МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ЭКОНОМИКИ И СЕРВИСА

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

ОТЧЁТ

ПО УЧЕБНОЙ ОЗНАКОМИТЕЛЬНОЙ

ПРАКТИКЕ

Студент

Гр. БПИ-20-01 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И. И. Рязанов

Руководитель

Старший преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О. Б. Богданова

Владивосток 2023Задание

1. Используя произвольные язык программирования и среду разработки, создайте программу, которая отображает на экране монитора график брахистохроны в декартовой и полярной системах координат с центром в центре экрана монитора (окна или иной прямоугольной области экрана). При изменении размеров окна, график и все его атрибуты (координатная сетка, метки на шкале, подписи и т. д.) должны автоматически масштабироваться;

2. Используя результаты предыдущего задания, создайте анимацию примитива, движущегося по траектории построенной кривой: в качестве примитива используется закрашенная окружность красного цвета радиуса r>2.

Содержание

[Введение](#Введение)4

[1 Анализ технического задания](#Анализ)5

[1.1 Брахистохрона: основные сведения и область применения](#Гипоциклоида)5

[1.2 Математический аппарат для построения графика](#Матаппарат)5

[2 Выбор языка программирования и среды разработки](#Выборяс)7

[2.1 Выбор языка программирования](#язык)7

[2.2 Выбор среды разработки](#среда)8

[3 Разработка программы](#программа)10

[3.1 Блок-схема работы программы](#блоксхема)10

[3.2 Реализация графического интерфейса](#интерфейс)12

[3.3 Реализация программного кода](#код)14

[4. Тестирование программы](#тест)21

[4.1 Проверка работы программы при вводе корректных данных](#коррданные)21

[4.2 Проверка работы программы при вводе некорректных данных](#некоррданные)23

[5 Руководство пользователя](#руководство)25

[Заключение](#заключение)26

[Список использованной литературы](#список)27

[Приложение А](#приложение)28

# Введение

Учебная ознакомительная практика играет важную роль в образовательном процессе. Её целью является закрепление и углубление знаний, полученных в ходе учебных занятий, а также подготовка студентов к изучению более продвинутых предметов. Во время практики студенты приобретают навыки и компетенции, необходимые для будущей профессиональной деятельности.

Основная задача учебной ознакомительной практики – получение знаний, умений и навыков, которые будут полезными при решении реальных задач в области информационных систем и технологий. Также это время для закрепления и углубления знаний, полученных в теоретической части обучения, а также для приобретения новых знаний.

В рамках этой практики студентам предлагается разработать приложение для создания графиков и добавить анимацию движения примитива по заданной траектории на этом графике. Этот процесс позволяет студентам совершенствовать навыки алгоритмизации, работу с объектно-ориентированным программированием и создание графического интерфейса.

# 

# 1 Анализ технического задания

# 1.1 Брахистохрона: основные сведения и область применения

Брахистохро́на (от греч. βράχιστος «кратчайший» + χρόνος «время») — кривая скорейшего спуска. Задача о её нахождении была поставлена в июне 1696 года Иоганном Бернулли следующим образом: Среди плоских кривых, соединяющих две данные точки и, лежащих в одной вертикальной плоскости (ниже), найти ту, двигаясь по которой под действием только силы тяжести, сонаправленной отрицательной полуоси, материальная точка из достигнет за кратчайшее время.

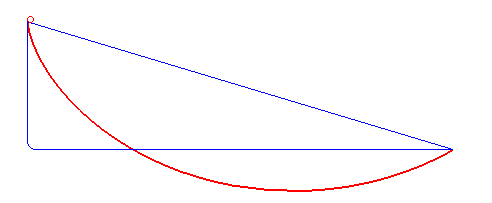


Рисунок 1 – Брахистохрона

Решением задачи о брахистохроне является дуга циклоиды с горизонтальным основанием, точка возврата которой находится в точке{\displaystyle A}, или иными словами, имеющая вертикальную касательную в точке{\displaystyle A}. Примечательно, что время спуска до нижней точки не зависит от расположения начальной точки на дуге брахистохроны.

1.2 Математический аппарат для построения графика

Для построения брахистохроны мы можем использовать как параметрическое уравнение, так и алгебраическое. Брахистохрона – это кривая, по которой тело движется от точки A до точки B под действием силы тяжести так, чтобы время движения было минимальным.

В декартовой системе координат (x, y), где начальная точка - (0, 0) и конечная точка - (a, -h), уравнение для брахистохроны может быть записано следующим алгебраическим уравнением (1):

Это уравнение дл брахистохроны может быть задано параметрическим уравнением (2):

(2)

Где и – это координаты точки на брахистохроне в момент времени , а – радиус кривой.

Для того чтобы построить график брахистохроны, необходимо выбрать конкретное значение параметра и нарисовать точки () для диапазона значений . Таким образом, мы получим форму брахистохроны для заданного .

Важно отметить, что форма брахистохроны зависит от значения параметра , и при различных значениях кривые будут иметь различные размеры и формы, но они все будут являться брахистохронами, так как обеспечивают минимальное время движения между точками A и B.

В полярной системе координат (r, θ), уравнение для брахистохроны может быть записано следующим образом (3):

(3)

Выразим через параметрическое уравнение (4):

(4)

где "" - параметр формы брахистохроны, "r" – расстояние от начальной точки до точки на кривой, "θ" - угол между радиус-вектором и положительным направлением оси x в полярных координатах.

2 Выбор языка программирования и среды разработки

2.1 Выбор языка программирования

В университете мы ознакомились с различными языками программирования, такими как Python, C, C++ и Java. Важно отметить, что все эти языки, за исключением C, имеют объектно-ориентированный характер.

Однако стоит подчеркнуть, что C отличается от других перечисленных языков в том, что он предоставляет программисту большую свободу и контроль, но при этом требует более осторожного и внимательного подхода. Например, в Си программисту необходимо самостоятельно управлять динамической памятью, следить за выделением и освобождением ресурсов, а также учитывать особенности работы с указателями и массивами. Это может привести к потенциально большему числу ошибок в коде.

Исходя из этого, в контексте нашего обучения, мы не рассматривали C как основной язык программирования. Вместо этого, мы углубились в изучение других объектно-ориентированных языков, таких как C#, который также является востребованным языком среди разработчиков. Эти четыре языка подойдут для выполнения поставленной задачи. Также будем учитывать, что выбранной платформой для программы является система Windows. В таблице 1 представлено сравнение языков по факторам по десятибалльной шкале: 1-4 – плохо, 5-7 –хорошо, 8-10 – отлично соответственно [3].

Таблица 1 – сравнение языков программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Язык программирования | | | |
| Факторы | Python | C++ | Java | C# |
| Скорость обработки | 2 | 8 | 6 | 7 |
| Читабельность (синтаксис) | 5 | 6 | 6 | 7 |
| Возможность оптимизации кода | 3 | 4 | 7 | 8 |
| Легкость использования GUI | 8 | 4 | 7 | 7 |
| Надежность | 6 | 2 | 8 | 8 |
| Поддержка Windows | 7 | 7 | 8 | 10 |

Несмотря на простой и легко понимаемый синтаксис, у Python есть свои ограничения в производительности, что может оказаться проблемой для приложений, требующих высокой скорости работы. Кроме того, оптимизация кода на Python может быть сложной задачей, и это ограничивает его способность достигать высокой эффективности.

C++, в свою очередь, обладает более сложным синтаксисом по сравнению с Python, и в нем легко допустить ошибки, которые могут привести к падению программы или некорректной работе. Кроме того, скорость выполнения кода на C++ не всегда является выдающейся, и этот язык может столкнуться с проблемами поддержки модульности, особенно при использовании большого количества внешних модулей.

Java имеет простой синтаксис, который похож на человеческую речь, и позволяет создавать кроссплатформенные приложения. Однако низкая скорость выполнения Java обусловлена тем, что код компилируется для виртуальной машины, что сказывается на производительности. Также сходство с естественными языками может привести к избыточной сложности и многословности в коде.

В данном контексте выбор был сделан в пользу C#, который в основном ориентирован на платформу Windows и активно поддерживается Microsoft. C# объединяет лучшие аспекты других объектно-ориентированных языков программирования. Например, он предоставляет возможности, подобные Java, и обладает хорошей оптимизацией кода, благодаря разнообразию синтаксических конструкций и возможности работы на платформе .Net. Кроме того, C# характеризуется надежностью благодаря работе среды CLR (Common Language Runtime), которая компилирует программу на виртуальной машине и изолирует ошибки, что не влияет на работу других программ в системе.

Были рассмотрены следующие IDE в таблице 2 [4].

2.2 Выбор среды разработки

После выбора языка программирования следует выбрать подходящую среду разработки. Выбор среды будет зависеть от таких факторов как удобность пользования, надежность, хорошая поддержка платформы Windows, встроенный GUI редактор, доступность и функциональность.

Были рассмотрены следующие IDE в таблице 2 [4].

Таблица 2 - Сравнение сред разработки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | IDE | | | |
| Факторы | Visual Studio | Project Rider | Eclipse | Code::Blocks |
| Удобность | + | + | - | + |
| Надежность | + | + | - | - |
| Поддержка Windows | + | + | + | + |
| Встроенный GUI редактор | + | + | - | - |
| Доступность | + | - | + | + |
| Функциональность | + | +- | + | - |

Сравнив несколько вариантов сред разработки, лучшим выбором оказалась Visual Studio по нескольким параметрам. Эта среда разработки удобна в использовании, разработана компанией Microsoft и обладает множеством функциональных возможностей для создания приложений под Windows, включая использование платформы .NET. Бесплатная версия Visual Studio подходит для выполнения различных задач, и в ней также присутствует встроенный редактор графического интерфейса, который легко интегрируется с кодом приложения.

С другой стороны, Project Rider, хотя имеет хороший набор функций, не является оптимальным выбором из-за платной лицензии. Среда Eclipse оценивается как менее удобная в использовании и не предоставляет встроенного редактора интерфейса. Что касается Code::Blocks, она характеризуется нестабильностью и отсутствием встроенного редактора графического интерфейса, который работает на основе библиотеки wxWidgets.

3 Разработка программы

3.1 Блок-схема работы программы и диаграммы классов алгоритма

В блок-схеме на рисунке 2 изображен алгоритм работы приложения.

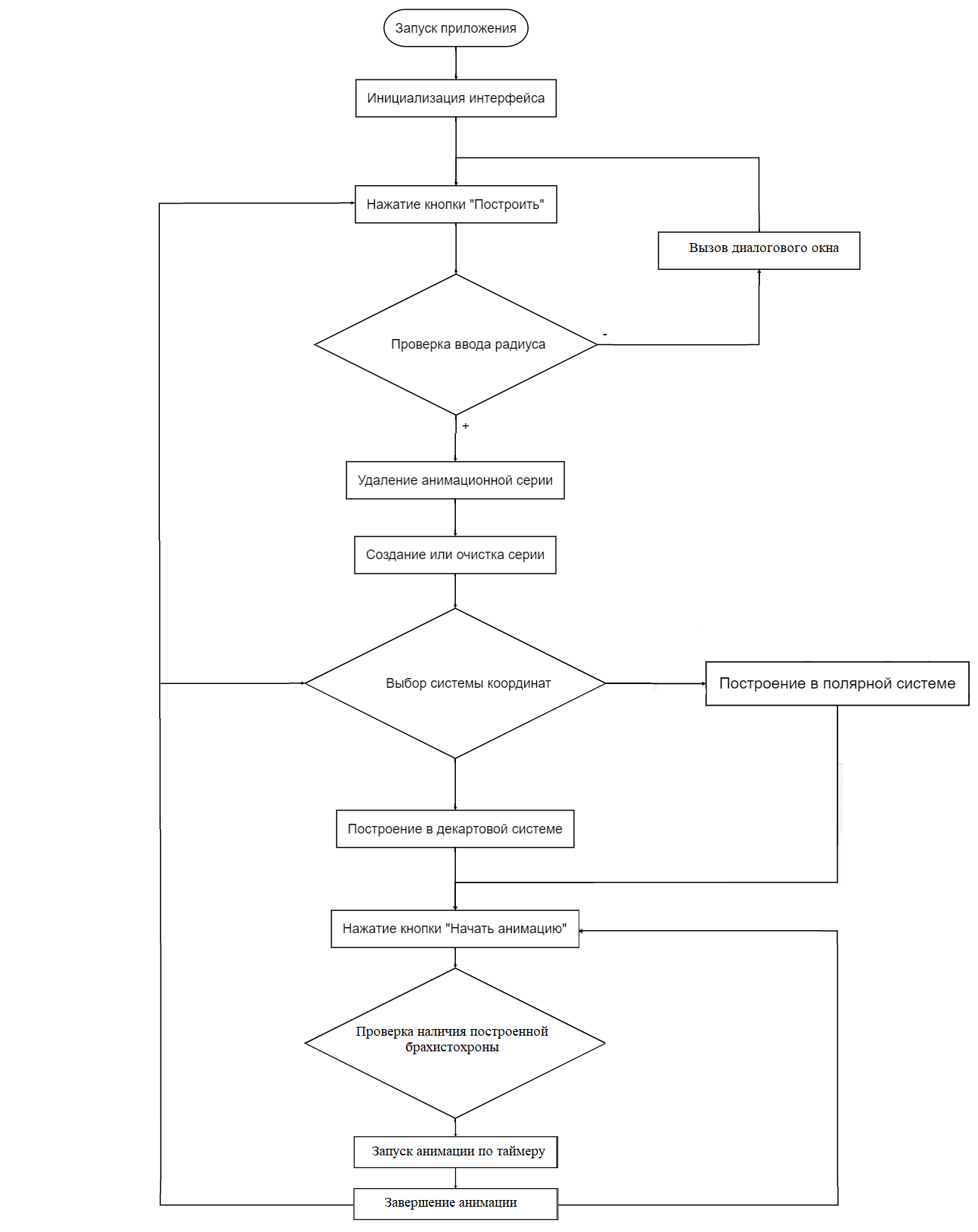


Рисунок 2 – Блок-схема работы приложения

Принцип работы программы заключается в том, что при запуске программы запускается метод Main(), который создает экземпляр класса Form1 и вызывает его метод InitializeComponent().

Затем происходит инициализация пользовательского интерфейса: В методе InitializeUI(), который вызывается из конструктора Form1. Создается и настраивается выпадающий список (comboBoxCoordinateSystem), кнопки buttonDraw и buttonStartAnimation, а также область графика (ChartArea).

Следующий этап - Построение брахистохроны. При нажатии кнопки "Построить" (buttonDraw), вызывается метод ButtonDraw\_Click. При этом программа пытается считать значение радиуса из текстового поля (textBoxRadius) и сохраняет его в переменную radius.

Проверяется выбранная система координат (Декартовая или Полярная). Соответственно, вызывается соответствующий код для построения брахистохроны с использованием формулы в выбранной системе координат.

Анимация брахистохроны осуществляется при нажатии кнопки "Начать анимацию" (buttonStartAnimation), вызывается метод ButtonStartAnimation\_Click. При этом программа проверяет, была ли построена брахистохрона. Если брахистохрона построена, то запускается анимация точки, движущейся вдоль построенной кривой. Анимация реализуется с использованием Timer, который вызывает метод AnimationTimer\_Tick с интервалом времени.

В этом методе происходит вычисление координат точки на брахистохроне в зависимости от системы координат и параметра анимации. Координаты точки добавляются в специальную серию графика для отображения анимации.

Изменение системы координат:

Пользователь может выбрать систему координат (Декартовая или Полярная) из выпадающего списка (comboBoxCoordinateSystem). Выбор системы координат обрабатывается в методе ComboBoxCoordinateSystem\_SelectedIndexChanged.

Визуализация графика: График брахистохроны и анимация визуализируются с использованием элементов управления Windows Forms, включая графический элемент Chart.

По завершении анимации приложение возвращает возможность изменить систему координат и перезапустить анимацию.

Приложение так же обеспечивает обработку ошибок, таких как некорректный ввод радиуса, а также информирует пользователя через диалоговые окна.

## 3.2 Реализация графического интерфейса

## Для этого этапа в конструкторе интерфейса было создано новое окно Windows forms. К данному окну были добавлены ComboBox для изменения системы координат, TextBox для ввода значения радиуса, два текстовых элемента label к ним, две кнопки (Button), одна для построения брахистохроны, вторая для запуска анимации примитива. Как интерфейс выглядит в редакторе изображено на рисунке 4.

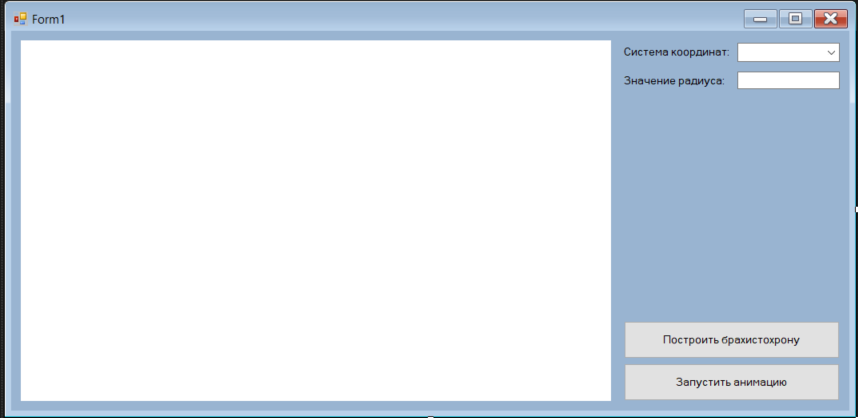


Рисунок 4 – Вид интерфейса в конструкторе

При запуске программы в системе Windows 10 окно программы выглядит в соответствии с рисунком 5.

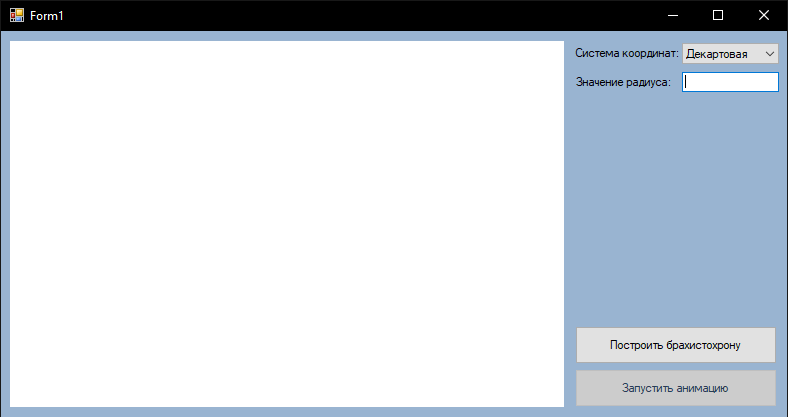


Рисунок 5 – Вид интерфейса при запуске в Windows 10

Для работы программы в поле ComboBox по умолчанию выбирается декартовая система координат. Поле TextBox ожидает на ввод значение радиуса, которое задаёт пользователь.

Построение осей графика и кривой брахистохроны происходит после нажатия кнопки “Построить брахистохрону”. Кнопка “Запустить анимацию” становиться доступной только после построения кривой брахистохроны. Во время проигрывания анимации смена системы координат становиться недоступной.

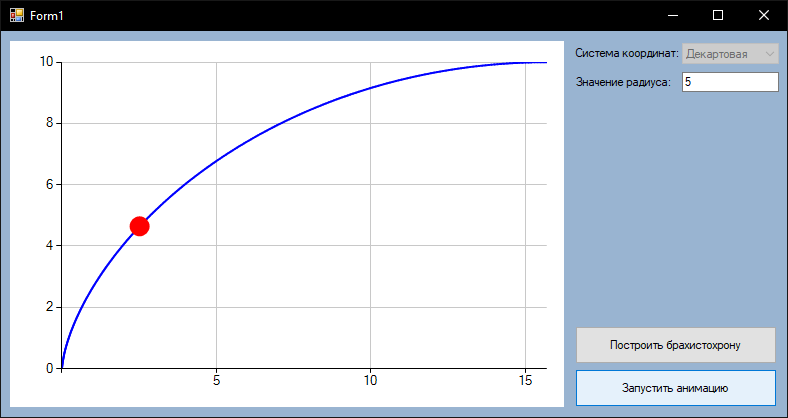


Рисунок 6 – Вид окна с активированной анимацией примитива

Ко всем элементам окна применено свойство anchor, которое не дает им менять свое расположение при изменении размеров окна, например, выйти за границы окна.

Для корректной работы программы при масштабировании окна был задан минимальный возможный размер окна в 200 пикселей в высоту и 400 пикселей в ширину как показано на Рисунке 7.

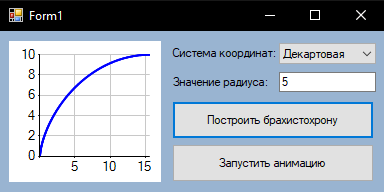


Рисунок 7 – Вид минимального размера интерфейса

3.3 Реализация программного кода

Работа кода вызывается методом Main() в статическом классе Program при запуске программе (рисунок 8). Этот метод в свою очередь вызывает класс Form1, содержащую в себе все методы, переменные, а также данные графического интерфейса программы. Сами коды обоих классов полностью приведены в приложении А.

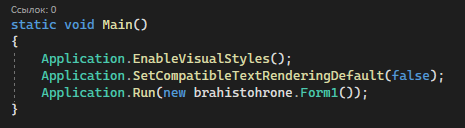


Рисунок 8 – Код метода Main()

Конструктор Form1():

Этот конструктор инициализирует экземпляр формы. Внутри него вызывается метод InitializeComponent(), который создает и настраивает компоненты формы, и InitializeUI(), который настраивает пользовательский интерфейс. Рисунок 9.

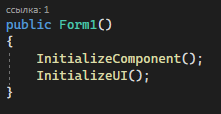


Рисунок 9 – Код конструктора Form1()

Метод InitializeUI():

Этот метод настраивает пользовательский интерфейс формы. Он выполняет следующие действия:

Устанавливает начальное значение выбора системы координат в выпадающем списке (comboBoxCoordinateSystem) на 0 (Декартовая система); Задает минимальные размеры формы (400x200); Устанавливает выпадающий список в режим только для чтения и привязывает обработчик события SelectedIndexChanged к методу ComboBoxCoordinateSystem\_SelectedIndexChanged; Назначает обработчики событий для кнопок "Построить" и "Начать анимацию"; Создает область графика и настраивает ее, включая установку пересечения осей в центре и уменьшение прозрачности сетки; Инициализирует таймер для анимации, устанавливая интервал и назначая обработчик события Tick.; Добавляет элемент графика на форму. Рисунок 10.

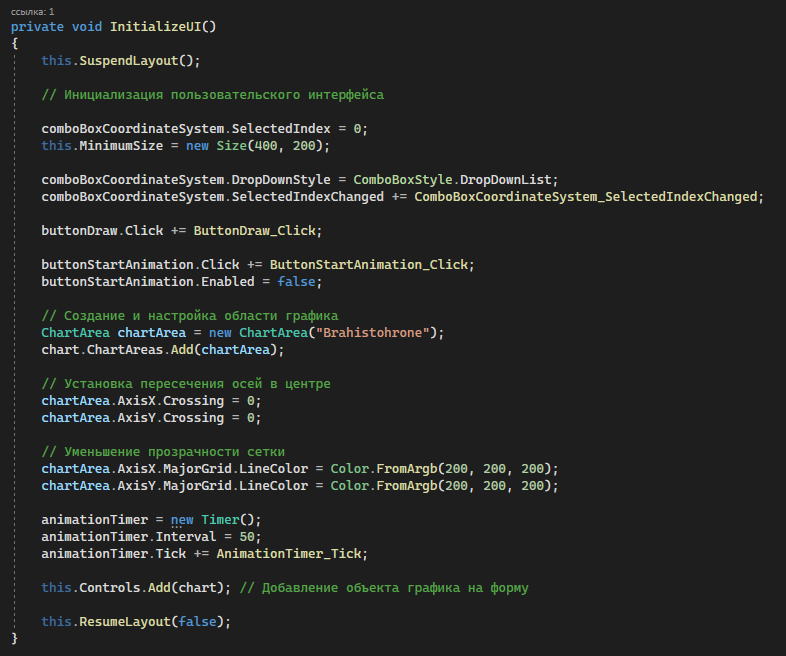


Рисунок 10 – Код метода InitializeUI()

Метод ButtonDraw\_Click():

Этот метод вызывается при нажатии кнопки "Построить". Он выполняет следующие действия:

Проверяет, было ли успешно разобрано введенное значение радиуса из текстового поля textBoxRadius. Если значение успешно разобрано, удаляет серию "Animation" (если она существует) с графика. Ищет серию "Brahistohrone" на графике. Если она не существует, создает новую серию и настраивает ее для отображения брахистохроны. Если серия уже существует, очищает ее точки. В зависимости от выбранной системы координат (Декартовая или Полярная), вычисляет и добавляет точки брахистохроны на график. Включает кнопку "Начать анимацию", так как брахистохрона успешно построена.

Форматирует ось X для отображения значения радиуса с одним знаком после запятой для радиуса, близкого к 1, и как целое число для остальных значений радиуса. Если введенное значение радиуса некорректно, выводит сообщение об ошибке. Рисунок 11, 12.

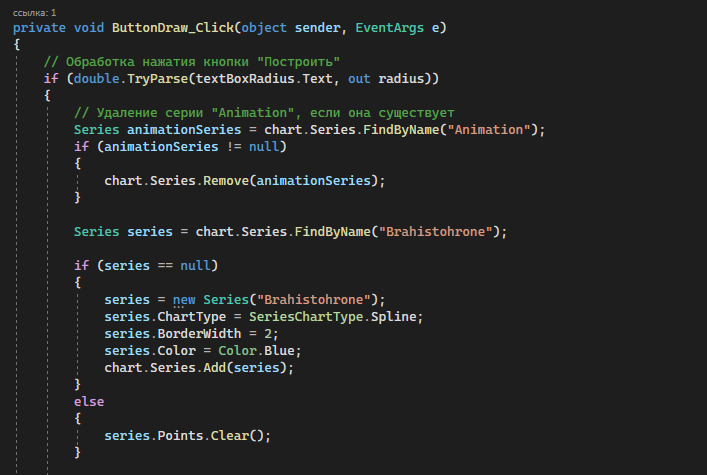


Рисунок 11 – Код метода ButtonDraw\_Click() (Часть 1)

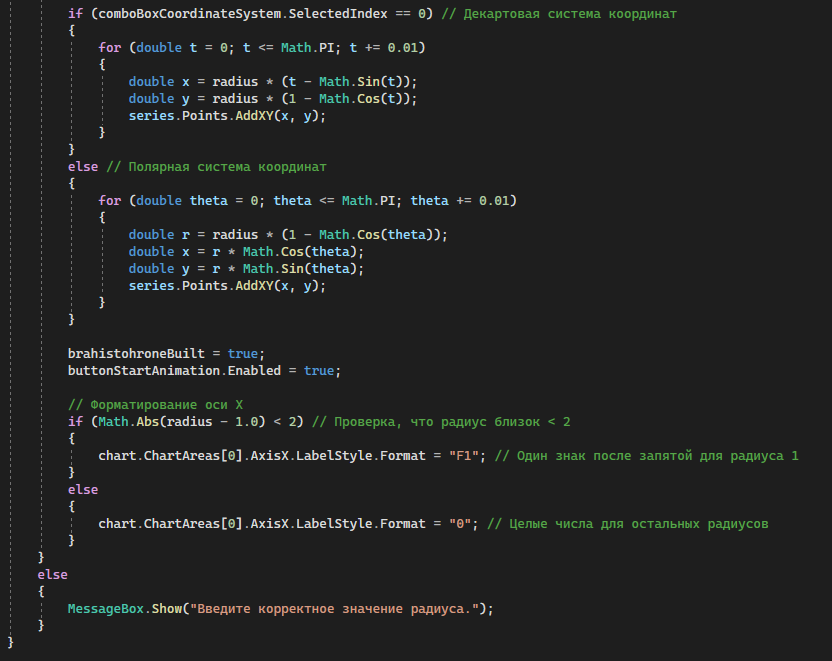


Рисунок 12 – Код метода ButtonDraw\_Click() (Часть 2)

Метод ButtonStartAnimation\_Click():

Этот метод вызывается при нажатии кнопки "Начать анимацию". Он выполняет следующие действия:

Проверяет, была ли построена брахистохрона (brahistohroneBuilt); Если брахистохрона построена и система координат доступна для изменения, останавливает выбор системы координат (comboBoxCoordinateSystem), сбрасывает параметр анимации (animationParameter) и запускает таймер анимации. Рисунок 13.

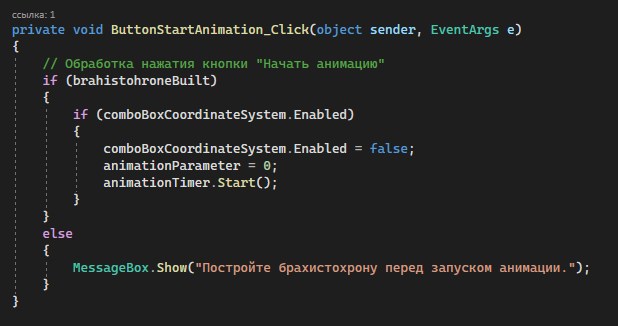


Рисунок 13 – Код метода ButtonStartAnimation\_Click()

Метод AnimationTimer\_Tick():

Этот метод вызывается при каждом срабатывании таймера анимации. Он выполняет следующие действия:

1. Проверяет, превышает ли параметр анимации (animationParameter) значение π (половина оборота по брахистохроне).
2. Если да, останавливает таймер анимации и разрешает выбор системы координат; В зависимости от выбранной системы координат (Декартовая или Полярная), рассчитывает координаты точки на брахистохроне;
3. Очищает серию "Animation" и создает ее, если она не существует;
4. Добавляет координаты точки в серию, что обновляет анимацию;
5. Увеличивает параметр анимации для движения точки по брахистохроне.

Данный метод представлен на рисунке 14:



Рисунок 14 – Код метода AnimationTimer\_Tick()

Метод ComboBoxCoordinateSystem\_SelectedIndexChanged():

Этот метод вызывается при изменении выбора системы координат в выпадающем списке comboBoxCoordinateSystem. Он выполняет следующие действия:

1. Проверяет выбор системы координат: если выбраны "Декартовая" или "Полярная", активирует кнопку "Построить", если выбрано что-то другое, отключает кнопку "Построить";
2. Отключает кнопку "Начать анимацию", так как изменение системы координат потребует новой постройки брахистохроны. Рисунок 15.

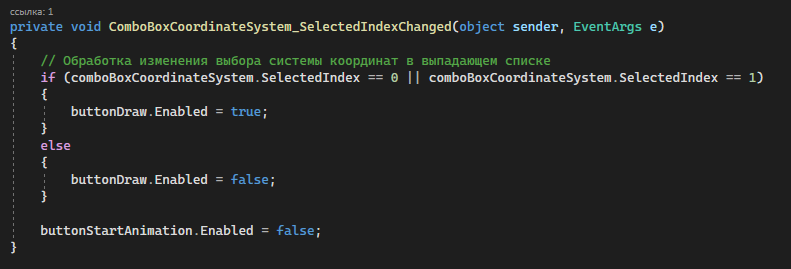


Рисунок 15 – Код метода ComboBoxCoordinateSystem\_SelectedIndexChanged()

Метод Form1\_Load():

Этот метод вызывается при загрузке формы и может быть использован для дополнительных настроек, которые должны быть выполнены при запуске приложения.

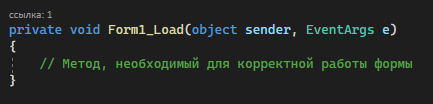


Рисунок 16 – Код метода Form1\_Load()

4 Тестирование программы

4.1 Проверка работы программы при вводе корректных данных

Проверяем программу на правильное построение графика при вводе корректных данных. У переменной r корректными вводимыми значениями являются целые числа от 1 до 25 включительно, у переменной k – от 2 до 50 включительно. На рисунке показана 12 работа программы при корректных данных и включенной анимации.

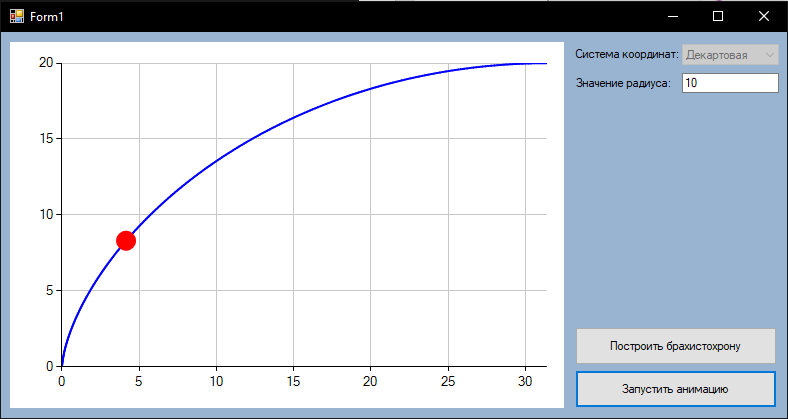


Рисунок 17 – Вид окна программы при вводе корректных данных в декартовой системе координат

Меняем систему координат на полярную. Приложение работает корректно, как показано на рисунке 18:

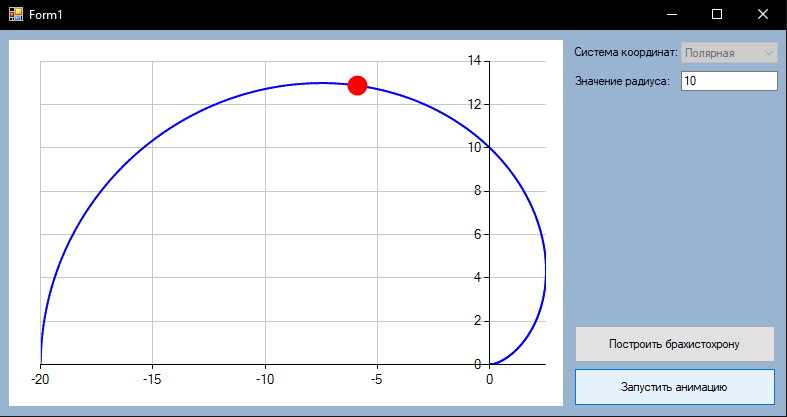


Рисунок 18 – Вид окна программы при вводе корректных данных в полярной системе координат

При изменении размеров окна и масштабировании, программа также работает правильно в соответствии с рисунком 14. Сверху вместо названия программы отображается множитель масштаба.

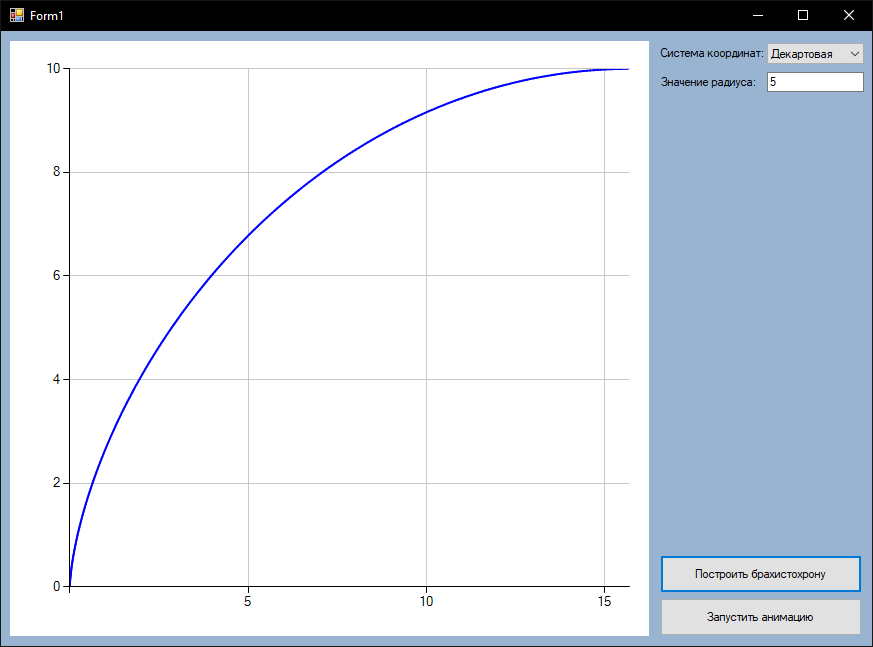


Рисунок 19 – Вид окна программы при изменении размеров окна и масштабировании

Завершение работы программы выполняется корректно, для этого нажимаем на кнопку крестика.

4.2 Проверка работы программы при вводе некорректных данных

Значение радиуса не может принимать символы, знаки препинания, кроме чисел и запятой. То есть пользователь не сможет ввести некорректные данные, кроме нецелых и отрицательных чисел.

На рисунках 20, 21 изображена попытка ввода некорректных данных.

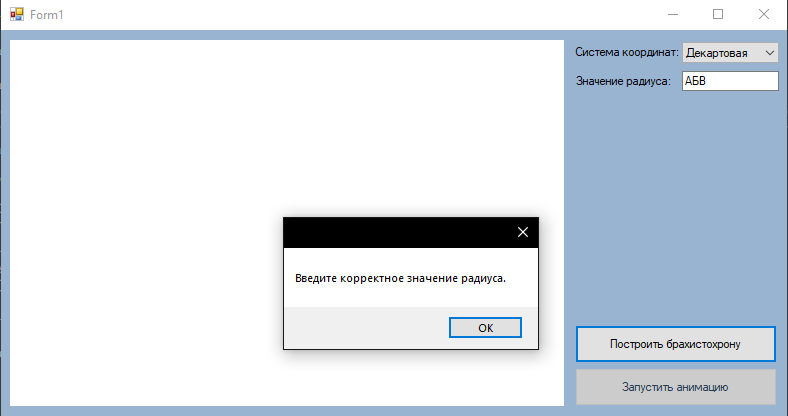


Рисунок 20 – Обработка исключения при вводе некорректных данных (символы).

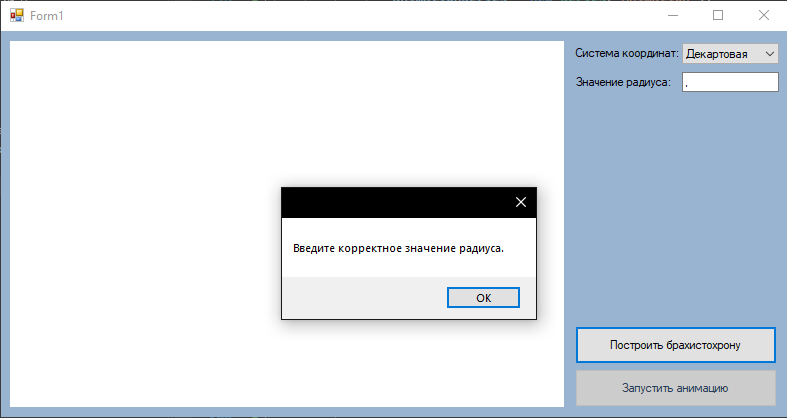


Рисунок 21 – Обработка исключения при вводе некорректных данных (Знаки препинания).

При вводе символов, знаков препинания программа не будет строить брахистохрону, а выведет пользователю диалоговое окно с просьбой ввести корректные данные.

Из результатов тестирования можно сделать вывод, что программа работает корректно.

5 Руководство пользователя

На рисунке 22 изображены основные элементы управления программы.

Для построения графика брахистохроны пользователю необходимо ввести значение радиуса и выбрать систему координат. Радиус может принимать любые числовые значения, кроме символов, знаков препинаний и т.п. При вводе некорректных значений радиуса программа попросит ввести корректное значение радиуса. По умолчанию установлена декартовая система координат.

График кривой будет строиться после нажатия на кнопку “Построить брахистохрону”.

Для запуска анимации примитива по построенной кривой брахистохроны необходимо нажать на кнопку «Анимация». При изменении размеров окна, его сворачивании и масштабировании графика анимация не прекращается.

Для завершения работы программы и его выхода, необходимо нажать на крестик закрытия окна.

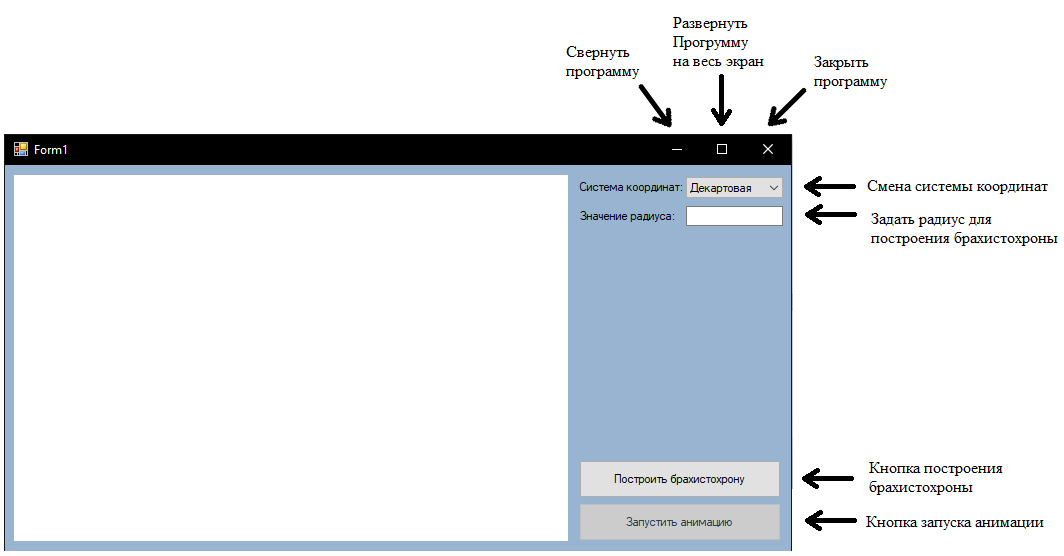


Рисунок 22 - Основные элементы управления программой

# Заключение

В ходе прохождения учебной ознакомительной практики было разработано приложение, позволяющее построить график брахистохроны, а также анимацию движения примитива по траектории данной кривой.

В процессе разработки приложения были усовершенствованы навыки в области объектно-ориентированного программирования, алгоритмизации, а также умения построения графического интерфейса на основе Windows forms.

В результате были закреплены теоретические и практические знания, полученные в процессе обучения, а также получены навыки, которые окажутся полезными в ходе последующего изучения учебных дисциплин.

Список использованной литературы

1 Hypocycloid // Wikipedia – свободная интернет-энциклопедия – режим доступа: <https://en.wikipedia.org/wiki/Brachistochrone_curve>

2 Белоцерковская И.Е., Втюрин М.Ю. моделирование движения тел во времени. задача о брахистохроне / Белоцерковская И.Е., Втюрин М.Ю. // научная электронная библиотека «eLIBRARY.ru» - 2019 – режим доступа: <https://www.elibrary.ru/item.asp?id=42371360>

3 Какой язык программирования выбрать исходя из требований к проекту? // «АрдуиноПлюс.ру» - 2018 – режим доступа: <https://arduinoplus.ru/vibor-yazika-programmirovania/>

4 Бубнов И. Лучшие IDE для разработки на C# / Бубнов И. // «GeekBrains» - образовательный портал – 2018 – режим доступа: <https://gb.ru/posts/c_sharp_ides>

5 Руководства по Visual Studio | C# // Microsoft – режим доступа: <https://docs.microsoft.com/ru-ru/visualstudio/get-started/csharp/?view=vs-2022>

6 Документация по Visual Studio 2022 // Microsoft – режим доступа: <https://docs.microsoft.com/ru-ru/visualstudio/windows/?view=vs-2022&preserve-view=true>

Приложение А

Класс Main()

using System;

using System.Windows.Forms;

namespace last

{

internal static class Program

{

// Главная точка входа для приложения

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new brahistohrone.Form1());

}

}

}

Класс Form1

using System;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

using System.Drawing;

namespace brahistohrone

{

public partial class Form1 : Form

{

private double radius;

private Timer animationTimer;

private double animationParameter = 0;

private bool brahistohroneBuilt = false;

public Form1()

{

InitializeComponent();

InitializeUI();

}

private void InitializeUI()

{

this.SuspendLayout();

// Инициализация пользовательского интерфейса

comboBoxCoordinateSystem.SelectedIndex = 0;

this.MinimumSize = new Size(400, 200);

comboBoxCoordinateSystem.DropDownStyle = ComboBoxStyle.DropDownList;

comboBoxCoordinateSystem.SelectedIndexChanged += ComboBoxCoordinateSystem\_SelectedIndexChanged;

buttonDraw.Click += ButtonDraw\_Click;

buttonStartAnimation.Click += ButtonStartAnimation\_Click;

buttonStartAnimation.Enabled = false;

// Создание и настройка области графика

ChartArea chartArea = new ChartArea("Brahistohrone");

chart.ChartAreas.Add(chartArea);

// Установка пересечения осей в центре

chartArea.AxisX.Crossing = 0;

chartArea.AxisY.Crossing = 0;

// Уменьшение прозрачности сетки

chartArea.AxisX.MajorGrid.LineColor = Color.FromArgb(200, 200, 200);

chartArea.AxisY.MajorGrid.LineColor = Color.FromArgb(200, 200, 200);

animationTimer = new Timer();

animationTimer.Interval = 50;

animationTimer.Tick += AnimationTimer\_Tick;

this.Controls.Add(chart); // Добавление объекта графика на форму

this.ResumeLayout(false);

}

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

{

// Обработка нажатия кнопки "Построить"

if (double.TryParse(textBoxRadius.Text, out radius))

{

// Удаление серии "Animation", если она существует

Series animationSeries = chart.Series.FindByName("Animation");

if (animationSeries != null)

{

chart.Series.Remove(animationSeries);

}

Series series = chart.Series.FindByName("Brahistohrone");

if (series == null)

{

series = new Series("Brahistohrone");

series.ChartType = SeriesChartType.Spline;

series.BorderWidth = 2;

series.Color = Color.Blue;

chart.Series.Add(series);

}

else

{

series.Points.Clear();

}

if (comboBoxCoordinateSystem.SelectedIndex == 0) // Декартовая система координат

{

for (double t = 0; t <= Math.PI; t += 0.01)

{

double x = radius \* (t - Math.Sin(t));

double y = radius \* (1 - Math.Cos(t));

series.Points.AddXY(x, y);

}

}

else // Полярная система координат

{

for (double theta = 0; theta <= Math.PI; theta += 0.01)

{

double r = radius \* (1 - Math.Cos(theta));

double x = r \* Math.Cos(theta);

double y = r \* Math.Sin(theta);

series.Points.AddXY(x, y);

}

}

brahistohroneBuilt = true;

buttonStartAnimation.Enabled = true;

// Форматирование оси X

if (Math.Abs(radius - 1.0) < 2) // Проверка, что радиус близок < 2

{

chart.ChartAreas[0].AxisX.LabelStyle.Format = "F1"; // Один знак после запятой для радиуса 1

}

else

{

chart.ChartAreas[0].AxisX.LabelStyle.Format = "0"; // Целые числа для остальных радиусов

}

}

else

{

MessageBox.Show("Введите корректное значение радиуса.");

}

}

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

{

// Обработка нажатия кнопки "Начать анимацию"

if (brahistohroneBuilt)

{

if (comboBoxCoordinateSystem.Enabled)

{

comboBoxCoordinateSystem.Enabled = false;

animationParameter = 0;

animationTimer.Start();

}

}

else

{

MessageBox.Show("Постройте брахистохрону перед запуском анимации.");

}

}

private void AnimationTimer\_Tick(object sender, EventArgs e)

{

// Обработка события таймера для анимации

if (animationParameter > Math.PI)

{

animationTimer.Stop();

comboBoxCoordinateSystem.Enabled = true;

return;

}

double x, y;

if (comboBoxCoordinateSystem.SelectedIndex == 0) // Декартовая система координат

{

x = radius \* (animationParameter - Math.Sin(animationParameter));

y = radius \* (1 - Math.Cos(animationParameter));

}

else // Полярная система координат

{

double theta = animationParameter;

double r = radius \* (1 - Math.Cos(theta));

x = r \* Math.Cos(theta);

y = r \* Math.Sin(theta);

}

Series animationSeries = chart.Series.FindByName("Animation");

if (animationSeries != null)

{

animationSeries.Points.Clear();

}

else

{

animationSeries = new Series("Animation");

animationSeries.ChartType = SeriesChartType.Line;

animationSeries.MarkerStyle = MarkerStyle.Circle;

animationSeries.MarkerSize = 20; // Устанавливаем размер маркера

animationSeries.Color = Color.Red;

chart.Series.Add(animationSeries);

}

animationSeries.Points.AddXY(x, y);

animationParameter += 0.1;

}

private void ComboBoxCoordinateSystem\_SelectedIndexChanged(object sender, EventArgs e)

{

// Обработка изменения выбора системы координат в выпадающем списке

if (comboBoxCoordinateSystem.SelectedIndex == 0 || comboBoxCoordinateSystem.SelectedIndex == 1)

{

buttonDraw.Enabled = true;

}

else

{

buttonDraw.Enabled = false;

}

buttonStartAnimation.Enabled = false;

}

private void Form1\_Load(object sender, EventArgs e)

{

// Код, который выполняется при загрузке формы

}

}

}