color.White);

// перо

Pen blackPen2 = newPen(Color.Black, 2);

// Масштабированиепопиксельное

graph2.PageUnit = GraphicsUnit.Pixel;

// ->Ось X

// Подпишемось X

Int32HeightCenterY = Convert.ToInt32(pictureBox2.Height / 2); // Позицияточкисерединыпо Y

Point point1 = newPoint(0, HeightCenterY - 2); Point point2 = newPoint(0, HeightCenterY + 2); Point point3 = newPoint(0, HeightCenterY + 1);

Double j = 0.0;

Fonttextfont = newFont("arial", 10);

SolidBrushbrushtext = newSolidBrush(Color.Black);

for (inti = 28; i<= pictureBox2.Width - 40; i += 75)

{

point1.X = i; point2.X = i; point3.X = Convert.ToInt32(i \* InternalVariables.scale);

// Рисуемделение

graph2.DrawLine(blackPen2, point1, point2);

// Подписываемзначение

graph2.DrawString(Math.Round(j, 2).ToString(), textfont, brushtext, point3);

j += InternalVariables.t / 10.0;

}

point3.X += 45;

// Выведем единицу измерения оси Х (t, сек)

graph2.DrawString("сек", textfont, brushtext, point3);

// Координатыоси X

point1.X = 0; point1.Y = HeightCenterY;

point2.X = pictureBox2.Width; point2.Y = HeightCenterY;

// Рисуемось X

graph2.DrawLine(blackPen2, point1, point2);

// <- Ось X

// -> Ось Y

// Величина натяжения струны в мм

Double tension = InternalVariables.deltaY \* 1000;

// Подпишемось Y

point1.X = 26; point2.X = 30; point3.X = 0;

j = 0.0; Int32 step = Convert.ToInt32((pictureBox2.Height - 40) / 4);

for (inti = HeightCenterY; i>= 20; i -= step)

{

if (j != 0) // Цифра 0 уже подписана, накладывать на нее еще один 0 нет смысла.

{

point1.Y = i; point2.Y = i; point3.Y = Convert.ToInt32(i \* InternalVariables.scale);

// Рисуемделение

graph2.DrawLine(blackPen2, point1, point2);

// Подписываемзначение

graph2.DrawString(Math.Round(j, 2).ToString(), textfont, brushtext, point3);

}

j += tension / 2.0;

}

j = 0.0;

for (inti = HeightCenterY; i<= pictureBox2.Height - 20; i += step)

{

if (j != 0) // Цифра 0 уже подписана, накладывать на нее еще один 0 нет смысла.

{

point1.Y = i; point2.Y = i; point3.Y = Convert.ToInt32(i \* InternalVariables.scale);

// Рисуемделение

graph2.DrawLine(blackPen2, point1, point2);

// Подписываемзначение

graph2.DrawString(Math.Round(j, 2).ToString(), textfont, brushtext, point3);

}

j -= tension / 2.0;

}

point3.X = 4; point3.Y = 7;

point1.X = 28; point1.Y = 0;

point2.X = 28; point2.Y = pictureBox2.Height;

// Рисуемоcь Y

graph2.DrawLine(blackPen2, point1, point2);

// <- Ось Y

}

}

После отрисовки системы координат, начинает отрисовывать в цикле непосредственного график изменения амплитуды колебания от времени. Для этого сначала рассчитывается амплитуда по формуле (8), затем рассчитанное значение переводится в пиксели, и строится линия, соединяющая предыдущую точку с текущей. Начальная точка имеет координаты (0; ∆y), где ∆y – это величина сдвига амплитуды, задаваемая пользователем. Как и при создании динамического изображения колебаний струны, интервал времени для расчёта значений амплитуды выбран равным 0,1 с.

///<summary>

/// Метод рисования графика колебаний после окончания колебаний струны.

///</summary>

internalvoidDiagram()

{

pictureBox2.Image = newBitmap(pictureBox2.Width, pictureBox2.Height);

SystemCoordinate2();

// Указываем pictureBox2 в качестве области рисования

using (Graphics graph = Graphics.FromImage(pictureBox2.Image))

{

// Перо

PenredPen = newPen(Color.Red, 2);

// Масштабированиепопиксельное

graph.PageUnit = GraphicsUnit.Pixel;

// Величинанатяженияструнывмм

Double tension = InternalVariables.deltaY \* 1000;

// Зададимначальнуюточку

Point p1 = newPoint(28, 20);

// Переменная для второй точки

//Double sumt = 0.0; Double sumA = 0.0;

DoubleAmpl = InternalVariables.deltaY; Double t = 0.0; Double t1 = 0.0; Double Ampl1 = 0.0;

while (Ampl> 0.0)

{

t1 = t + 0.1;

Ampl1 = Ampl \* Math.Exp(-InternalVariables.attenuation \* t1);

Double X = Ampl1 \* Math.Cos(2 \* Math.PI \* InternalVariables.frequency \* t1);

DoubleshiftX = (pictureBox2.Width - 28) / InternalVariables.t;

p2.X += Convert.ToInt32(shiftX); p2.Y = Convert.ToInt32(shiftY);

// Рисуемсдвигамплитуды

graph.DrawLine(redPen, p1, p2);

p1 = p2; t = t1; Ampl = Ampl1;

}

}

}

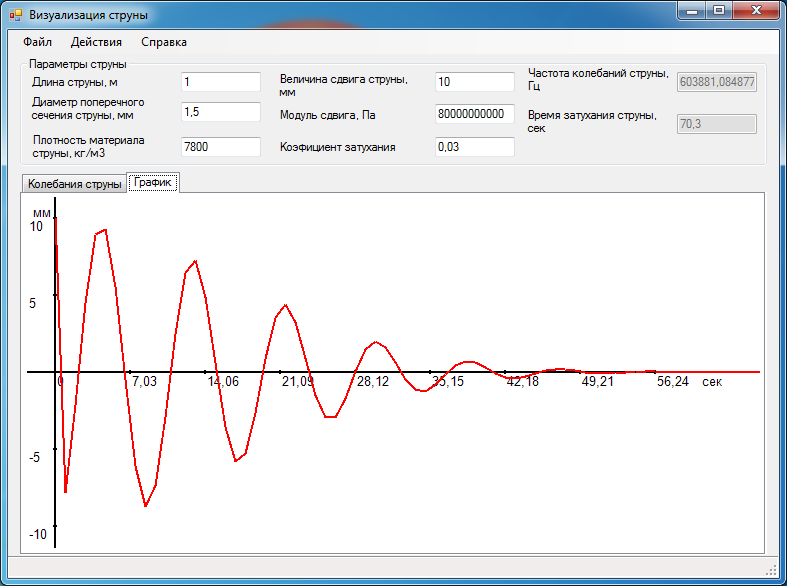
В результате выполнения этого кода для стальной струны диаметров 1,5 мм, длиной 1 м и начальным отклонением струны от положения равновесия получился график, показанный на рисунке10.  
  


Рисунок 10 – График колебаний стальной струны

**ЗАКЛЮЧЕНИЕ**  
 В ходе выполнения работы были выполнены задачи: 1 Созданы визуальные примеры колебаний струны, представленные в видеграфика.  
 2 Изучена литература по теме колебание струны.  
 3 Проанализированы виды колебаний.  
 4 Создано полноценное приложение с пользовательским интерфейсом. 5 Описание алгоритма программным способом реализовано при помощи языка С#.  
 Область применения алгоритма достаточно обширна, так как сам алгоритм представляет собой один из способов обработки информации.  
 Работа с различного рода данными является очень актуальной.  Физико-математическое моделирование осуществлялось путем полного векторного анализа поляризационных эффектов в трёхмерных и двумерных интегрально-оптических волноводах и волноводных структурах общего вида.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 Лафоре Р. Структуры данных и алгоритмы в Java / Р.Лафоре. – Спб.: Питер, 2013. – 702 с.

2 Фофанов О. Б. Алгоритмы и структуры данных / О.Б.Фофанов. –Томск: Томский политехнический университет, 2014.– 123 с.

3 ПышкинЕ. В. Структуры данных и алгоритмы: реализация на С/С++ / Е.В.Пышкин. – Спб.: ФТК СПбГПУ, 2009. – 200 с.

4 Шлее М. Qt 5.3. Профессиональное программирование на C++ / М.Шлее. – Спб.: БХВ – Питербугр,2015. – 928 с.

5 ЛиппманС. Язык программирования C++ / С.Липпман. – Спб.: Вильямс,2014. – 1120 с.

6 Дейтел Х.М. Дейтел П. Дж. Как программировать на C++/Х.М. Дейтел, П.Дж. Дейтел. Москва :Бином, 2008. – 1456 с.

7 Васильев А. Н. Самоучитель С++ с задачами и примерами / А. Н. Васильев. – Спб.: Наука и Техника, 2016. – 480 с.

8 Доусон М. Изучаем С++/ М. Доусон. – Спб.: Питер,2016. – 352 с.

**ПРИЛОЖЕНИЕ А  
  
Листинг кода файла Form1.cs**

usingSystem;

usingSystem.Collections.Generic;

usingSystem.ComponentModel;

usingSystem.Data;

usingSystem.Drawing;

usingSystem.Linq;

usingSystem.Text;

usingSystem.Threading.Tasks;

usingSystem.Windows.Forms;

namespaceStringVibration

{

publicpartialclassForm1 :Form

{

public Form1()

{

InitializeComponent();

}

classInternalVariables

{

// -->Переменные, значениякоторыхвводятсясклавиатуры ---

///<summary>

///Длинаструны

///</summary>

staticinternalDouble length = 0.0;

///<summary>

/// Диаметр поперечного сечения струны

///</summary>

staticinternalDouble d = 0.0;

///<summary>

///Коэффициентзатухания

///</summary>

staticinternalDouble attenuation = 0.0;

///<summary>

///Плотностьматериаластруны

///</summary>

staticinternalDoublero = 0.0;

///<summary>

///Величинасдвигаструны

///</summary>

staticinternalDoubledeltaY = 0.0;

///<summary>

///Модульсдвига

///</summary>

staticinternalDoubleG = 0.0;

// <-- Переменные, значения которых вводятся с клавиатуры ---

///<summary>

///Масштаб

///</summary>

staticinternalInt32 scale = 1; // задаетмасштаб

///<summary>

/// Точки опоры струны

///</summary>

staticinternalInt32startEndPositionStringY = 70 \* 2; // НачальнаяпозицияструныпоосиY

///<summary>

///Частотаколебанийструны

///</summary>

staticinternalDoublefrequency = 0.0;

///<summary>

/// Интервал времени, когда хотим перерисовать струну

///</summary>

staticinternalInt32intervalTime = 100; // 100 мс = 0,1 сек

///<summary>

///Амплитуда

///</summary>

staticinternalDouble A = 0.0;

///<summary>

///Количествотиковтаймера

///</summary>

staticinternalInt32numberTick = 0;

///<summary>

///Времязатуханияколебаний

///</summary>

staticinternalDoublet = 0.0;

///<summary>

/// Сумма сдвигов середины струны во время колебаний.

///</summary>

staticinternalDoublesumY = 0.0;

///<summary>

///Списокамплитуд

///</summary>

staticinternalList<Double>listAmplitude = newList<Double>();

///<summary>

/// Начальная точка (используется для вывода графика)

///</summary>

staticinternalPoint p = newPoint(0, 0);

}

///<summary>

/// Метод рисования системы координат

///</summary>

internalvoidSystemCoordinate()

{

Graphicsgraph;

// Указываем pictureBox1 в качестве области рисования

graph = pictureBox1.CreateGraphics();

// Очистимкартинку

graph.Clear(Color.White);

// перо

PenblackPen = newPen(Color.Black, 2);

// Масштабированиепопиксельное

graph.PageUnit = GraphicsUnit.Pixel;

// ->Ось X

// Подпишемось X

Point point1 = newPoint(0, pictureBox1.Height - 18); Point point2 = newPoint(0, pictureBox1.Height - 22); Point point3 = newPoint(0, pictureBox1.Height - 21);

Double j = 0.0;

Fonttextfont = newFont("arial", 10);

SolidBrushbrushtext = newSolidBrush(Color.Black);

for (inti = 28; i<= pictureBox1.Width; i += 60)

{

point1.X = i; point2.X = i; point3.X = Convert.ToInt32(i \* InternalVariables.scale);

// Рисуемделение

graph.DrawLine(blackPen, point1, point2);

// Подписываемзначение

graph.DrawString(Math.Round(j, 2).ToString(), textfont, brushtext, point3);

j += InternalVariables.length/11.0;

}

point3.X += 15;

// Выведем единицу измерения оси Х (м)

graph.DrawString("м", textfont, brushtext, point3);

// Координатыоси X

point1.X = 0; point1.Y = pictureBox1.Height - 20;

point2.X = pictureBox1.Width; point2.Y = pictureBox1.Height - 20;

// Рисуемось X

graph.DrawLine(blackPen, point1, point2);

// <- Ось X

// -> Ось Y

// Величина натяжения струны в мм

Double tension = InternalVariables.deltaY \* 1000;

// Подпишемось Y

point1.X = 26; point2.X = 30; point3.X = 0;

j = 0.0;

for (inti = pictureBox1.Height - 20; i>= 0; i -= 100)

{

if (j != 0) // Цифра 0 уже подписана, накладывать на нее еще один 0 нет смысла.

{

point1.Y = i; point2.Y = i; point3.Y = Convert.ToInt32(i \* InternalVariables.scale);

// Рисуемделение

graph.DrawLine(blackPen, point1, point2);

// Подписываемзначение

graph.DrawString(Math.Round(j, 2).ToString(), textfont, brushtext, point3);

}

j += tension/3.0;

}

point3.X = 4; point3.Y -= 55;

// Выведем единицу измерения оси Y (мм)

graph.DrawString("мм", textfont, brushtext, point3);

// Координатыоси Y

point1.X = 28; point1.Y = 0;

point2.X = 28; point2.Y = pictureBox1.Height;

// Рисуемоcь Y

graph.DrawLine(blackPen, point1, point2);

// <- Ось Y

}

///<summary>

/// Метод рисования натянутой струны

///</summary>

internalvoidTautString()

{

// Указываем pictureBox1 в качестве области рисования

Graphicsgraph = pictureBox1.CreateGraphics();

// Перо

PenredPen = newPen(Color.Red, 2);

// Масштабированиепопиксельное

graph.PageUnit = GraphicsUnit.Pixel;

// Начальнаяточкаструны

Point point1 = newPoint(28, InternalVariables.startEndPositionStringY);

// Расчитаем, сколько в пикселях надо натянуть струну

DoublepartLenght = 100;

// Точканатяженияструны

Point point2 = newPoint(28 + Convert.ToInt32(60 \* 11 / 2), InternalVariables.startEndPositionStringY + Convert.ToInt32(partLenght));

// Конечнаяточкаструны

Point point3 = newPoint(28 + 60 \* 11, InternalVariables.startEndPositionStringY);

// Рисуемструну

graph.DrawLine(redPen, point1, point2);

graph.DrawLine(redPen, point2, point3);

// перо

PenblackPen = newPen(Color.Black, 2);

blackPen.Width = 1; // Ширинапера

// Рисуем струну до натяжения

graph.DrawLine(blackPen, point1, point3);

}

///<summary>

/// Метод рисования колебаний струны

///</summary>

internalvoidVibration()

{

// Начальнаяамплитуда

InternalVariables.A = InternalVariables.deltaY;

// Запомним начальную амплитуду в список амплитуд

InternalVariables.listAmplitude.Add(InternalVariables.A);

InternalVariables.t = 0.0;

InternalVariables.numberTick = 0;

InternalVariables.sumY = 0.0;

// Зададиминтервалтаймера

timer1.Interval = InternalVariables.intervalTime;

// Подписываемсянаобработчиксобытия Tick

timer1.Tick += newEventHandler(CalculationAmplitude);

// Запусктаймера

timer1.Start();

}

privatevoidCalculationAmplitude(object sender, EventArgs e)

{

// Увеличим счетчик количества тиков таймера на 1

InternalVariables.numberTick++;

// Время в сек, прошедшее после начала колебаний

InternalVariables.t = Convert.ToDouble(InternalVariables.numberTick \* InternalVariables.intervalTime) / 1000;

// Рассчитаем текущий сдвиг точки середины струны

Double Y = InternalVariables.A \* Math.Cos(2 \* Math.PI \* InternalVariables.frequency \* InternalVariables.t);

// Рассчитаем количество пикселей сдвига точки середины струны

DoublepixY = 100.0 \* Y / InternalVariables.deltaY;

// Рассчитаемтекущуюамплитудуструны

InternalVariables.A = InternalVariables.A \* Math.Exp(-InternalVariables.attenuation \* InternalVariables.t);

// Рисуем, если амплитуда не равна нулю

if (InternalVariables.A != 0.0)

{

if (InternalVariables.sumY == 0.0)

{

if (Math.Abs(pixY) > 1.0)

{

SystemCoordinate();

// Указываем pictureBox1 вкачествеобластирисования

Graphics graph = pictureBox1.CreateGraphics();

// Перо

PenredPen = newPen(Color.Red, 2);

// Масштабированиепопиксельное

graph.PageUnit = GraphicsUnit.Pixel;

// Начальнаяточкаструны

Point point1 = newPoint(28, InternalVariables.startEndPositionStringY);

// Точкасерединыструны

Point point2 = newPoint(28 + Convert.ToInt32(60 \* 11 / 2), InternalVariables.startEndPositionStringY + Convert.ToInt32(Math.Truncate(pixY)));

// Конечнаяточкаструны

Point point3 = newPoint(28 + 60 \* 11, InternalVariables.startEndPositionStringY);

// Рисуемструну

graph.DrawLine(redPen, point1, point2);

graph.DrawLine(redPen, point2, point3);

InternalVariables.sumY = pixY - Math.Truncate(pixY);

}

else

InternalVariables.sumY = InternalVariables.sumY + pixY;

}

else

{

InternalVariables.sumY = InternalVariables.sumY + pixY;

if (Math.Abs(InternalVariables.sumY) > 1.0)

{

SystemCoordinate();

// Указываем pictureBox1 в качестве области рисования

Graphicsgraph = pictureBox1.CreateGraphics();

// Перо

PenredPen = newPen(Color.Red, 2);

// Масштабированиепопиксельное

graph.PageUnit = GraphicsUnit.Pixel;

// Начальнаяточкаструны

Point point1 = newPoint(28, InternalVariables.startEndPositionStringY);

// Точкасерединыструны

Point point2 = newPoint(28 + Convert.ToInt32(60 \* 11 / 2), InternalVariables.startEndPositionStringY + Convert.ToInt32(Math.Truncate(InternalVariables.sumY)));

// Конечнаяточкаструны

Point point3 = newPoint(28 + 60 \* 11, InternalVariables.startEndPositionStringY);

// Рисуемструну

graph.DrawLine(redPen, point1, point2);

graph.DrawLine(redPen, point2, point3);

InternalVariables.sumY = InternalVariables.sumY - Math.Truncate(InternalVariables.sumY);

}

}

}

else

{

// Выводрезультатавычислений

textBox6.Text = InternalVariables.frequency.ToString();

textBox8.Text = InternalVariables.t.ToString();

timer1.Stop(); // Остановкатаймера

}

}

///<summary>

///Методвычислениячастоты

///</summary>

internalvoidCalculationOscillationFrequency()

{

// Вычисляемчастоту

InternalVariables.frequency = 4 / InternalVariables.d \* InternalVariables.length \* Math.Sqrt(InternalVariables.G \* InternalVariables.deltaY / (2 \* InternalVariables.length \* InternalVariables.ro));

}

///<summary>

/// Метод рисования графика колебаний после окончания колебаний струны.

///</summary>

internalvoidDiagram()

{

pictureBox2.Image = newBitmap(pictureBox2.Width, pictureBox2.Height);

SystemCoordinate2();

// Указываем pictureBox2 в качестве области рисования

using (Graphics graph = Graphics.FromImage(pictureBox2.Image))

{

// Перо

PenredPen = newPen(Color.Red, 2);

// Масштабированиепопиксельное

graph.PageUnit = GraphicsUnit.Pixel;

// Величинанатяженияструнывмм

Double tension = InternalVariables.deltaY \* 1000;

// Зададимначальнуюточку

Point p1 = newPoint(28, 20);

// Переменная для второй точки

Point p2 = newPoint(28, 20);

// Сумма времени и сумма амплитуд в пикселях

//Double sumt = 0.0; Double sumA = 0.0;

DoubleAmpl = InternalVariables.deltaY; Double t = 0.0; Double t1 = 0.0; Double Ampl1 = 0.0;

while (Ampl> 0.0)

{

t1 = t + 0.1;

Ampl1 = Ampl \* Math.Exp(-InternalVariables.attenuation \* t1);

Double X = Ampl1 \* Math.Cos(2 \* Math.PI \* InternalVariables.frequency \* t1);

// Вычислимсдвигпооси Y

DoubleshiftY = X \* (pictureBox2.Height / 2 - 20) / InternalVariables.deltaY + pictureBox2.Height / 2;

// ВычислимсдвигпоосиХ

DoubleshiftX = (pictureBox2.Width - 28) / InternalVariables.t;

p2.X += Convert.ToInt32(shiftX); p2.Y = Convert.ToInt32(shiftY);

// Рисуемсдвигамплитуды

graph.DrawLine(redPen, p1, p2);

p1 = p2; t = t1; Ampl = Ampl1;

}

}

}

///<summary>

///Метод

///</summary>

internalvoid SystemCoordinate2()

{

// Указываем pictureBox2s в качестве области рисования

using (Graphics graph2 = Graphics.FromImage(pictureBox2.Image))

{

// Очистимкартинку

graph2.Clear(Color.White);

// перо

Pen blackPen2 = newPen(Color.Black, 2);

// Масштабированиепопиксельное

graph2.PageUnit = GraphicsUnit.Pixel;

// ->Ось X

// Подпишемось X

Int32HeightCenterY = Convert.ToInt32(pictureBox2.Height / 2); // Позицияточкисерединыпо Y

Point point1 = newPoint(0, HeightCenterY - 2); Point point2 = newPoint(0, HeightCenterY + 2); Point point3 = newPoint(0, HeightCenterY + 1);

Double j = 0.0;

Fonttextfont = newFont("arial", 10);

SolidBrushbrushtext = newSolidBrush(Color.Black);

for (inti = 28; i<= pictureBox2.Width - 40; i += 75)

{

point1.X = i; point2.X = i; point3.X = Convert.ToInt32(i \* InternalVariables.scale);

// Рисуемделение

graph2.DrawLine(blackPen2, point1, point2);

// Подписываемзначение

graph2.DrawString(Math.Round(j, 2).ToString(), textfont, brushtext, point3);

j += InternalVariables.t / 10.0;

}

point3.X += 45;

// Выведем единицу измерения оси Х (t, сек)

graph2.DrawString("сек", textfont, brushtext, point3);

// Координатыоси X

point1.X = 0; point1.Y = HeightCenterY;

point2.X = pictureBox2.Width; point2.Y = HeightCenterY;

// Рисуемось X

graph2.DrawLine(blackPen2, point1, point2);

// <- Ось X

// -> Ось Y

// Величина натяжения струны в мм

Double tension = InternalVariables.deltaY \* 1000;

// Подпишемось Y

point1.X = 26; point2.X = 30; point3.X = 0;

j = 0.0; Int32 step = Convert.ToInt32((pictureBox2.Height - 40) / 4);

for (inti = HeightCenterY; i>= 20; i -= step)

{

if (j != 0) // Цифра 0 уже подписана, накладывать на нее еще один 0 нет смысла.

{

point1.Y = i; point2.Y = i; point3.Y = Convert.ToInt32(i \* InternalVariables.scale);

// Рисуемделение

graph2.DrawLine(blackPen2, point1, point2);

// Подписываемзначение

graph2.DrawString(Math.Round(j, 2).ToString(), textfont, brushtext, point3);

}

j += tension / 2.0;

}

j = 0.0;

for (inti = HeightCenterY; i<= pictureBox2.Height - 20; i += step)

{

if (j != 0) // Цифра 0 уже подписана, накладывать на нее еще один 0 нет смысла.

{

point1.Y = i; point2.Y = i; point3.Y = Convert.ToInt32(i \* InternalVariables.scale);

// Рисуемделение

graph2.DrawLine(blackPen2, point1, point2);

// Подписываемзначение

graph2.DrawString(Math.Round(j, 2).ToString(), textfont, brushtext, point3);

}

j -= tension / 2.0;

}

point3.X = 4; point3.Y = 7;

// Выведем единицу измерения оси Y (мм)

graph2.DrawString("мм", textfont, brushtext, point3);

// Координатыоси Y

point1.X = 28; point1.Y = 0;

point2.X = 28; point2.Y = pictureBox2.Height;

// Рисуемоcь Y

graph2.DrawLine(blackPen2, point1, point2);

// <- Ось Y

}

}

privatevoid button1\_Click(object sender, EventArgs e)

{

Vibration();

}

privatevoid button2\_Click(object sender, EventArgs e)

{

// IsNullOrWhiteSpace - указывает, является ли указанная строка значением null, пустой строкой или строкой, состоящей только из пробельных символов.

if(!String.IsNullOrWhiteSpace(textBox1.Text) && !String.IsNullOrWhiteSpace(textBox2.Text) && !String.IsNullOrWhiteSpace(textBox3.Text) && !String.IsNullOrWhiteSpace(textBox4.Text) && !String.IsNullOrWhiteSpace(textBox5.Text))

{

button1.Enabled = true;

SystemCoordinate();

TautString();

CalculationOscillationFrequency();

}

else

MessageBox.Show("Введеныневсезначенияпараметровструны. Пожалуйста, заполните все поля параметров струны", "Не все поля параметров струны заполнены.", MessageBoxButtons.OK);

}

privatevoid textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

Char number = e.KeyChar;

if ((number <= 47 || number >= 58) &&number != 8 && number != 44) //цифры, клавишаBackSpaceизапятаяв ASCII

e.Handled = true;

else

{

// Вычислим, какое сейчас значение в строке ввода длины струны

StringinputString;

if (number == 8) // ЕслибыланажатаBackSpace...

inputString = textBox1.Text.Substring(0, textBox1.TextLength - 1);

elseif (number == 44) // или если была нажата запятая

inputString = textBox1.Text;

else

inputString = textBox1.Text + number;

// В зависимости от полученного значения inputString зададим доступность поля величины сдвига струны.

if(!String.IsNullOrWhiteSpace(inputString))

textBox4.Enabled = true;

else

textBox4.Enabled = false;

}

}

privatevoid textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

Char number = e.KeyChar;

if ((number <= 47 || number >= 58) &&number != 8 && number != 44) //цифры, клавишаBackSpaceизапятаяв ASCII

e.Handled = true;

else

{

// Вычислим, какое сейчас значение в строке диаметра поперечного сечения струны

StringinputString;

if (number == 8) // ЕслибыланажатаBackSpace...

inputString = textBox1.Text.Substring(0, textBox1.TextLength - 1);

elseif (number == 44) // или если была нажата запятая

inputString = textBox1.Text;

else

inputString = textBox1.Text + number;

if(!String.IsNullOrWhiteSpace(inputString))

{

InternalVariables.d = Convert.ToDouble(inputString) / 1000.0;

}

}

}

privatevoid textBox3\_KeyPress(object sender, KeyPressEventArgs e)

{

Char number = e.KeyChar;

if ((number <= 47 || number >= 58) &&number != 8 && number != 44) //цифры, клавишаBackSpaceизапятаяв ASCII

e.Handled = true;

else

{

// Вычислим, какое сейчас значение в строке плотности материала струны

StringinputString;

if (number == 8) // ЕслибыланажатаBackSpace...

inputString = textBox3.Text.Substring(0, textBox1.TextLength - 1);

elseif (number == 44) // или если была нажата запятая

inputString = textBox3.Text;

else

inputString = textBox3.Text + number;

}

}

privatevoid textBox4\_KeyPress(object sender, KeyPressEventArgs e)

{

Char number = e.KeyChar;

if ((number <= 47 || number >= 58) &&number != 8 && number != 44) //цифры, клавишаBackSpaceизапятаяв ASCII

e.Handled = true;

}

privatevoid textBox5\_KeyPress(object sender, KeyPressEventArgs e)

{

Char number = e.KeyChar;

if ((number <= 47 || number >= 58) &&number != 8 && number != 44) //цифры, клавишаBackSpaceизапятаяв ASCII

e.Handled = true;

}

privatevoid textBox4\_KeyUp(object sender, KeyEventArgs e)

{

if(!String.IsNullOrWhiteSpace(textBox4.Text))

{

DoublestrValue = Convert.ToDouble(textBox4.Text) / 1000.0;

if (strValue>InternalVariables.length \* 0.01)

{

MessageBox.Show("Введенослишкомбольшоезначениесдвигаструны. Допускается значение до 0,01 длины струны.", "Ошибка ввода!", MessageBoxButtons.OK);

textBox4.Text = "";

}

else

InternalVariables.deltaY = strValue;

}

}

privatevoid textBox7\_KeyPress(object sender, KeyPressEventArgs e)

{

Char number = e.KeyChar;

if ((number <= 47 || number >= 58) &&number != 8 && number != 44) //цифры, клавишаBackSpaceизапятаяв ASCII

e.Handled = true;

}

privatevoid textBox7\_KeyUp(object sender, KeyEventArgs e)

{

if(!String.IsNullOrWhiteSpace(textBox7.Text))

{

InternalVariables.attenuation = Convert.ToDouble(textBox7.Text);

}

}

privatevoid textBox1\_KeyUp(object sender, KeyEventArgs e)

{

if(!String.IsNullOrWhiteSpace(textBox1.Text))

{

InternalVariables.length = Convert.ToDouble(textBox1.Text);

}

}

privatevoid textBox2\_KeyUp(object sender, KeyEventArgs e)

{

if(!String.IsNullOrWhiteSpace(textBox2.Text))

{

InternalVariables.d = Convert.ToDouble(textBox2.Text) / 1000.0;

}

}

privatevoid textBox3\_KeyUp(object sender, KeyEventArgs e)

{

if(!String.IsNullOrWhiteSpace(textBox3.Text))

{

InternalVariables.ro = Convert.ToDouble(textBox3.Text);

}

}

privatevoid textBox5\_KeyUp(object sender, KeyEventArgs e)

{

if(!String.IsNullOrWhiteSpace(textBox5.Text))

{

InternalVariables.G = Convert.ToDouble(textBox5.Text);

}

}

privatevoidвыходToolStripMenuItem\_Click(object sender, EventArgs e)

{

ActiveForm.Close();

}

privatevoidоПрограммеToolStripMenuItem\_Click(object sender, EventArgs e)

{

AboutBox1aboutBox1 = newAboutBox1();

aboutBox1.ShowDialog();

}

privatevoid tabControl1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (tabControl1.SelectedIndex == 1)

{

Diagram();

}

}

}

}