минобрнауки россии

федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |  |  |
| --- | --- | --- |
| Институт (факультет) | | Институт информационных технологий |
| Кафедра | Математического и программного обеспечения ЭВМ | |

КУРСОВАЯ РАБОТА

|  |  |
| --- | --- |
| по дисциплине | Структурное программирование |

|  |  |
| --- | --- |
| на тему | Программирование на языке высокого уровня: Тетраэдр |

|  |
| --- |
| Выполнила студентка группы |
| 1ПИб-02-1оп-22 |
| направление подготовки (специальности) |
| 09.03.04., Программная инженерия |
| шифр, наименование |
| Рябкова Анна Леонидовна |
| фамилия, имя, отчество |

|  |
| --- |
| Руководитель |
| Пышницкий Константин Михайлович |
| фамилия, имя, отчество |
| Старший преподаватель |
| должность |

|  |
| --- |
| Дата представления работы |
| « » \_\_\_\_\_\_ 2023 г. |
| Заключение о допуске к защите |
|  |
|  |
|  |
| Оценка \_\_\_\_\_, \_\_\_\_\_\_ количество баллов |
|  |
| Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Череповец, 2023

# Аннотация

Курсовой проект по дисциплине «Структурное программирование» на тему «Программирование на языке высокого уровня: Тетраэдр» выполнила студентка группы 1ПИб-02-1оп-22 Института информационных технологий, Рябкова Анна Леонидовна.

Главной целью курсового проекта является разработка программы, которая будет выводить на экран двухмерную проекцию геометрического тела с заданными параметрами.

Курсовая работа содержит в себе: введение, описание предметной области, используемые классы, переменные и элементы интерфейса, а также заключение, код программы и список источников.

Оглавление

[Аннотация 2](#_Toc136886333)

[Введение 4](#_Toc136886334)

[1. Описание предметной области. 5](#_Toc136886335)

[2. Описание классов Graphics, Pen, Brush. 7](#_Toc136886336)

[2.1. Класс Graphics 7](#_Toc136886337)

[2.3. Класс Brush 9](#_Toc136886338)

[3. Описание созданного приложения 10](#_Toc136886339)

[3.1. Постановка задачи 10](#_Toc136886340)

[3.2. Логическое проектирование 10](#_Toc136886341)

[3.3. Физическое проектирование 13](#_Toc136886342)

[3.4. Проектирование интерфейса 16](#_Toc136886343)

[3.5. Тестирование 18](#_Toc136886344)

[3.6. Результат работы 19](#_Toc136886345)

[Заключение 20](#_Toc136886346)

[Приложение 1. Техническое задание 22](#_Toc136886347)

[Приложение 2. Руководство пользования 30](#_Toc136886348)

[Приложение 3. Программный код 32](#_Toc136886349)

# Введение

C++ - компилируемый, статический типизированный язык программирования общего назначения.

Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником - языком C, - наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр [7].

Данная курсовая работа является закрепление знаний, полученных на лекциях лабораторных работах по дисциплине «Структурное программирование».

Целью курсовой работы является написание программы с использованием технологии Windows Forms на языке программирования С++, которая будет строить и отображать тетраэдр и выполнять его вращение.

Windows Forms — это технология интеллектуальных клиентов для NET Framework. Она представляет собой набор управляемых библиотек, упрощающих выполнение стандартных задач, таких как чтение из файловой системы и запись в нее. С помощью такой среды разработки, как Visual Studio, можно создавать интеллектуальные клиентские приложения Windows Forms, которые отображают информацию, запрашивают ввод от пользователей и обмениваются данными с удаленными компьютерами по сети [8].

# 1. Описание предметной области.

Целью курсовой работы было изучить возможности работы с графикой в C++. В результате была создана программа, выполняющая построение закрашенного тетраэдра с возможностью: изменения параметров фигуры (высота и ширина), вращения фигуры, изменения скорости вращения, изменения цвета граней фигуры, а также фона.

Данный продукт является уникальным среди других продуктов, написанных на языке C++ с использованием Windows Forms.

Предметной областью является тетраэдр (рис.1). Это геометрическое тело из четырех граней, каждая из которых является правильным треугольником [6].

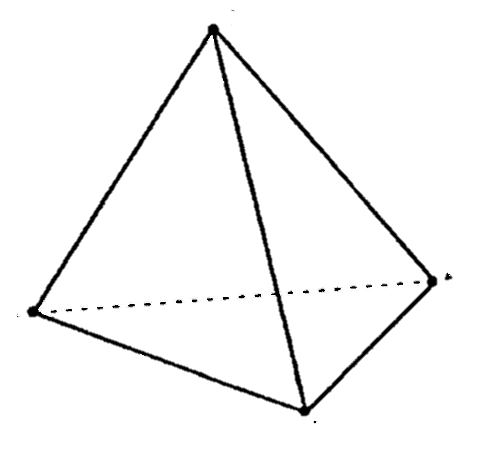


Рис. 1. Тетраэдр

Для того, чтобы начать осуществлять вращение тетраэдра нужно вспомнить Декартову систему координат.

Декартовой прямоугольной системой координат на плоскости (рис.2) (в пространстве) называют две или три взаимно перпендикулярные оси с общим началом. Первая ось OX называется осью абсцисс, вторая ось OY - осью ординат, третья ось OZ - осью аппликат.



Рис.2. Декартова система координат

За центр, относительно которого будет вращаться тетраэдр, была взята точка, делящая пополам одну из ребер тетраэдра.

# 2. Описание классов Graphics, Pen, Brush.

## 2.1. Класс Graphics

Класс Graphics входит в пространство имен System.Drawing, как и большинство классов для работы с графикой. Класс Graphics определяет набор методов для вывода текста, изображений и геометрических фигур [5].

Класс Graphics содержит огромное количество методов, некоторые из которых описаны в табл. 1.

Таблица 1

Методы Graphics

|  |  |
| --- | --- |
| Метод | Описание |
| BeginContainer() | Сохраняет графический контейнер, содержащий текущее состояние данного объекта Graphics, а затем открывает и использует новый графический контейнер |
| DrawString() | Создает заданную текстовую строку в указанном прямоугольнике с помощью заданных объектов Brush и Font. |
| DrawLine() | Рисует линию. Передаваемые параметры: объект типа Pen, определяющий цвет и стиль линии. Соединяет координаты точки начала и координаты конца |
| Clear | Очищает всю поверхность рисования и выполняет заливку поверхности указанным цветом фона. |
| DrawRectangle(Pen,…) | Рисует прямоугольник, который определен парой координат, шириной и высотой. |
| FillEllipse(Brush,…) | Заполняет внутреннюю часть эллипса, определяемого ограничивающим прямоугольником, заданным с помощью пары координат, ширины и высоты. |
| FillPolygon() | Заполняет внутреннюю часть многоугольника, определяемого массивом точек, заданных структурами Point. |
| FillRectangles() | Заполняет внутреннюю часть набора прямоугольников, определяемых структурами RectangleF. |
| Flush() | Вызывает принудительное выполнение всех отложенных графических операций и немедленно возвращается, не дожидаясь их окончания. |
| FromImage() | Создает новый объект Graphics из указанного объекта Image. |
| TransformPoints() | Преобразует массив точек из одной системы координат в другую, используя текущие мировое и страничное преобразования данного объекта Graphics. |

2.2. Класс Pen

Этот класс входит в пространство имен «System.Drawing». Определяет объект, используемый для рисования прямых линий и кривых. Линия, нарисованная с помощью Pen, может быть заполнена различными стилями заливки, включая сплошные цвета и текстуры. Этот класс не наследуется [1].

Описание некоторых методов класса Pen представлены в табл. 2.

Таблица 2

Методы Pen

|  |  |
| --- | --- |
| Метод | Описание |
| Clone() | Создает копию объекта Pen |
| Dispose() | Освобождает все ресурсы, используемые этим объектом Pen. |
| Finalize() | Позволяет объекту попытаться освободить ресурсы и выполнить другие операции очистки, перед тем как он будет уничтожен во время сборки мусора. |
| MultiplyTransform() | Умножает матрицу преобразования объекта Pen на заданный объект Matrix. |
| ResetTransform() | Возвращает матрице геометрического преобразования объекта Pen единичное значение. |
| Equals(Object) | Определяет, равен ли указанный объект текущему объекту. |
| RotateTransform(Single) | Поворачивает локальное геометрическое преобразование на заданный угол. Этот метод добавляет поворот перед преобразованием. |

## 2.3. Класс Brush

Этот класс входит в пространство имён «System.Drawing». Определяет объекты, которые используются для заливки внутри графических фигур, таких как прямоугольники, круги, многоугольники и дорожки [2].

Описание некоторых методов класса Brush представлены в табл. 3.

Таблица 3

Методы класса Brush

|  |  |
| --- | --- |
| Название метода | Описание работы метода |
| Clone() | Создает копию этого объекта Brush. |
| Dispose() | Освобождает все ресурсы, используемые этим объектом Brush. |
| Dispose(Boolean) | Освобождает управляемые и неуправляемые ресурсы, используемые объектом Brush. |
| GetHashCode() | Служит функцией по умолчанию. |
| SetNativeBrush(IntPtr) | В производном классе задает ссылку на объект GDI+ Вrush. |

# 3. Описание созданного приложения

## 3.1. Постановка задачи

Программа должна:

* Строить трехмерное рамочное изображение закрашенного тетраэдра;
* Иметь возможность менять цвет ребер тетраэдра;
* Иметь возможность менять цвет фона;
* Иметь возможность изменять масштаб тетраэдра;
* Иметь возможность менять длину ребер тетраэдра;
* Иметь возможность начинать и останавливать вращение тетраэдра;

## 3.2. Логическое проектирование

Программа на входе получает следующие значения:

* Высоту тетраэдра;
* Ширину тетраэдра;
* Скорость вращения;
* Цвет ребер тетраэдра;
* Цвет заднего фона.

Пользователь получает закрашенную фигуру уже с заранее заданными координатами вершин.

Координаты середины холста высчитываются исходя из размеров длины и ширины холста, которые мы делим пополам:

* Center(x) = w/2 - относительно оси абсцисс;
* Center(y) = h/2 - относительно оси ординат;

Где: x,y – абсцисса и ордината точки, соответственно, Center(x), Center(y) – координаты середины холста, w – длина холста, h – ширина холста.

Координаты вершин тетраэдра рассчитываются по формулам:

* p1(x) = Center(x) - wf / 2 = w/2 - wf/2;
* p1(y) = Center(y)+ hf / 2 = h/2 + hf/2;
* p2(x) = Center(x) + wf = w/2 + wf;
* p2(y) = Center(y) + hf / 2 = h/2 + hf/2;
* p3(x) = Center(x) + 1,6 \* wf = w/2 + 1,6\*wf;
* p3(y) = Center(y) + 0,6 \* hf = h/2 + 0,6 \* hf;
* p4(x) = Center(x) + wf = w/2 + wf;
* p4(y) = Center(y) - hf / 2 = h/2 – hf/2;

Где: p(1,2,3,4) – вершины тетраэдра, x и y – абсцисса и ордината точки, соответственно, Center(x), Center(y) – координаты середины холста, w – длина холста, h – ширина холста, wf – ширина тетраэдра, hf – высота тетраэдра.

Для того, чтобы тетраэдр был закрашенным, мы создаем 3 массива из трех точек, чтобы закрасить каждую из 3 видимых граней.

После того, как координаты вершин будут найдены, необходимо соединить их линиями, которые будут являться ребрами тетраэдра.

* Соединяем вершину p1 и р2;
* Соединяем вершину p1 и р3;
* Соединяем вершину p1 и р4;
* Соединяем вершину p2 и р3;
* Соединяем вершину p2 и р4;
* Соединяем вершину p3 и р4;

Для того чтобы тетраэдр начал вращение необходимо нажать на кнопку, которая запускает таймер. Таймер выполняет пересчет координат вершин тетраэдра на каждый тик с помощью изменения угла.

Для того, чтобы выполнить пересчет координат необходимо использовать первоначальные значения выбранной точки. Кроме этого, нужно ввести переменную angle (она изменяется от 1-6), которая определяет во сколько раз будет меняться угол поворота, и которая вместе с этим и отвечает за измерение скорости вращения фигуры.

angle = 1, 2, 3, 4, 5, 6

alpha = ((-15 \* angle) \* π)/ 360◦

oldp = p1(х)

p1(х) = (oldp – Сenter(х)) \* cos(alpha) - (p1(у) – Сenter(у)) \* sin(alpha) + Сenter(х)

p1(у) = (oldp – center(х)) \* sin(alpha) + (p1(у) – Сenter(у)) \* cos(alpha) + Сenter(у);

oldp = p2(х);

p2(х) = (oldp – Сenter(х)) \* cos(alpha) - (p2(у) – Сenter(у)) \* sin(alpha) + Сenter(х);

p2(у) = (oldp – Сenter(х) \* sin(alpha) + (p2(у) – Сenter(у)) \* cos(alpha) + Сenter(у);

oldp = p3(х);

p3(х) = (oldp – Сenter(х)) \* cos(alpha) - (p3(у) – Сenter(у)) \* sin(alpha) + Сenter(х);

p3(у) = (oldp – Сenter(х)) \* sin(alpha) + (p3(у) – Сenter(у)) \* cos(alpha) + Сenter(у);

oldp = p4(х);

p4(х) = (oldp – Сenter(х)) \* cos(alpha) - (p4(у) – Сenter(у)) \* sin(alpha) + Сenter(х);

p4(у) = (oldp – Сenter(х)) \* sin(alpha) + (p4(у) – Сenter(у)) \* cos(alpha) + Сenter(у);

Где: p(1,2,3,4) – вершины тетраэдра, x и y – абсцисса и ордината точки, соответственно, Center(x), Center(y) – координаты середины холста, oldp – координата первоначального значения точки р1,2,3,4(х), alpha – угол поворота фигуры, angle – коэффициент вращения фигуры.

Затем нужно залить фоном на месте старого тетраэдра. После пересчета координат выполняется построение линий, соединяющих новые вершины, при помощи алгоритма, показанного выше.

Алгоритм работы программы представлен в виде блок-схемы (рис.3).

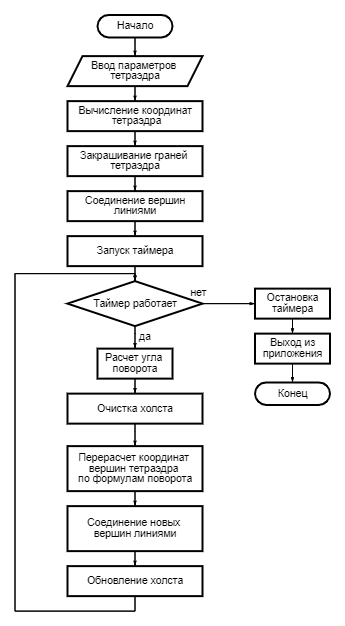


Рис.3. Блок-схема программы

## 3.3. Физическое проектирование

Для построения тетраэдра были использованы переменные, приведенные в табл.4.

Таблица 4

Переменные

|  |  |  |
| --- | --- | --- |
| Наименование | Обозначение | Тип данных |
| Высота тетраэдра | height | int |
| Ширина тетраэдра | width | int |
| Высота холста | Height | int |
| Ширина холста | Width | int |
| Переменная, регулирующая значение угла | angle | int |
| Угол поворота | alpha | int |
| Центр холста | center | int |
| Локальная переменная для хранения значения прошлой координаты по Х | oldp | float |
| Координаты первой точки | p1 | float |
| Координаты второй точки | p2 | float |
| Координаты третьей точки | p3 | float |
| Координаты четвертой точки | p4 | float |
| Переменная для рисования новой фигуры | Triangle | bool |
| Холст | image | Bitmap^ |
| Поверхность для рисования | gr | Graphics^ |
| Массив, содержащий цвета для рисования линий | ge2 | Pen^ |
| Локальная переменная для хранения значения координаты х | x | int |
| Массив из точек для закрашивая определенную грань | point1 | Point^ |
| Массив из точек для закрашивая определенную грань | point2 | Point^ |
| Массив из точек для закрашивая определенную грань | point3 | Point^ |
| Перо для рисования фигуры | pen | Pen^ |
| Кисть для закрашивания фигуры | Bruh | Brush^ |

Методы, использующиеся в программе, описаны в табл.5.

Таблица 5

Функции

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Тип данных | Параметры | Выполняемое действие |
| timer\_Tick | void | System::Object^ sender, System::EventArgs^ e | Функция пересчитывает координаты вершин тетраэдра и рисует новую фигуру |
| onLoad | void | System::Object^ sender, System::EventArgs^ e | Функция присваивает переменным (height, width, angle, image, gr), находит середину холста и рисует треугольную призму |
| startR | void | System::Object^ sender, System::EventArgs^ e | Функция запускает таймер |
| stopR | void | System::Object^ sender, System::EventArgs^ e | Функция останавливает таймер |
| changeH | void | System::Object^ sender, System::EventArgs^ e | Функция изменяет высоту куба |
| changeW | void | System::Object^ sender, System::EventArgs^ e | Функция изменяет ширину куба |
| colorApp | void | System::Object^ sender, System::EventArgs^ e | Функция рисует треугольную призму и меняет цвет ребер |
| changeColor | void | System::Object^ sender, System::EventArgs^ e | Функция рисует треугольную призму и меняет цвет ребер |
| coordinate() | void | Отсутствует | Функция считает координаты вершин треугольной призмы в зависимости от введенных параметров |
| SpeedR | void | System::Object^ sender, System::EventArgs^ e | Изменяет скорость вращения фигуры |

## 3.4. Проектирование интерфейса

Для работы программы необходимы следующие элементы, они указаны в табл.6.

Таблица 6

Элементы интерфейса

|  |  |  |  |
| --- | --- | --- | --- |
| №  элемента | Наименование | Тип данных | Назначение |
| 1 | pictureBox | PictureBox^ | Холст |
| 2 | start | Button^ | Кнопка запуска вращения фигуры |
| 3 | stop | Button^ | Кнопка остановки вращения фигуры |
| 4 | label5 | Label^ | Текст «Выберите:» |
| 5 | Label | Label^ | Текст «Высота:» |
| 6 | numeric | NumericUpDown^ | Регулятор высоты |
| 7 | label1 | Label^ | Текст «Ширина:» |
| 8 | numeric1 | NumericUpDown^ | Регулятор ширины |
| 9 | label4 | Label^ | Текст «Скорость:» |
| 10 | RAngle | NumericUpDown^ | Регулятор скорости |
| 11 | label2 | Label^ | Текст «Цвет линий:» |
| 12 | comboBox1 | ComboBox^ | Выбор цвета линии фигуры |
| 13 | label3 | Label^ | Текст «Цвет фона:» |
| 14 | comboBox2 | ComboBox^ | Выбор цвета фона |

После того, как основные элементы определены, необходимо смоделировать интерфейс (рис.4.).

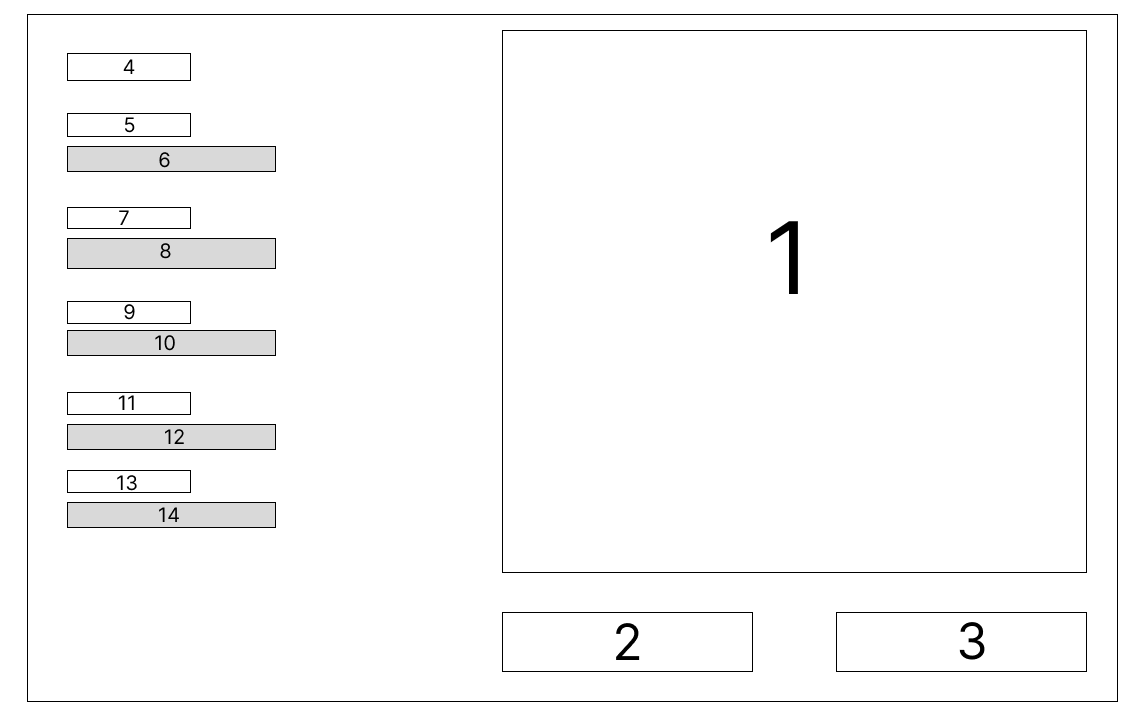


Рис.4. Макет

Окончательный вариант интерфейса (рис.5.).

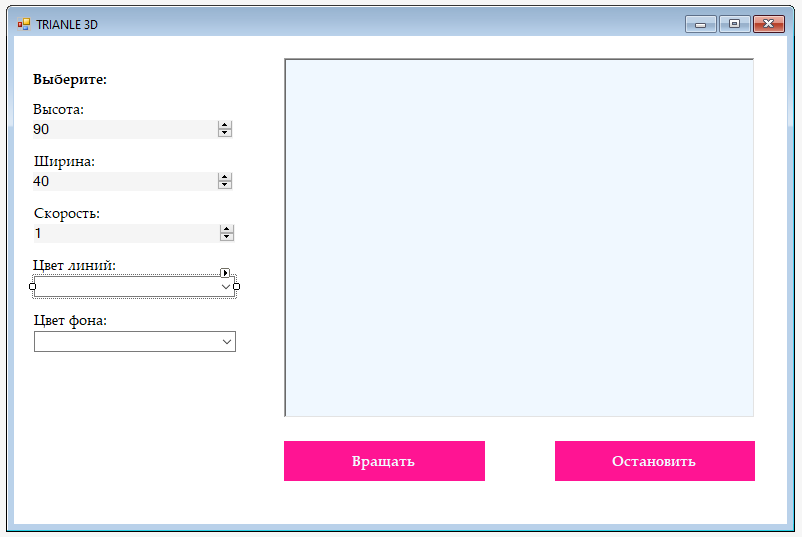


Рис.5. Интерфейс

## 3.5. Тестирование

Результаты тестирования представлены в табл.7.

Таблица 7

Тестирование

|  |  |  |  |
| --- | --- | --- | --- |
| Дата тестирования | Тестировщик | Описание теста | Результаты теста |
| 29.05.2023 | Рябкова А.Л. | Проверка работы программы на построение координат вершин и отрисовку тетраэдра при статических параметрах | Ошибки отсутствуют |
| 29.05.2023 | Рябкова А.Л. | Проверка работы программы на построение координат вершин и отрисовку тетраэдра при меняющихся параметрах | Ошибки отсутствуют |
| 29.05.2023 | Юдичев В.А. | Проверка работы программы на отрисовку и вращение тетраэдра | Ошибки отсутствуют |
| 30.05.2023 | Рябкова А.Л. | Проверка работы программы на отрисовку, изменение цветов, вращение и остановку вращения тетраэдра | Ошибки отсутствуют |
| 30.05.2023 | Смирнова П.Ю. | Проверка работы программы с внесенными изменениями | Ошибки отсутствуют |

# 

## 3.6. Результат работы

В результате курсовой работы было создано приложение, которое строит изображение тетраэдра и осуществляет его вращение (рис.6). В приложении есть возможность менять цвет фона и цвет ребер тетраэдра, можно менять скорость, с которой тетраэдр будет вращаться и изменять высоту и ширину фигуры.

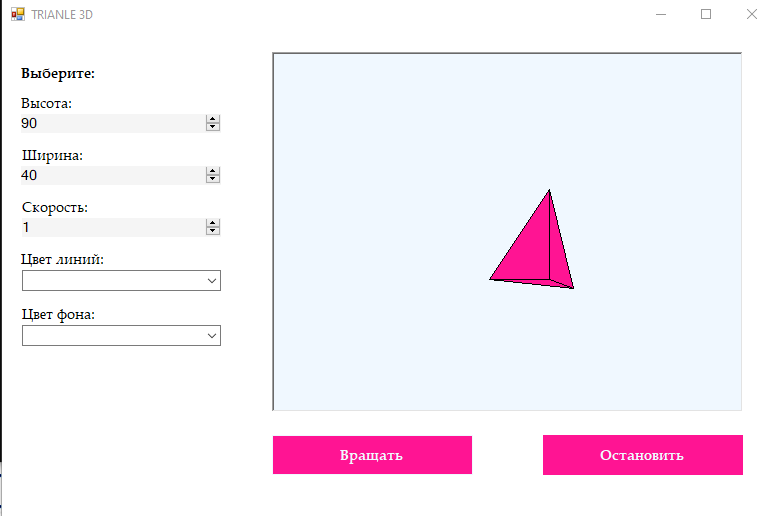


Рис.6. Интерфейс программы

# Заключение

В ходе выполнения курсовой работы была разработана программа на языке С++. В результате, созданное приложение работает корректно: строит изображение тетраэдра, меняет его высоту и ширину, скорость, цвет ребер и фона, может начинать и останавливать вращение. также были изучены необходимые элементы и возможности графического класса Graphics, Pen, Brush.

Считается, что программа должна помочь молодым пользователям, изучающим программирование и информатику поближе познакомится с данной отраслью нашей жизни.

Источники:

1. Pen Класс (System.Drawing) | Microsoft Docs – [электр.ресурс] - <https://docs.microsoft.com/ru-ru/dotnet/api/system.drawing.pen?view=dotnet-plat-ext-6.0.>

Дата обращения: 28.05.2023.

2. Brush Класс (System.Drawing) | Microsoft Docs – [электр.ресурс] - <https://docs.microsoft.com/ru-ru/dotnet/api/system.drawing.brush?view=dotnet-plat-ext-6.0>.

Дата обращения: 28.05.2023.

3. Timer Класс (Windows Forms) Microsoft Docs – [электр.ресурс] - <https://learn.microsoft.com/ruru/dotnet/api/system.windows.forms.timer?view=windowsdesktop-7.0>.

Дата обращения: 29.05.2023.

4. Поворот фигуры относительно заданной точки или начала координат – [электр. ресурс] - <https://blog.foolsoft.ru/c-povorot-figury-otnositelno-zadannoj-tochki-ili-nachala-koordinat/>.

Дата обращения: 29.05.2023.

5. Graphics Класс (System.Drawing) | Microsoft Docs – [электр.ресурс] - <https://learn.microsoft.com/ru-ru/dotnet/api/system.drawing.graphics?view=windowsdesktop-8.0>.

Дата обращения: 29.05.2023.

6. Тетраэдр | Википедия – [электр.ресурс] - <https://ru.wikipedia.org/wiki/Тетраэдр>.

Дата обращения: 28.05.2023.

7. Язык программирования C++ | Википедия – [электр.ресурс] - <https://ru.wikipedia.org/wiki/C%2B%2B>.

Дата обращения: 28.05.2023.

8. Windows Forms | Википедия – [электр.ресурс] - <https://ru.wikipedia.org/wiki/Windows_Forms>.

Дата обращения: 28.05.2023.

Приложение 1. Техническое задание

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий

наименование института (факультета)

Кафедра математического и программного обеспечения ЭВМ

наименование кафедры

Структурное программирование

наименование дисциплины в соответствии с учебным планом

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ

д.т.н., профессор Ершов Е.В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023г.

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ: ТЕТРАЭДР

Техническое задание на курсовую работу

Листов 8

Руководитель: Пышницкий К.М.

Исполнитель: студентка гр.

1ПИб-02-1оп-22

Рябкова А.Л.

Череповец

2023 год

Введение

Курсовой проект посвящен разработке программы, которая рисует тетраэдр, позволяет изменять параметры фигуры, цвет ребер и фона, вращать фигуру и изменять скорость её вращения.

1. Основания для разработки.

Основанием для разработки является задание на курсовую работу по дисциплине «Структурное программирование», выданное на кафедре МПО ЭВМ ИИТ ЧГУ.

Дата утверждения: 23 февраля 2023 года.

Наименование темы разработки: «Программирование на языке высокого уровня: Тетраэдр».

2. Назначение разработки

С помощью программы пользователь сможет построить двумерное представление трехмерного тетраэдра на холсте с заданными параметрами (длина ребра, цвет ребер, цвет фона) с возможностью вращения фигуры.

3. Требования к программе

3.1. Требования к функциональным характеристикам

Для выполнения программа должна соответствовать следующим требованиям:

1. Иметь набор значений по умолчанию или получать на вход пользовательские значения: длины ребра, цвета ребер, масштаба.
2. По выбранным данным строить тетраэдр.
3. Иметь возможность вращать построенную фигуру.

3.2. Требование к структуре программы

Программа должна иметь простой и понятный дизайн, а также соответствовать макету (рис. П1.1).

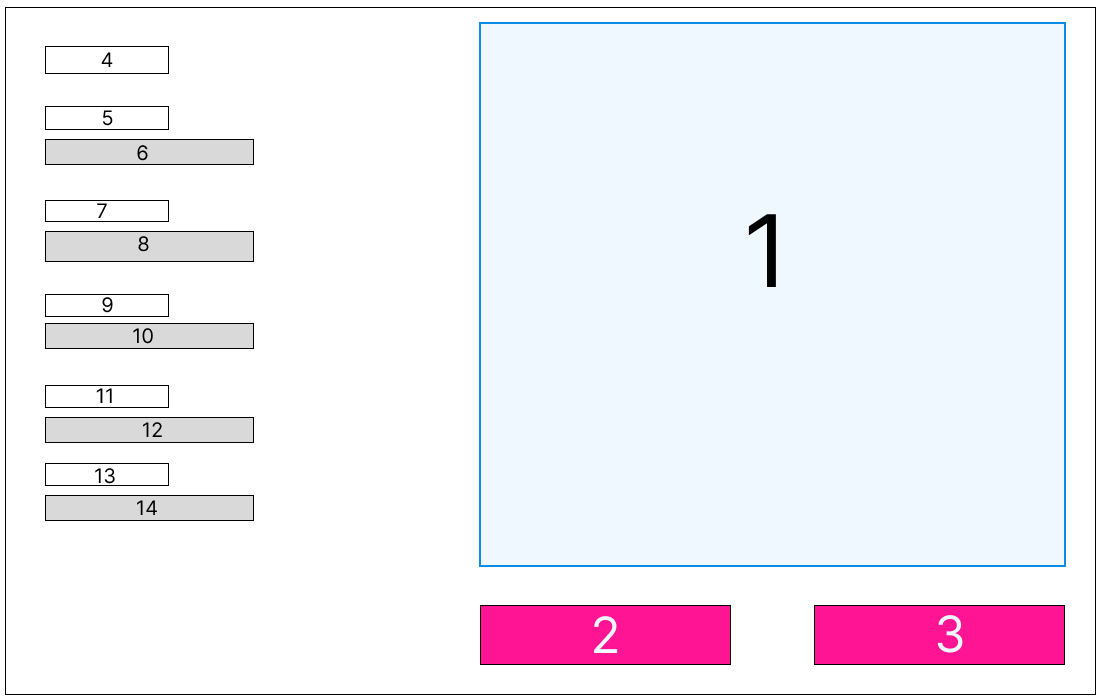


Рис. П1.1. Макет окна программы

3.3. Требования к надежности

* Если не были изменены значения, то построение призмы происходит с параметрами по умолчанию (Высота – 90, ширина – 40, скорость - 1, цвет ребер черный, цвет фона «AliceBlue»);
* При вводе некорректных значений сбой программы отсутствует.

3.4. Условия эксплуатации

Для правильной работы программы необходимо устройство ПК с программным обеспечением Windows 7 и выше.

3.5. Требования к составу и параметрам технических средств

Для нормального функционирования программного средства минимальный состав и параметры технических средств должны соответствовать следующему:

* процессор с тактовой частотой не менее 2000MHz, частота 2,3 Ghz;
* архитектура с разрядностью 32 бит или 64 бит;
* оперативная память 2 ГБ и выше;
* свободное место на жёстком диске от 120 МБ;
* наличие монитора (для персонального компьютера);
* наличие компьютерной мыши;
* наличие клавиатуры.

3.6. Требования к информационной и программной совместимости

Код программы написан на языке высокого уровня C++ и скомпилирован в Visual Studio 2012 на оперативной системе Windows 10. Для пользования программой необходимо устройство, имеющее операционную систему (Windows 7 и выше).

3.7. Требования к маркировке и упаковке

Отсутствуют.

3.8. Требования к транспортированию и хранению.

Файлы, которые необходимы для корректной работы, необходимо записать на диск, либо в облачном хранилище, это требуется для того, чтобы предотвратить утерю данных.

3.9. Специальные требования

Разрабатываемая программа должна соответствовать следующим требованиям:

* удобный и понятный интерфейс;
* строить фигуру без ошибок;
* форма для выбора цвета ребер (красный, коричневый, зелёный, фиолетовый, темно-синий, чёрный);
* форма для выбора цвета фона (оранжевый, фиолетовый, коралловый, белый, светло-зеленый, светло-голубой);
* форма для выбора высоты тетраэдра (значения от 50 до 150);
* форма для выбора ширины тетраэдра (значения от 20 до 100);
* форма для выбора скорости вращения тетраэдра (значения от 1 до 6);
* кнопка для запуска вращения фигуры;
* кнопка для остановки вращения фигуры;

4. Требование к программной документации

4.1. Содержание расчётно-пояснительной записки

Программная документация должна содержать расчетно-пояснительную записку с содержанием:

Титульный лист

Аннотация

Оглавление

Введение

1. Описание предметной области

2. Описание классов Graphics, Pen, Brush

3. Описание разработанного приложения

Заключение

Источники

Приложения

4.2. Требования к содержательной части

В описываемой работе должны быть применены следующие конструкции:

4.2.1. Класс Graphics;

4.2.2. Класс Pen;

4.2.3. Класс Brush.

4.3. Требования к оформлению

Требования к оформлению, установленные ГОСТ, должны быть выполнены на протяжении всей работы без каких-либо изменений. Требования к оформлению представлены в табл. П1.1.

Таблица П1.1

Требования к оформлению

|  |  |
| --- | --- |
| Документ | Печать на отдельных листах формата А4 (210х297 мм); оборотная сторона не заполняется; листы нумеруются. Печать возможна ч/б.  Файлы предъявляются на компакт-диске: РПЗ с ТЗ; программный код.  Листы и диск в конверте вложены в пластиковую папку скоросшивателя. |
| Страницы | Ориентация – книжная; отдельные страницы, при необходимости, альбомная.  Поля: верхнее, нижнее – по 2 см, левое – 3 см, правое – 1 см. |
| Абзацы | Межстрочный интервал – 1.5, перед и после абзаца – 0. |
| Шрифты | Кегль – 14. В таблицах шрифт 12. Шрифт листинга – 10 (возможно в 2 колонки). |
| Рисунки | Подписывается под ним по центру: Рис.Х. Название  В приложениях: Рис.П1.3. Название |
| Таблицы | Подписывается: над таблицей, выравнивание по правому: «Таблица Х».  В следующей строке по центру Название  Надписи в «шапке» (имена столбцов, полей) – по центру.  В теле таблицы (записи) текстовые значения – выровнены по левому краю, числа, даты – по правому. |

5. Стадии и этапы разработки

Стадии и этапы разработки представлены в табл. П1.2.

Таблица П1.2

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки | Сроки разработки | Результат выполнения | Отметка о выполнении |
| Определение темы для курсовой работы | 23.02.2023 | Утверждена тема для разработки |  |
| Поиск информации, необходимой для написания программы | 28.05.2023 | Найдены все необходимые материалы |  |
| Оформление технического задания | 01.06.2023-03.06.2023 | Оформленное техническое задание |  |
| Написание программы | 29.05.2023-30.05.2023 | Доработанная программа |  |
| Тестирование программы и доработка недочетов | 29.05.2023-31.05.2023 | Ошибки при тестировании отсутствуют |  |
| Оформление сопроводительной документации - расчётно-пояснительной записки (РПЗ) | 01.06.2023-03.06.2023 | Выполненное РПЗ |  |

6. Порядок контроля и приемки

Порядок контроля и приемки представлены в таблице П1.3

Таблица П1.3

Порядок контроля и приемки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование контрольного этапа выполнения курсовой работы | Сроки контроля | Результат выполнения | Отметка о приемке результата контрольного этапа |
| Разработка технического задания | 28.05.2023 | Оформленное техническое задание |  |
| Описание алгоритма программы | 29.05.2023 | Составленный алгоритм работы приложения |  |
| Написание программы | 29.05.2023 – 30.05.2023 | Конечная версия программы |  |
| Оформление сопроводительной документации - РПЗ | 01.06.2023 – 03.06.2023 | Оформленная РПЗ |  |
| Сдача РПЗ, оценка | 06.06.2023 | Итоговая оценка за курсовую работу |  |

Приложение 2. Руководство пользования

Запуск программы

Двойным щелчком левой кнопкой мыши по ярлыку программы «Kurs.exe» (рис. П2.1), производится запуск программы.

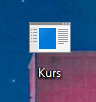


Рис. П2.1. Ярлык программы

Представленный интерфейс

Интерфейс программы (рис. П2.2) очень прост в использовании. Каждой кнопки соответствует своё логическое сопровождение.

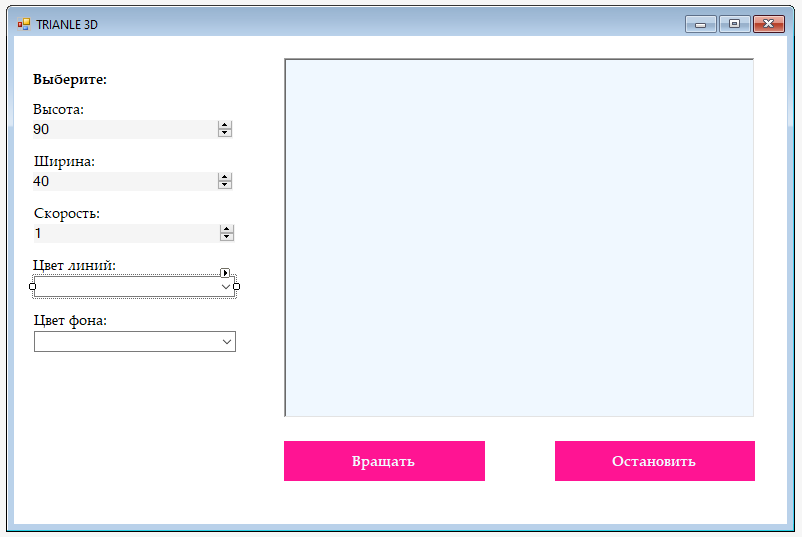


Рис. П2.2. Интерфейс программы

Для поля с выбором или вводом значения параметров тетраэдра есть ограничения по диапазону: высота – от 50 до 150 (по умолчанию 90), ширина – от 20 до 100 (по умолчанию 40). Так же можно настроить цвет фона и цвет ребер фигуры. При запуске можно будет изменять скорость вращения фигуры, значения от 1 до 6 (по умолчанию 1) (рис. П2.3).

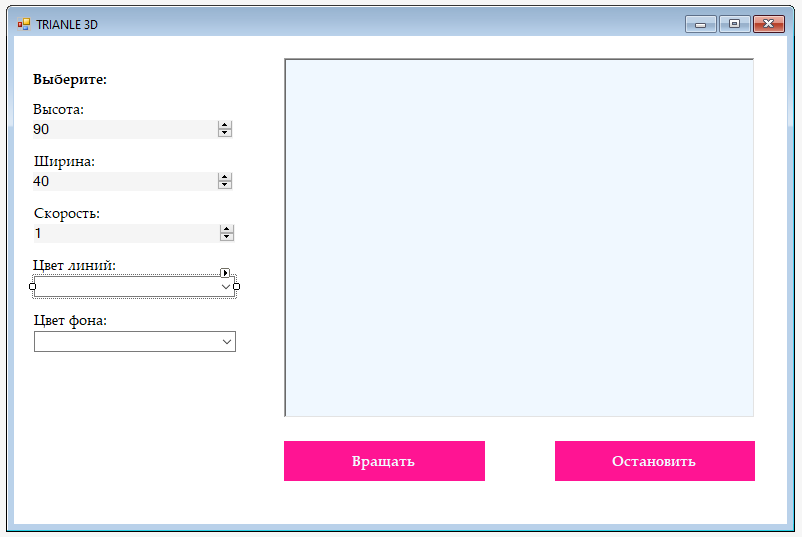


Рис. П2.3. Блок с параметрами тетраэдра

Кнопка «Вращать» начинает вращать фигуру, кнопка «Остановить», соответственно, её останавливает (рис. П2.4)

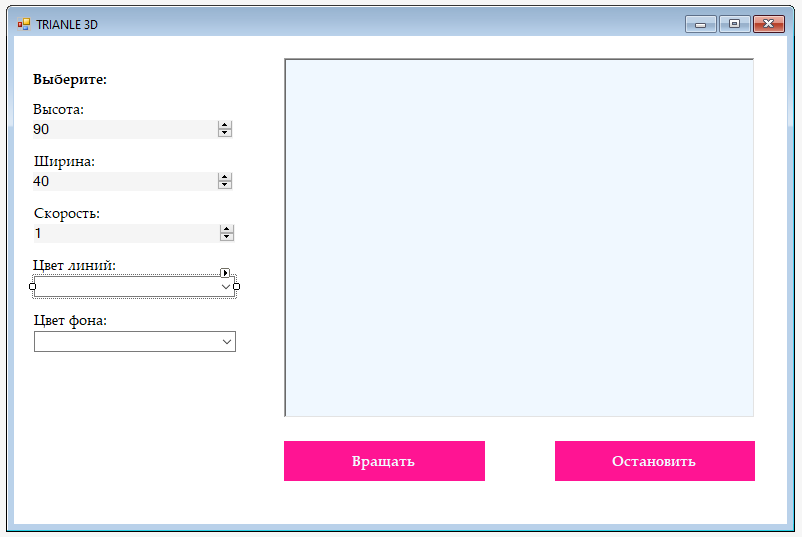


Рис. П2.4. Блок с кнопками вращения

При выполнении построения с параметрами по умолчанию программа принимает вид, показанный на рисунке П2.5.

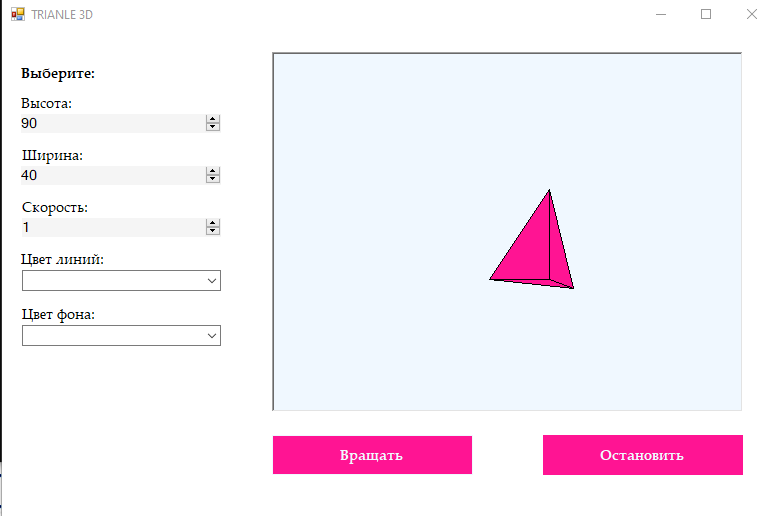


Рис. П2.5. Работающая программа

На этом функционал программы заканчивается, при завершении работы, программа просто закрывается, как и любое другое приложение.

Приложение 3. Программный код

Код файла «MyForm.cpp»:

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

int main(array<String^> ^ args)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Kurs::MyForm form;

Application::Run(%form);

}

Код файла «MyForm.h»:

#pragma once

namespace Kurs {

#include <math.h>

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace System::Drawing::Drawing2D;

using namespace System::Drawing::Imaging;

int height, width, center[2], angle;

float p1[2], p2[2], p3[2], p4[2];

bool Triangle = true;

/// <summary>

/// Сводка для MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

Pen^ pen;

Bitmap^ image;

Graphics^ gr;

SolidBrush^ brush;

private: System::Windows::Forms::Button^ stop;

private: System::Windows::Forms::Label^ Label;

private: System::Windows::Forms::NumericUpDown^ numeric;

private: System::Windows::Forms::Label^ label6;

private: System::Windows::Forms::NumericUpDown^ numeric1;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::ComboBox^ comboBox1;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::ComboBox^ comboBox2;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::NumericUpDown^ RAngle;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::Button^ start;

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MyForm()

{

if (components)

{

delete components;}}

protected:

private: System::Windows::Forms::PictureBox^ pictureBox;

private: System::Windows::Forms::Timer^ timer;

private: System::ComponentModel::IContainer^ components;

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

this->components = (gcnew System::ComponentModel::Container());

this->pictureBox = (gcnew System::Windows::Forms::PictureBox());

this->timer = (gcnew System::Windows::Forms::Timer(this->components));

this->start = (gcnew System::Windows::Forms::Button());

this->stop = (gcnew System::Windows::Forms::Button());

this->Label = (gcnew System::Windows::Forms::Label());

this->numeric = (gcnew System::Windows::Forms::NumericUpDown());

this->numeric1 = (gcnew System::Windows::Forms::NumericUpDown());

this->label1 = (gcnew System::Windows::Forms::Label());

this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());

this->label2 = (gcnew System::Windows::Forms::Label());

this->comboBox2 = (gcnew System::Windows::Forms::ComboBox());

this->label3 = (gcnew System::Windows::Forms::Label());

this->RAngle = (gcnew System::Windows::Forms::NumericUpDown());

this->label4 = (gcnew System::Windows::Forms::Label());

this->label5 = (gcnew System::Windows::Forms::Label());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^ >(this->pictureBox))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^ >(this->numeric))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^ >(this->numeric1))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^ >(this->RAngle))->BeginInit();

this->SuspendLayout();

//

// pictureBox

//

this->pictureBox->BackColor = System::Drawing::Color::AliceBlue;

this->pictureBox->BackgroundImageLayout = System::Windows::Forms::ImageLayout::None;

this->pictureBox->BorderStyle = System::Windows::Forms::BorderStyle::Fixed3D;

this->pictureBox->Location = System::Drawing::Point(270, 22);

this->pictureBox->Name = L"pictureBox";

this->pictureBox->Size = System::Drawing::Size(471, 360);

this->pictureBox->TabIndex = 2;

this->pictureBox->TabStop = false;

//

// timer

//

this->timer->Tick += gcnew System::EventHandler(this, &MyForm::timer\_Tick);

//

// start

//

this->start->BackColor = System::Drawing::Color::DeepPink;

this->start->FlatAppearance->BorderSize = 0;

this->start->FlatStyle = System::Windows::Forms::FlatStyle::Flat;

this->start->Font = (gcnew System::Drawing::Font(L"Palatino Linotype", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->start->ForeColor = System::Drawing::Color::AliceBlue;

this->start->Location = System::Drawing::Point(270, 405);

this->start->Name = L"start";

this->start->Size = System::Drawing::Size(201, 40);

this->start->TabIndex = 3;

this->start->Text = L"Вращать";

this->start->UseVisualStyleBackColor = false;

this->start->Click += gcnew System::EventHandler(this, &MyForm::startR);

//

// stop

//

this->stop->BackColor = System::Drawing::Color::DeepPink;

this->stop->FlatAppearance->BorderSize = 0;

this->stop->FlatStyle = System::Windows::Forms::FlatStyle::Flat;

this->stop->Font = (gcnew System::Drawing::Font(L"Palatino Linotype", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->stop->ForeColor = System::Drawing::Color::AliceBlue;

this->stop->Location = System::Drawing::Point(541, 405);

this->stop->Name = L"stop";

this->stop->Size = System::Drawing::Size(200, 40);

this->stop->TabIndex = 4;

this->stop->Text = L"Остановить";

this->stop->UseVisualStyleBackColor = false;

this->stop->Click += gcnew System::EventHandler(this, &MyForm::stopR);

//

// Label

//

this->Label->AutoSize = true;

this->Label->BackColor = System::Drawing::Color::White;

this->Label->FlatStyle = System::Windows::Forms::FlatStyle::Flat;

this->Label->Font = (gcnew System::Drawing::Font(L"Palatino Linotype", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->Label->ForeColor = System::Drawing::SystemColors::Desktop;

this->Label->Location = System::Drawing::Point(15, 63);

this->Label->Margin = System::Windows::Forms::Padding(0);

this->Label->Name = L"Label";

this->Label->Size = System::Drawing::Size(60, 19);

this->Label->TabIndex = 9;

this->Label->Text = L"Высота:";

//

// numeric

//

this->numeric->BackColor = System::Drawing::Color::WhiteSmoke;

this->numeric->BorderStyle = System::Windows::Forms::BorderStyle::None;

this->numeric->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10.18868F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->numeric->Location = System::Drawing::Point(19, 84);

this->numeric->Margin = System::Windows::Forms::Padding(2);

this->numeric->Maximum = System::Decimal(gcnew cli::array< System::Int32 >(4) {150, 0, 0, 0});

this->numeric->Minimum = System::Decimal(gcnew cli::array< System::Int32 >(4) {50, 0, 0, 0});

this->numeric->Name = L"numeric";

this->numeric->Size = System::Drawing::Size(200, 19);

this->numeric->TabIndex = 11;

this->numeric->Value = System::Decimal(gcnew cli::array< System::Int32 >(4) {90, 0, 0, 0});

this->numeric->ValueChanged += gcnew System::EventHandler(this, &MyForm::changeH);

//

// numeric1

//

this->numeric1->BackColor = System::Drawing::Color::WhiteSmoke;

this->numeric1->BorderStyle = System::Windows::Forms::BorderStyle::None;

this->numeric1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10.18868F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->numeric1->Location = System::Drawing::Point(19, 136);

this->numeric1->Margin = System::Windows::Forms::Padding(2);

this->numeric1->Minimum = System::Decimal(gcnew cli::array< System::Int32 >(4) {20, 0, 0, 0});

this->numeric1->Name = L"numeric1";

this->numeric1->Size = System::Drawing::Size(200, 19);

this->numeric1->TabIndex = 11;

this->numeric1->Value = System::Decimal(gcnew cli::array< System::Int32 >(4) {40, 0, 0, 0});

this->numeric1->ValueChanged += gcnew System::EventHandler(this, &MyForm::changeW);

//

// label1

//

this->label1->AutoSize = true;

this->label1->BackColor = System::Drawing::Color::White;

this->label1->FlatStyle = System::Windows::Forms::FlatStyle::Flat;

this->label1->Font = (gcnew System::Drawing::Font(L"Palatino Linotype", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label1->Location = System::Drawing::Point(16, 115);

this->label1->Margin = System::Windows::Forms::Padding(0);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(70, 19);

this->label1->TabIndex = 14;

this->label1->Text = L"Ширина:";

//

// comboBox1

//

this->comboBox1->BackColor = System::Drawing::SystemColors::Window;

this->comboBox1->FormattingEnabled = true;

this->comboBox1->Items->AddRange(gcnew cli::array< System::Object^ >(6) {L"Red", L"Brown", L"Green", L"Purple", L"Dark Blue",

L"Black"});

this->comboBox1->Location = System::Drawing::Point(20, 240);

this->comboBox1->Margin = System::Windows::Forms::Padding(2);

this->comboBox1->Name = L"comboBox1";

this->comboBox1->Size = System::Drawing::Size(199, 21);

this->comboBox1->TabIndex = 15;

this->comboBox1->SelectedIndexChanged += gcnew System::EventHandler(this, &MyForm::colorApp);

//

// label2

//

this->label2->AutoSize = true;

this->label2->FlatStyle = System::Windows::Forms::FlatStyle::Flat;

this->label2->Font = (gcnew System::Drawing::Font(L"Palatino Linotype", 10.2F));

this->label2->ForeColor = System::Drawing::SystemColors::Desktop;

this->label2->Location = System::Drawing::Point(15, 219);

this->label2->Margin = System::Windows::Forms::Padding(0);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(92, 19);

this->label2->TabIndex = 16;

this->label2->Text = L"Цвет линий:";

//

// comboBox2

//

this->comboBox2->FormattingEnabled = true;

this->comboBox2->Items->AddRange(gcnew cli::array< System::Object^ >(6) {L"Orange", L"Violet", L"Coral", L"White", L"Light Green",

L"Moccasin"});

this->comboBox2->Location = System::Drawing::Point(20, 295);

this->comboBox2->Margin = System::Windows::Forms::Padding(2);

this->comboBox2->Name = L"comboBox2";

this->comboBox2->Size = System::Drawing::Size(199, 21);

this->comboBox2->TabIndex = 17;

this->comboBox2->SelectedIndexChanged += gcnew System::EventHandler(this, &MyForm::changeColor);

//

// label3

//

this->label3->AutoSize = true;

this->label3->FlatStyle = System::Windows::Forms::FlatStyle::Flat;

this->label3->Font = (gcnew System::Drawing::Font(L"Palatino Linotype", 10.2F));

this->label3->ForeColor = System::Drawing::SystemColors::Desktop;

this->label3->Location = System::Drawing::Point(16, 274);

this->label3->Margin = System::Windows::Forms::Padding(0);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(82, 19);

this->label3->TabIndex = 18;

this->label3->Text = L"Цвет фона:";

//

// RAngle

//

this->RAngle->BackColor = System::Drawing::Color::WhiteSmoke;

this->RAngle->BorderStyle = System::Windows::Forms::BorderStyle::None;

this->RAngle->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->RAngle->Location = System::Drawing::Point(20, 188);

this->RAngle->Margin = System::Windows::Forms::Padding(2);

this->RAngle->Maximum = System::Decimal(gcnew cli::array< System::Int32 >(4) {6, 0, 0, 0});

this->RAngle->Minimum = System::Decimal(gcnew cli::array< System::Int32 >(4) {1, 0, 0, 0});

this->RAngle->Name = L"RAngle";

this->RAngle->Size = System::Drawing::Size(199, 19);

this->RAngle->TabIndex = 19;

this->RAngle->Value = System::Decimal(gcnew cli::array< System::Int32 >(4) {1, 0, 0, 0});

this->RAngle->ValueChanged += gcnew System::EventHandler(this, &MyForm::SpeedR);

//

// label4

//

this->label4->AutoSize = true;

this->label4->BackColor = System::Drawing::Color::White;

this->label4->FlatStyle = System::Windows::Forms::FlatStyle::Flat;

this->label4->Font = (gcnew System::Drawing::Font(L"Palatino Linotype", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label4->Location = System::Drawing::Point(16, 167);

this->label4->Margin = System::Windows::Forms::Padding(0);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(75, 19);

this->label4->TabIndex = 20;

this->label4->Text = L"Скорость:";

//

// label5

//

this->label5->AutoSize = true;

this->label5->BackColor = System::Drawing::Color::White;

this->label5->FlatStyle = System::Windows::Forms::FlatStyle::Flat;

this->label5->Font = (gcnew System::Drawing::Font(L"Palatino Linotype", 10.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label5->ForeColor = System::Drawing::SystemColors::Desktop;

this->label5->Location = System::Drawing::Point(15, 33);

this->label5->Margin = System::Windows::Forms::Padding(0);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(83, 19);

this->label5->TabIndex = 21;

this->label5->Text = L"Выберите:";

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackColor = System::Drawing::Color::White;

this->ClientSize = System::Drawing::Size(773, 488);

this->Controls->Add(this->label5);

this->Controls->Add(this->label4);

this->Controls->Add(this->RAngle);

this->Controls->Add(this->label3);

this->Controls->Add(this->comboBox2);

this->Controls->Add(this->label2);

this->Controls->Add(this->comboBox1);

this->Controls->Add(this->label1);

this->Controls->Add(this->numeric1);

this->Controls->Add(this->numeric);

this->Controls->Add(this->Label);

this->Controls->Add(this->stop);

this->Controls->Add(this->start);

this->Controls->Add(this->pictureBox);

this->ForeColor = System::Drawing::SystemColors::ActiveCaptionText;

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedSingle;

this->Name = L"MyForm";

this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;

this->Text = L"TRIANLE 3D";

this->Load += gcnew System::EventHandler(this, &MyForm::onLoad);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^ >(this->pictureBox))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^ >(this->numeric))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^ >(this->numeric1))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^ >(this->RAngle))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

public: System::Void onLoad(System::Object^ sender, System::EventArgs^ e)

{

height = (int)numeric->Value;

width = (int)numeric1->Value;

center[0] = pictureBox->Width / 2;

center[1] = pictureBox->Height / 2;

image = gcnew Bitmap(pictureBox->Width, pictureBox->Height);

pictureBox->Image = image;

gr = Graphics::FromImage(image);

colorApp(sender, e);

}

private: System::Void coordinate()

{

p1[0] = center[0] - width / 2;

p1[1] = center[1]+ height / 2;

p2[0] = center[0] + width;

p2[1] = center[1] + height / 2;

p3[0] = center[0] + (1.6 \* width);

p3[1] = center[1] + (0.6 \* height);

p4[0] = center[0] + width;

p4[1] = center[1] - height / 2;

}

private: System::Void timer\_Tick(System::Object^ sender, System::EventArgs^ e)

{

angle = (int)RAngle->Value;

float alpha((-15 \* angle) \* 3.14 / 360);

gr->Clear(pictureBox->BackColor);

float oldp = p1[0];

p1[0] = (oldp - center[0]) \* cos(alpha) - (p1[1] - center[1]) \* sin(alpha) + center[0];

p1[1] = (oldp - center[0]) \* sin(alpha) + (p1[1] - center[1]) \* cos(alpha) + center[1];

oldp = p2[0];

p2[0] = (oldp - center[0]) \* cos(alpha) - (p2[1] - center[1]) \* sin(alpha) + center[0];

p2[1] = (oldp - center[0]) \* sin(alpha) + (p2[1] - center[1]) \* cos(alpha) + center[1];

oldp = p3[0];

p3[0] = (oldp - center[0]) \* cos(alpha) - (p3[1] - center[1]) \* sin(alpha) + center[0];

p3[1] = (oldp - center[0]) \* sin(alpha) + (p3[1] - center[1]) \* cos(alpha) + center[1];

oldp = p4[0];

p4[0] = (oldp - center[0]) \* cos(alpha) - (p4[1] - center[1]) \* sin(alpha) + center[0];

p4[1] = (oldp - center[0]) \* sin(alpha) + (p4[1] - center[1]) \* cos(alpha) + center[1];

Triangle = false;

colorApp(sender, e);

Triangle = true;

pictureBox->Refresh();

}

private: System::Void startR(System::Object^ sender, System::EventArgs^ e)

{

timer->Enabled = true;

}

private: System::Void stopR(System::Object^ sender, System::EventArgs^ e)

{

timer->Enabled = false;

}

private: System::Void changeH(System::Object^ sender, System::EventArgs^ e)

{

height = (int)numeric->Value;

if (timer->Enabled) coordinate();

else colorApp(sender, e);

}

private: System::Void changeW(System::Object^ sender, System::EventArgs^ e)

{

width = (int)numeric1->Value;

if (timer->Enabled) coordinate();

else colorApp(sender, e);

}

private: System::Void colorApp(System::Object^ sender, System::EventArgs^ e) {

gr->Clear(pictureBox->BackColor);

array <Brush^>^ Bruh = gcnew array <Brush^> {Brushes::DeepPink};

array<Pen^>^ ge2 = gcnew array <Pen^>{ Pens::Red, Pens::Brown, Pens::Green, Pens::Purple, Pens::DarkBlue, Pens::Black};

int x;

if (comboBox1->SelectedIndex == 0) {

x = 0;

}

else if (comboBox1->SelectedIndex == 1) {

x = 1;

}

else if (comboBox1->SelectedIndex == 2) {

x = 2;

}

else if (comboBox1->SelectedIndex == 3) {

x = 3;

}

else if (comboBox1->SelectedIndex == 4) {

x = 4;

}

else x = 5;

if (Triangle) {

coordinate();

}

array<Point>^ points1 = gcnew array<Point>{ Point(p1[0], p1[1]), Point(p2[0], p2[1]), Point(p3[0], p3[1])};

gr->FillPolygon(Bruh[0], points1);

array<Point>^ points2 = gcnew array<Point>{ Point(p1[0], p1[1]), Point(p2[0], p2[1]), Point(p4[0], p4[1])};

gr->FillPolygon(Bruh[0], points2);

array<Point>^ points3 = gcnew array<Point>{ Point(p2[0], p2[1]), Point(p3[0], p3[1]), Point(p4[0], p4[1])};

gr->FillPolygon(Bruh[0], points3);

gr->DrawLine(ge2[x], p1[0], p1[1], p2[0], p2[1]);

gr->DrawLine(ge2[x], p1[0], p1[1], p3[0], p3[1]);

gr->DrawLine(ge2[x], p1[0], p1[1], p4[0], p4[1]);

gr->DrawLine(ge2[x], p2[0], p2[1], p3[0], p3[1]);

gr->DrawLine(ge2[x], p2[0], p2[1], p4[0], p4[1]);

gr->DrawLine(ge2[x], p3[0], p3[1], p4[0], p4[1]);

pictureBox->Refresh();}

private: System::Void changeColor(System::Object^ sender, System::EventArgs^ e) {

if (comboBox2->SelectedIndex == 0) {

pictureBox->BackColor = Color::Orange;}

else if (comboBox2->SelectedIndex == 1) {

pictureBox->BackColor = Color::Violet;}

else if (comboBox2->SelectedIndex == 2) {

pictureBox->BackColor = Color::Coral;}

else if (comboBox2->SelectedIndex == 3) {

pictureBox->BackColor = Color::White;}

else if (comboBox2->SelectedIndex == 4) {

pictureBox->BackColor = Color::LightGreen;}

else if (comboBox2->SelectedIndex == 5) {

pictureBox->BackColor = Color::AliceBlue;}

gr->Clear(pictureBox->BackColor);

if (!timer->Enabled) {

Triangle = false;

colorApp(sender, e);

Triangle = true;

}

pictureBox->Refresh();

}

private: System::Void SpeedR(System::Object^ sender, System::EventArgs^ e) {}

};

}