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 |

|  |
| --- |
| Выполнил студент группы |
| 1ПИб-02-3оп-22 |
| направление подготовки (специальности) |
| 09.03.04, Программная инженерия |
| *шифр, наименование* |
| Маркелов Сергей Александрович |
| *фамилия, имя, отчество* |

|  |
| --- |
| Руководитель |
| Журавлева Юлия Михайловна |
| *фамилия, имя, отчество* |
| старший преподаватель |
| *должность* |

|  |
| --- |
| Дата представления работы |
| «\_\_07\_\_»\_\_\_\_\_\_\_июня\_\_\_\_\_\_\_\_2023 г. |
|  |
| Заключение о допуске к защите |
|  |
|  |
|  |
|  |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| количество баллов |
| Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Череповец, 2023

*Год*

Оглавление

[Введение 3](#_Toc137046927)

[1. Описание предметной области 5](#_Toc137046928)

[2. Описание классов Graphics, Pen и Brush 8](#_Toc137046929)

[2.1. Класс Graphics 8](#_Toc137046930)

[2.2. Класс Pen 9](#_Toc137046931)

[2.3. Класс Brush 9](#_Toc137046932)

[3. Описание созданного приложения 10](#_Toc137046933)

[3.1. Постановка задачи 10](#_Toc137046934)

[3.2. Логическое проектирование – алгоритм работы 10](#_Toc137046935)

[3.3. Физическое проектирование – структуры данных и спецификация функций 14](#_Toc137046936)

[3.4. Описание интерфейса пользователя 16](#_Toc137046937)

[3.5. Тестирование 18](#_Toc137046938)

[3.6. Результат работы 19](#_Toc137046939)

[Заключение 20](#_Toc137046940)

[Список литературы 21](#_Toc137046941)

[Приложение 1. Техническое задание 22](#_Toc137046942)

[Приложение 2. Руководство пользователя 29](#_Toc137046943)

[Приложение 3. Программный код 34](#_Toc137046944)

# Введение

Программы играют огромную роль в современном мире. Существует большое количество различных видов программ: драйвера, утилиты, плееры, браузеры, различные редакторы (текстовые, графические, музыкальные видеоредакторы), мессенджеры, архиваторы, антивирусы и многие другие.

Благодаря программам мы можем смотреть, обрабатывать или создавать фотографии, видео, музыку, работать с документами, обрабатывать данные и многое другое. Также существует отдельный вид программ, который позволяет решать некоторые математические задачи – это различные калькуляторы, программы для выполнения сложных расчетов, построения графиков и т. д.

Программы создаются благодаря языкам программирования – формальным знаковым системам, предназначенным для описания алгоритмов обработки данных. Каждый язык программирования имеет свой алфавит и синтаксис [[3]](#_Источники).

Одним из наиболее популярных языков программирования является C++ –компилируемый язык программирования высокого уровня. Он имеет большую стандартную библиотеку, в которую входят, например, различные контейнеры и алгоритмы, функции ввода-вывода, регулярные выражения и некоторые другие возможности. Язык программирования C++ сочетает свойства и высокоуровневых, и низкоуровневых языков. Также в C++ есть поддержка объектно-ориентированного программирования. С помощью языка программирования C++ можно написать большое количество самых различных программ. С его помощью создаются операционные системы, прикладные программы, драйвера, а также компьютерные игры [[4]](#_Источники).

Целью курсовой работы является знакомство с принципами работы с графикой в языке программирования C++ и создание программы, которая выводит на экран трехмерное изображение шестиугольной пирамиды. Программа написана на языке программирования C++ с применением Windows Forms [[5]](#_Источники).

Необходимость такой программы заключается в том, что зачастую при решении каких-либо задач по геометрии требуется чертеж для более легкого восприятия условий задачи. Однако, у некоторых людей (особенно у тех, у кого плохо развито пространственное мышление) могут возникнуть проблемы с построением чертежей трехмерных фигур. С появлением программы пользователи смогут не строить чертеж вручную, а увидеть его на экране.

# Описание предметной области

Предметной областью программы является трехмерная геометрическая фигура – пирамида.

Пирамида – многогранник, одна из граней которого (основание) является произвольным многоугольником, а все остальные грани (боковые) – это треугольники, имеющие одну общую вершину. Бывают пирамиды треугольные (тетраэдр), четырёхугольные, пятиугольные, шестиугольные и т. д.

Если у пирамиды в основании лежит правильный многоугольник (т.е. многоугольник, у которого все стороны и все углы равны), а высота падает в центр основания, то такая пирамида называется правильной (рис. 1). Именно такую пирамиду и будет выводить на экран программа [[6]](#_Источники).

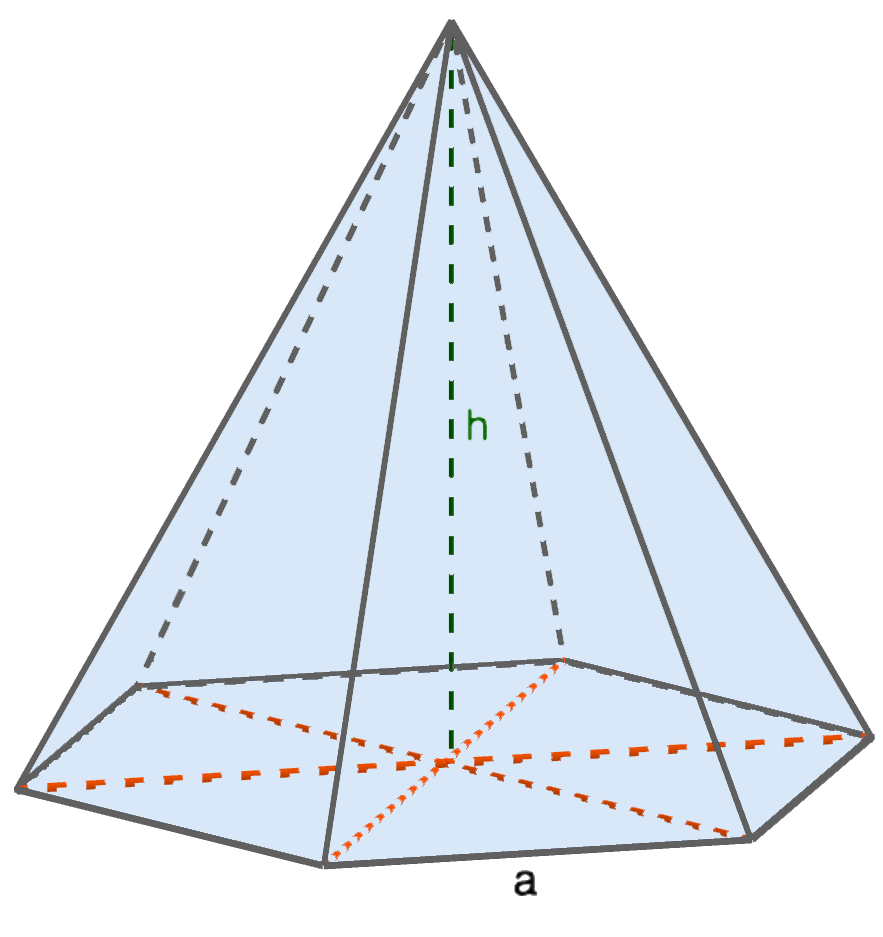


Рис. 1. Правильная шестиугольная пирамида

Вокруг основания правильной пирамиды можно описать окружность (рис. 2). Если чертить пирамиду в диметрической проекции, окружность на чертеже будет отображаться в виде эллипса (рис. 3).

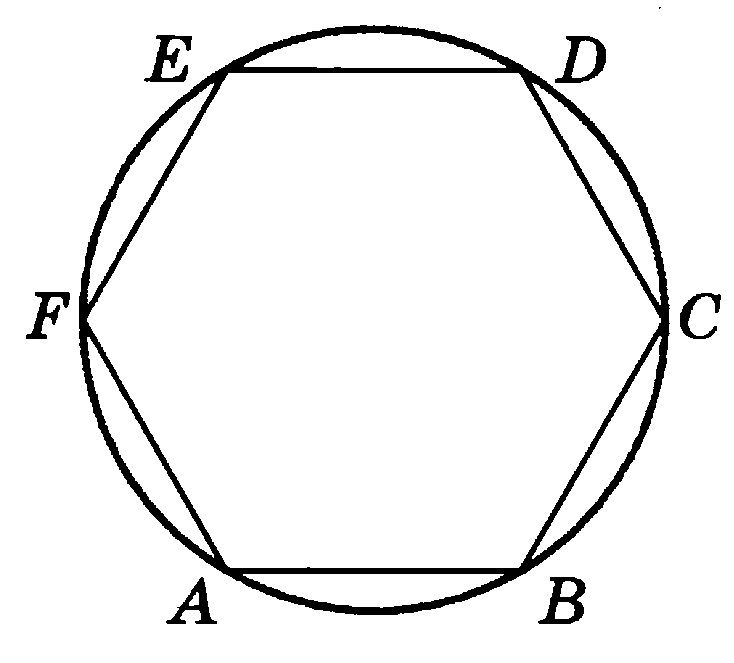


Рис. 2. Правильный шестиугольник с описанной вокруг него окружностью

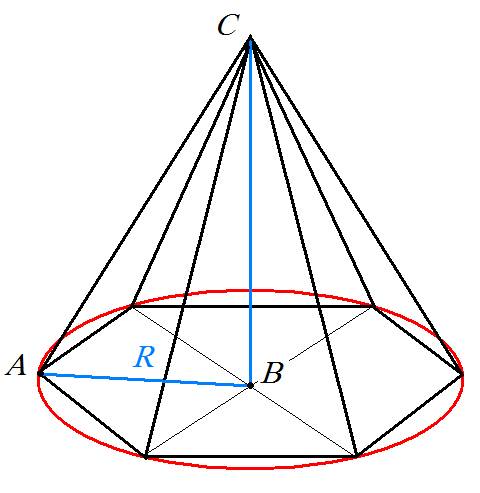


Рис. 3. Шестиугольная правильная пирамида с описанной вокруг основания окружностью

Это значит, что вершины основания на чертеже при вращении будут двигаться по эллипсоидной траектории. Для того, чтобы реализовать движение, в программе понадобится использовать уравнение эллипса (1.1):

где x, y – координаты произвольной точки эллипса, a – длина большой полуоси, b – длина малой полуоси (рис. 4) [[7]](#_Источники).

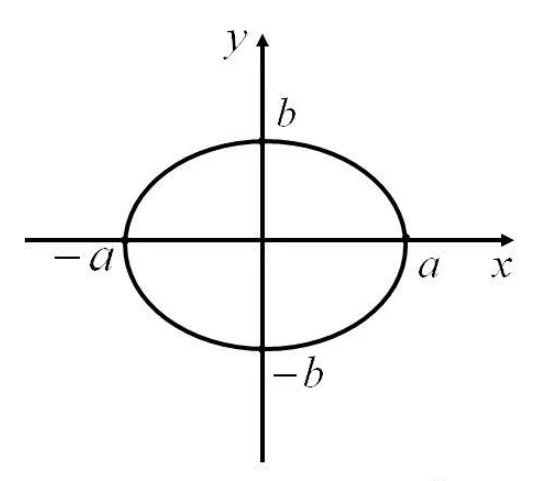


Рис. 4. Эллипс

# Описание классов Graphics, Pen и Brush

## Класс Graphics

Класс Graphics – это класс для работы с графикой, который входит в пространство имен System.Drawing. Класс Graphics включает в себя набор методов для рисования на экране. Они позволяют выводить текст, изображения, а также различные геометрические фигуры [[8]](#_Источники). Основные методы класса Graphics представлены в табл. 1.

Таблица 1

Методы класса Graphics

|  |  |
| --- | --- |
| Метод | Назначение |
| DrawLine | Рисует линию, соединяющую 2 точки |
| DrawRectangle | Рисует прямоугольник, заданный координатами верхнего левого угла, шириной и высотой |
| DrawEllipse | Рисует эллипс, заданный координатами ограничивающего прямоугольника |
| DrawPolygon | Рисует прямоугольник, заданный массивом точек |
| DrawString | Рисует заданный текст с использованием указанного шрифта, кисти и координат |
| FillRectangle | Заполняет прямоугольник, заданный координатами верхнего левого угла, шириной и высотой, используя заданную кисть |
| FillEllipse | Заполняет эллипс, заданный координатами ограничивающего прямоугольника, используя заданную кисть |
| FillPolygon | Заполняет прямоугольник, заданный массивом точек, используя заданную кисть |

Для применения методов класса Graphics необходимо использовать примитивы. Основными примитивами являются Pen (перо) и Brush (кисть). Для реализации разрабатываемой программы их будет достаточно.

## Класс Pen

Класс Pen – это примитив класса Graphics, предназначенный для рисования линий. С его помощью можно выбрать цвет пера (с помощью структуры Color), а также его толщину и стиль [[2]](#_Источники). Основные методы класса Graphics представлены в табл. 2.

Таблица 2

Методы класса Pen

|  |  |
| --- | --- |
| Метод | Назначение |
| Color | Устанавливает цвет пера |
| Width | Устанавливает толщину пера |
| DashStyle | Устанавливает стиль пунктира пера |
| StartCap, EndCap | Устанавливает стиль концов линий |
| LineJoin | Устанавливает стиль соединения линий |

## Класс Brush

Класс Brush – это примитив класса Graphics, предназначенный для заливки определенной области цветом или изображением. У класса Brush есть несколько подклассов. Подкласс SolidBrish позволяет залить определенную область заданным цветом, TextureBrush – изображением, а HatchBrush – узором [[2]](#_Источники). Основные методы класса Graphics представлены в табл. 3.

Таблица 3

Методы класса Brush

|  |  |
| --- | --- |
| Метод | Назначение |
| Color | Устанавливает цвет кисти |
| Clone | Создает и возвращает точную копию объекта Brush |
| GetType | Получает тип объекта Brush |
| Dispose | Освобождает все ресурсы, занятые объектом Brush |
| Equals | Определяет, равен ли указанный объект текущему объекту Brush |

# Описание созданного приложения

## Постановка задачи

В рамках выполнения курсовой работы необходимо написать программу, которая выводит на экран вращающуюся правильную шестиугольную пирамиду, а также позволяет выбирать ее цвет, масштаб, скорость и количество оборотов.

## Логическое проектирование – алгоритм работы

Разрабатываемая программа выполняет следующий алгоритм:

1. После запуска программы на экране появляется pictureBox1 (экран для рисования), comboBox1 (выпадающий список с цветами), trackBar1 (ползунок с выбором масштаба), trackBar2 (ползунок с выбором скорости), numericUpDown (выбор количества оборотов) и button1 (кнопка «Построить пирамиду»);
2. Пользователю необходимо ввести характеристики пирамиды: цвет (comboBox1), масштаб (trackBar1), скорость вращения (trackBar2) и количество оборотов (numericUpDown1);
3. Пользователю необходимо нажать на кнопку «Построить пирамиду» (button1);
4. Происходит описание и инициализация некоторых переменных и методов (Pen, w (длина pictureBox1), h (высота pictureBox1), stop = 0, coef, coef2);
5. Программа строит пирамиду по заданным пользователем характеристикам:
   1. Открывается цикл while (пока переменная stop не станет равна введенному в numericUpDown 1 числу);
   2. Переменным coef и coef2 присваиваются значения, выбранные в trackBar1 и trackBar2 соответственно;
   3. Задаются исходные значения координат x точек основания пирамиды (переменные ax, bx, cx, dx, ex, fx);
   4. Открывается цикл for со счетчиком i (от 0 до 40 с шагом 1), с каждым шагом которого переменные ax, cx увеличиваются на 1, dx, fx уменьшаются на 1, bx увеличивается на 3, ex уменьшается на 3;
   5. Исходные значения координат y точек основания пирамиды (переменные ay, by, cy, dy, ey, fy) высчитываются по формуле 1.1, из которой выражен y;
   6. Значения x, y точек пирамиды умножаются на коэффициент coef, введенный в trackBar1, и записываются в массивы A[2], B[2], C[2], D[2], E[2], F[2], K[2];
   7. Открывается оператор switch: значение case равно значению, выбранному в comboBox1;
   8. На экран с помощью пера одного из восьми цветов (в зависимости от значения case) выводятся линии от каждой из точек основания к вершине пирамиды, а также линии от точек основания к их соседним точкам;
   9. Программа делает паузу на 10/coef2 миллисекунд (coef2 – коэффициент из trackBar2);
   10. PictureBox1 очищается, цикл for повторяется с другими координатами точек;
   11. Цикл for закрывается, пирамида сделала оборот;
   12. Переменная stop увеличивается на 1. Начинается новый оборот. Цикл while повторяется до тех пор, пока переменная stop не достигнет значения, введенного в numericUpDown1;
6. Пирамида выполнила заданное количество оборотов и пропала с экрана. Пользователь может изменить параметры и вывести изображение пирамиды еще раз.

Блок-схема данного алгоритма представлена на рис. 5.



Рис. 5. Блок-схема программы

Продолжение рис. 5



## Физическое проектирование – структуры данных и спецификация функций

Структуры данных, которые были использованы в программе, представлены в табл. 4.

Таблица 4

Структуры данных

|  |  |  |
| --- | --- | --- |
| Обозначение | Тип данных | Назначение |
| w | int | Длина pictureBox1 |
| h | int | Высота pictureBox1 |
| stop | int | Счетчик в цикле while |
| coef | double | Коэффициент масштаба, равен значению trackBar1 |
| coef2 | double | Коэффициент скорости, равен значению trackBar2 |
| ax | int | Значение координаты x точки A |
| bx | int | Значение координаты x точки B |
| cx | int | Значение координаты x точки C |
| dx | int | Значение координаты x точки D |
| ex | int | Значение координаты x точки E |
| fx | int | Значение координаты x точки F |
| ay | int | Значение координаты y точки A |
| by | int | Значение координаты y точки B |
| cy | int | Значение координаты y точки C |
| dy | int | Значение координаты y точки D |
| ey | int | Значение координаты y точки E |
| fy | int | Значение координаты y точки F |
| A[2] | int | Массив координат точки A |
| B[2] | int | Массив координат точки B |
| C[2] | int | Массив координат точки C |
| D[2] | int | Массив координат точки D |
| E[2] | int | Массив координат точки E |
| F[2] | int | Массив координат точки F |
| K[2] | int | Массив координат точки K |

Спецификация функций, использованных в программе, отображена в табл. 5.

Таблица 5

Спецификация функций

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Тип | Формальные параметры | Назначение |
| button1\_Click | System::Void | System::Object^ sender, System::EventArgs^ e | Отвечает за рисование пирамиды |
| comboBox1\_SelectedIndexChanged | System::Void | System::Object^ sender, System::EventArgs^ e | Отвечает за действия при выборе элемента в comboBox1 |
| trackBar1\_Scroll | System::Void | System::Object^ sender, System::EventArgs^ e | Отвечает за действия при изменении положения ползунка в trackBar1 |
| trackBar2\_Scroll | System::Void | System::Object^ sender, System::EventArgs^ e | Отвечает за действия при изменении положения ползунка в trackBar2 |
| numericUpDown1\_ValueChanged | System::Void | System::Object^ sender, System::EventArgs^ e | Отвечает за действия при изменении числа в numericUpDown1 |
| pen0 | Pen^ | Color::Red | Задает красный цвет линии |
| pen1 | Pen^ | Color::Orange | Задает оранжевый цвет линии |

Продолжение табл. 5

|  |  |  |  |
| --- | --- | --- | --- |
| pen2 | Pen^ | Color::Yellow | Задает желтый цвет линии |
| pen3 | Pen^ | Color::Green | Задает зеленый цвет линии |
| pen4 | Pen^ | Color::Blue | Задает синий цвет линии |
| pen5 | Pen^ | Color::Purple | Задает фиолетовый цвет линии |
| pen6 | Pen^ | Color::Pink | Задает розовый цвет линии |
| pen7 | Pen^ | Color::Black | Задает черный цвет линии |

## Описание интерфейса пользователя

После запуска программы пользователь видит перед собой белый экран (поле, где будет выводиться пирамида), а также несколько элементов внизу окна программы (рис. 6):

* выпадающий список с цветами пирамиды;
* ползунок, регулирующий масштаб пирамиды;
* ползунок, регулирующий скорость вращения пирамиды;
* поле для ввода числа – количества оборотов пирамиды;
* кнопка «Вывести пирамиду».

После нажатия кнопки «Вывести пирамиду», на экране появится вращающаяся пирамида с заданными пользователем характеристиками (рис. 7). После выполнения заданного числа оборотов пирамида пропадает. Пользователь может изменить характеристики и вывести пирамиду на экран вновь.

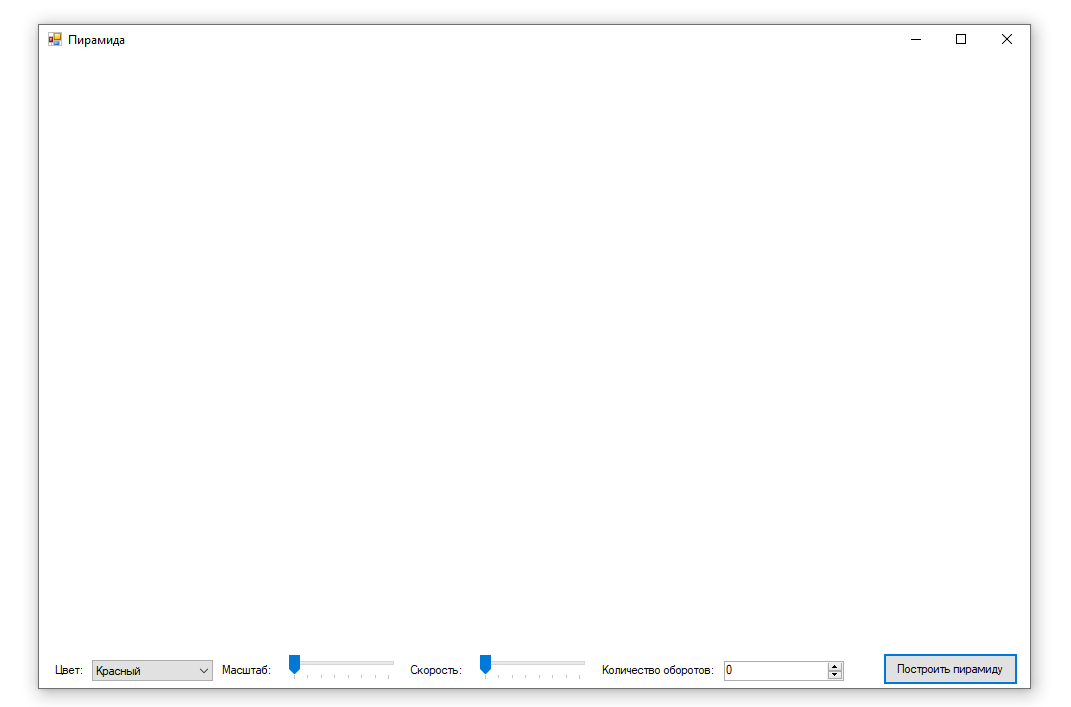


Рис. 6. Программа в начале работы

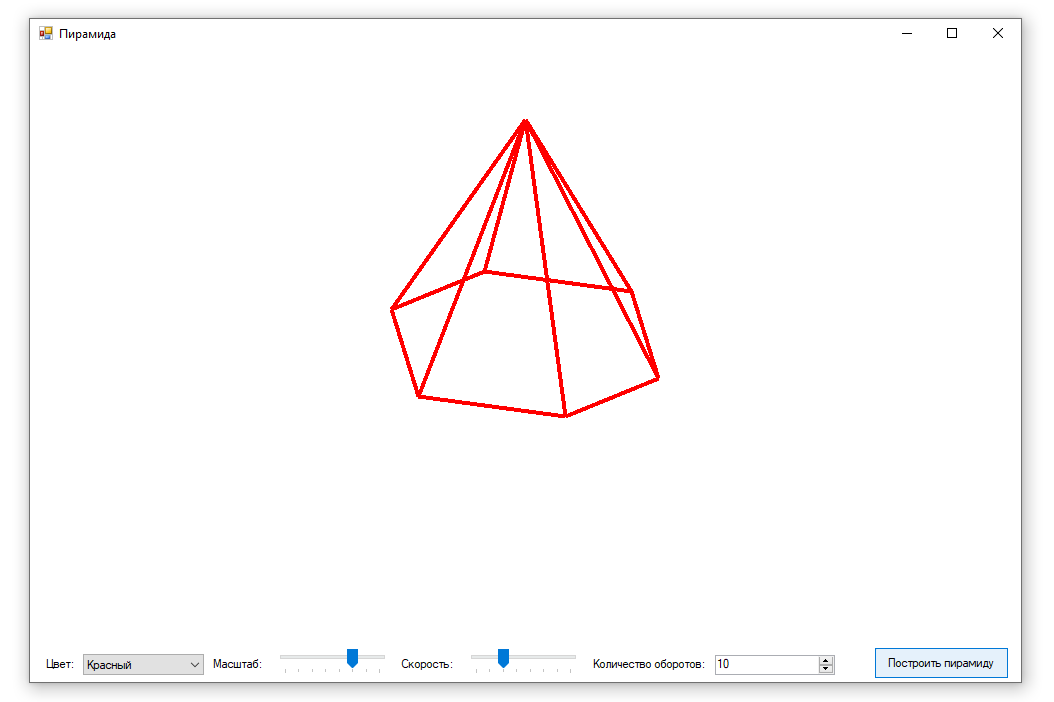


Рис. 7. Вывод пирамиды на экран

## Тестирование

Результаты тестирования программы представлены в табл. 6.

Таблица 6

Результаты тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата тестирования | Тестируемый модуль | Тестирование проводил | Описание теста | Результат тестирования |
| 22.04.2023 | Project1.sln | Маркелов С. А. | Вывод на экран пирамиды без вращения. Линии выведены неверно | Ошибка. Исправлено |
| 23.04.2023 | Project1.sln | Маркелов С. А. | Вывод на экран пирамиды без вращения | Успех |
| 29.04.2023 | Project1.sln | Маркелов С. А. | Вращение изображения пирамиды. При вращении пирамида искажается | Ошибка. Исправлено |
| 30.04.2023 | Project1.sln | Маркелов С. А. | Вращение изображения пирамиды | Успех |
| 01.05.2023 | Project1.sln | Маркелов С. А. | Смена цвета пирамиды | Успех |
| 02.05.2023 | Project1.sln | Маркелов С. А. | Изменение масштаба пирамиды | Успех |
| 06.05.2023 | Project1.sln | Маркелов С. А. | Изменение скорости вращения пирамиды | Успех |
| 07.05.2023 | Project1.sln | Маркелов С. А. | Добавление comboBox, trackBar и numericUpDown для выбора характеристик пирамиды. Характеристики пирамиды не изменялись | Ошибка. Исправлено |
| 08.05.2023 | Project1.sln | Маркелов С. А. | Выбор характеристик пирамиды | Успех |

## Результат работы

В рамках выполнения курсовой работы была написана программа, которая выводит на экран трехмерное изображение вращающейся правильной шестиугольной пирамидой с заданными пользователем параметрами: цветом, масштабом, скоростью вращения и количеством оборотов (см. рис 5-6). Программа была написана на языке программирования высокого уровня C++. Также были использованы Windows Forms.

# Заключение

В ходе выполнения курсовой работы была разработана программа, которая выводит на экран трехмерное изображение вращающейся правильной шестиугольной пирамиды. Программа имеет дополнительные функции, такие как изменение цвета, масштаба, скорости вращения и количества оборотов пирамиды. В разработке были применены Windows Forms, а также классы Graphics, Pen и Brush. Программа предназначена для школьников и студентов. Программа может помочь при решении задач по геометрии или при необходимости построения чертежа шестиугольной пирамиды.

# Список литературы

1. Методика и организация самостоятельной работы студентов: учебно-методическое пособие / Е.В. Ершов, Л.Н. Виноградова, В.В. Селивановских [и др.]. – Череповец: ЧГУ, 2015.
2. Пышницкий, К.М. Структурное программирование: курс лекций / К.М. Пышницкий. – Череповец: ЧГУ, 2023.
3. Язык программирования — Википедия [электр.ресурс] <https://ru.wikipedia.org/wiki/Язык_программирования>. Дата обращения: 31.05.2023.
4. C++ — Википедия [электр.ресурс] [https://ru.wikipedia.org/wiki/C++.](https://ru.wikipedia.org/wiki/C++) Дата обращения: 29.05.2023.
5. Руководство. Создание приложения Windows Forms с помощью .NET – Microsoft Learn [электр.ресурс] <https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/get-started/create-app-visual-studio>. Дата обращения: 30.05.2023
6. Пирамида — Википедия [электр.ресурс] <https://ru.wikipedia.org/wiki/Пирамида_(геометрия)>. Дата обращения: 29.05.2023.
7. Эллипс — Википедия [электр.ресурс] <https://ru.wikipedia.org/wiki/Эллипс>. Дата обращения: 29.05.2023.
8. Работа с графикой (C++/CLI) – Microsoft Learn [электр.ресурс] https://learn.microsoft.com/ru-ru/cpp/dotnet/graphics-operations-cpp-cli. Дата обращения: 30.05.2023.

# Приложение 1. Техническое задание

МИНОБРАНАУКИ РОССИИ

федеральное государственное бюджетное

образовательное учреждение высшего образования

ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт информационных технологий

наименование института (факультета)

Математическое и программное обеспечение ЭВМ

наименование кафедры

Структурное программирование

наименование дисциплины в соответствии с учебным планом

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ

д. т.н. \_\_\_\_\_\_\_\_\_ Ершов Е.В.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2023 г.

Программирование на языке высокого уровня

Техническое задание на курсовую работу

Листов 7

Руководитель: Журавлева Ю. М.

Исполнитель: студент гр. 1ПИб-02-3оп-22

Маркелов С. А.

2023 г.

Введение

Данная курсовая работа посвящена разработке программы, которая будет создавать трехмерное графическое изображение вращающейся шестиугольной пирамиды. Программа будет позволять менять его масштаб (размеры), цвет и скорость 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: 22 февраля 2023 года.

Наименование темы разработки: Трехмерное вращающееся изображение шестиугольной пирамиды.

2. Назначение разработки

Данная программа поможет пользователям построить трехмерный чертеж шестиугольной пирамиды. Она поможет решить следующие задачи:

* построение чертежа какой-либо детали, которая имеет форму шестиугольной пирамиды;
* построение чертежа пирамиды для решения задач по геометрии.

Модель пирамиды будет вращаться, что позволит пользователю рассмотреть ее с разных ракурсов. Программа позволит пользователю выбрать нужный масштаб, цвет, скорость вращения и количество оборотов пирамиды.

3. Требования к программе

* 1. Требования к функциональным характеристикам

Разрабатываемая программа должна:

* выводить на экран трехмерное изображение шестиугольной пирамиды;
* реализовывать вращение этого изображения;
* иметь возможность выбора цвета чертежа;
* иметь возможность выбора размера чертежа;
* иметь возможность выбора скорости вращения пирамиды;
* иметь возможность выбора количества оборотов пирамиды.
  1. Требования к надежности

Программа должна работать без ошибок. Для этого после создания она должна быть протестирована. При возникновении каких-либо ошибок они должны быть исправлены.

* 1. Условия эксплуатации

Для корректной работы программы необходимо:

* наличие любого устройства, на котором возможно запускать файлы формата .exe (компьютер, ноутбук);
* наличие на устройстве современной версии любой операционной системы.
  1. Требования к составу и параметрам технических средств

Для корректной работы программы необходимо:

* оперативная память не менее 2 Гб;
* процессор не менее чем с 2 ядрами и тактовой частотой не ниже 1,6 ГГц;
* разрешение экрана не менее 800 х 600;
* видеокарта с видеопамятью не менее 1 Гб;
* наличие мыши и клавиатуры.

В целом программа не очень требовательна к составу и параметрам технических средств, и сможет работать даже на старых и слабых компьютерах.

* 1. Требования к информационной и программной совместимости

Для корректной работы программы необходимо:

* современная операционная система (Windows 7, 8.1, 10, 11, macOS, Linux);
* установленный на устройстве пакет библиотек «Microsoft Visual C++».
  1. Требования к маркировке и упаковке

Требования к маркировке и упаковке программы не предъявляются.

* 1. Требования к транспортированию и хранению

Программа может храниться и транспортироваться в виде файлов, необходимых для ее работы, на носителе информации (флешка, CD-диск). Такую программу можно распространять, передавая его с одного компьютера на другой с помощью флешки или диска, либо отправляя ее по электронной почте.

* 1. Специальные требования

Специальные требования к программе не предъявляются.

4. Требование к программной документации

* 1. Содержание расчетно-пояснительной записки

Программная документация должна содержать расчётно-пояснительную записку с содержанием:

Титульный лист;

Оглавление;

1. Введение;

2. Описание предметной области;

3. Описание классов Graphics, Pen и Brush;

4. Описание созданного приложения;

5. Заключение;

6. Список литературы;

7. Техническое задание;

8. Руководство пользователя;

9. Программный код.

* 1. Требования к оформлению

В данном пункте представлены требования к оформлению документации в соответствии с ГОСТ (табл. П1.1).

Таблица П1.1

|  |  |
| --- | --- |
| 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

В данном пункте описаны стадии и этапы разработки программы (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 | 22.02.2023 | Утверждена тема для разработки |  |
| Изучение информации о необходимых инструментах для создания программы | 29.03.2023 | Получены теоретические знания для разработки программы |  |
| Проектирование программы | 12.04.2023 | Подготовлен проект программы |  |
| Написание программы | 10.05.2023 | Готова программа |  |
| Тестирование и отладка программы | 24.05.2023 | Программа проверена на наличие ошибок и работоспособность |  |

6. Порядок контроля и приемки

В данном пункте описан порядок контроля и приемки курсовой работы (табл. 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 |
| Определение темы для курсовой работы | 22.02.2023 | Утверждена тема для разработки |  |
| Разработка технического задания | 15.03.2023 | Готовое ТЗ |  |

Продолжение табл. П1.3

|  |  |  |  |
| --- | --- | --- | --- |
| Разработка программы по ТЗ | 10.05.2023 | Готовая программа |  |
| Написание расчетно-пояснительной записки | 24.05.2023 | Готовое РПЗ |  |
| Сдача курсовой работы | 07.06.2023 | Получение оценки за курсовую работу |  |

# Приложение 2. Руководство пользователя

Руководство пользователя

1. Запустить файл Project1.exe, который находится в расположении …\Project1\x64\Debug;
2. После открытия на экране появляется окно с белым фоном и несколькими кнопками внизу (рис. П2.1);

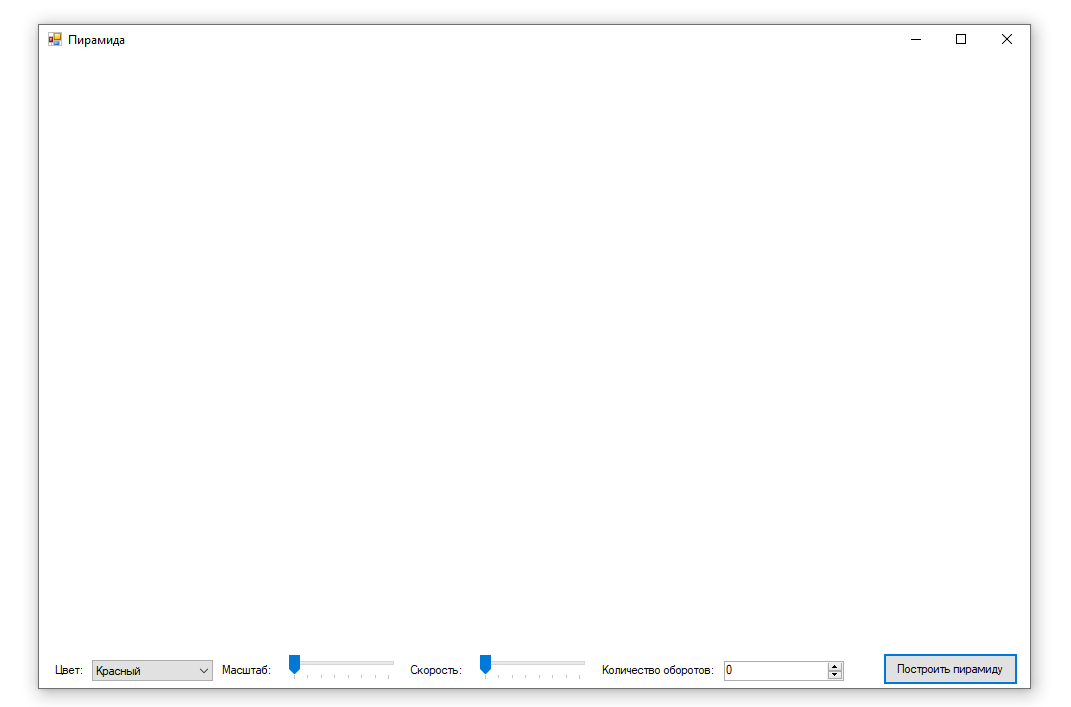


Рис. П2.1. Окно программы после запуска

1. Необходимо выбрать характеристики пирамиды:
   1. Из выпадающего списка необходимо выбрать цвет пирамиды (рис. П2.2);

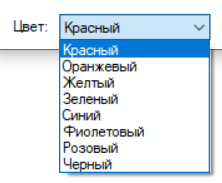


Рис. П2.2. Выпадающий список с выбором цвета

* 1. На первом ползунке необходимо выбрать масштаб пирамиды: слева – наименьший, справа – наибольший (рис. П2.3);



Рис. П2.3. Ползунок с выбором масштаба

* 1. На втором ползунке необходимо выбрать скорость вращения пирамиды: слева – наименьшая, справа – наибольшая (рис. П2.4);



Рис. П2.4. Ползунок с выбором скорости вращения

* 1. В числовое поле необходимо ввести с клавиатуры либо с помощью стрелок число оборотов пирамиды – от 0 до 100 (рис. П2.5);



Рис. П2.5. Числовое поле для ввода количества оборотов

1. После выбора необходимых характеристик нужно нажать на кнопку «Построить пирамиду» (рис. П2.6);



Рис. П2.6. Кнопка «Построить пирамиду»

1. Программа выводит на экран вращающуюся пирамиду с заданными характеристиками (рис. П2.7-П2.9). Во время вывода изображения на экран изменять характеристики пирамиды нельзя – ползунки и поля выбора становятся неактивными;

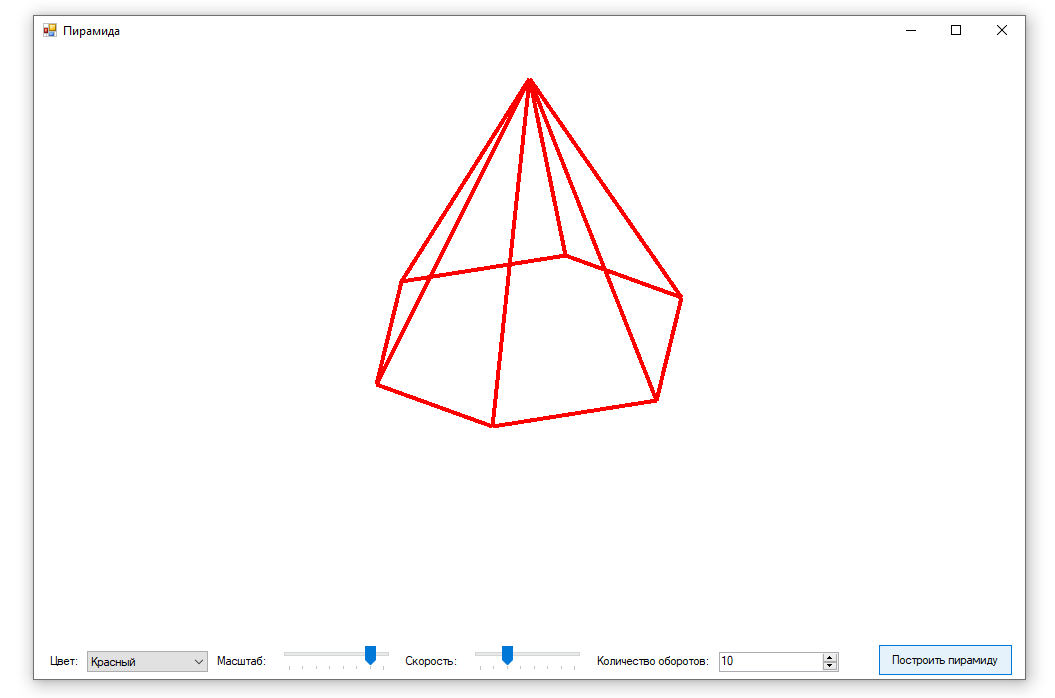


Рис. П2.7. Пирамида красного цвета, масштаб 1,75х, скорость 0,75х, 10 оборотов

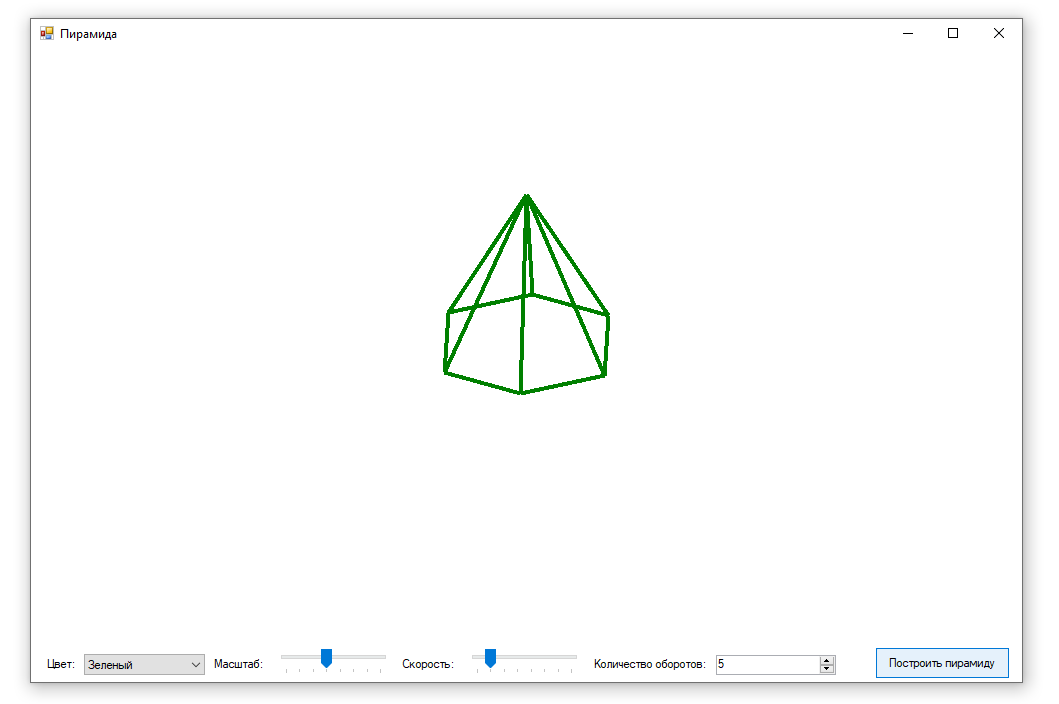


Рис. П2.8. Пирамида зеленого цвета, масштаб 1х, скорость 0,5х, 5 оборотов

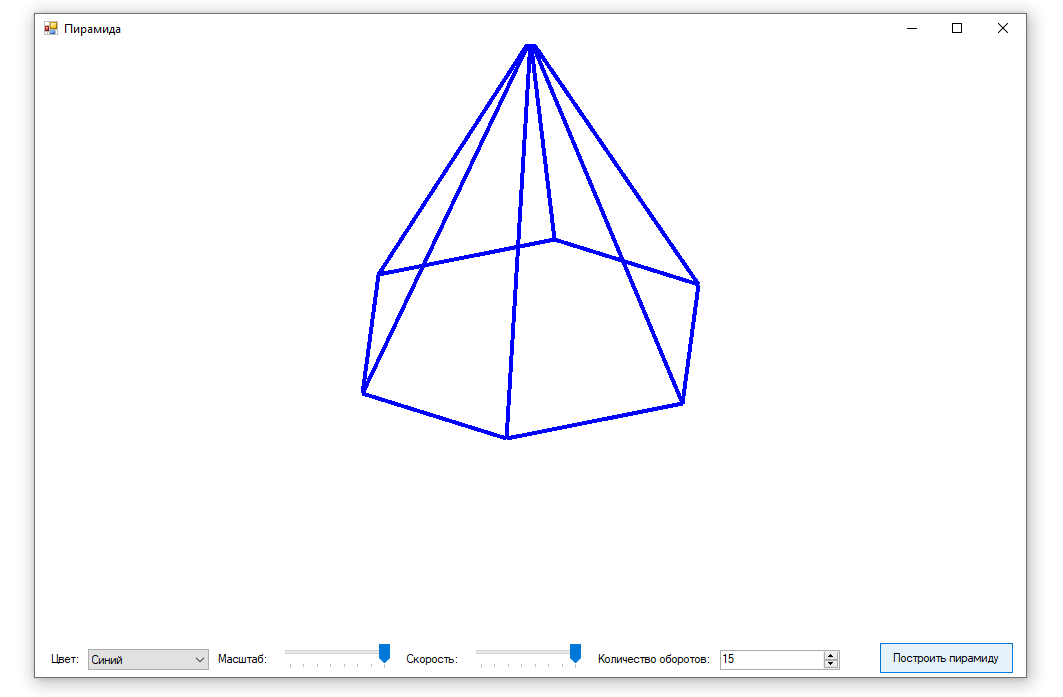


Рис. П2.9. Пирамида синего цвета, масштаб 2х, скорость 2х, 15 оборотов

1. После окончания вывода пирамиды на экран изменение характеристик пирамиды вновь становится доступно. Пирамиду можно вывести повторно – с теми же самыми либо измененными характеристиками.

# Приложение 3. Программный код

#pragma once

#include "windows.h"

#include <iostream>

#include <math.h>

namespace Project1 {

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 std;

/// <summary>

/// Сводка для MyForm

/// </summary>

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

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::PictureBox^ pictureBox1;

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

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

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

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

private: System::Windows::Forms::NumericUpDown^ numericUpDown1;

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

private: System::Windows::Forms::TrackBar^ trackBar1;

private: System::Windows::Forms::TrackBar^ trackBar2;

protected:

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#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->button1 = (gcnew System::Windows::Forms::Button());

this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());

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

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

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

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

this->numericUpDown1 = (gcnew System::Windows::Forms::NumericUpDown());

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

this->trackBar1 = (gcnew System::Windows::Forms::TrackBar());

this->trackBar2 = (gcnew System::Windows::Forms::TrackBar());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->numericUpDown1))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar1))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar2))->BeginInit();

this->SuspendLayout();

//

// button1

//

this->button1->Location = System::Drawing::Point(844, 598);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(135, 32);

this->button1->TabIndex = 0;

this->button1->Text = L"Построить пирамиду";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

//

// pictureBox1

//

this->pictureBox1->BackColor = System::Drawing::SystemColors::Window;

this->pictureBox1->Location = System::Drawing::Point(0, -1);

this->pictureBox1->Name = L"pictureBox1";

this->pictureBox1->Size = System::Drawing::Size(991, 593);

this->pictureBox1->TabIndex = 1;

this->pictureBox1->TabStop = false;

//

// comboBox1

//

this->comboBox1->DropDownStyle = System::Windows::Forms::ComboBoxStyle::DropDownList;

this->comboBox1->FormattingEnabled = true;

this->comboBox1->Items->AddRange(gcnew cli::array< System::Object^ >(8) {

L"Красный", L"Оранжевый", L"Желтый", L"Зеленый",

L"Синий", L"Фиолетовый", L"Розовый", L"Черный"

});

this->comboBox1->Location = System::Drawing::Point(53, 605);

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

this->comboBox1->Size = System::Drawing::Size(121, 21);

this->comboBox1->TabIndex = 2;

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

this->comboBox1->SelectedIndex = 0;

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(13, 608);

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

this->label1->Size = System::Drawing::Size(35, 13);

this->label1->TabIndex = 4;

this->label1->Text = L"Цвет:";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(180, 608);

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

this->label2->Size = System::Drawing::Size(56, 13);

this->label2->TabIndex = 5;

this->label2->Text = L"Масштаб:";

//

// label3

//

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(560, 608);

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

this->label3->Size = System::Drawing::Size(119, 13);

this->label3->TabIndex = 6;

this->label3->Text = L"Количество оборотов:";

//

// numericUpDown1

//

this->numericUpDown1->Location = System::Drawing::Point(685, 606);

this->numericUpDown1->Name = L"numericUpDown1";

this->numericUpDown1->Size = System::Drawing::Size(120, 20);

this->numericUpDown1->TabIndex = 8;

this->numericUpDown1->ValueChanged += gcnew System::EventHandler(this, &MyForm::numericUpDown1\_ValueChanged);

//

// label4

//

this->label4->AutoSize = true;

this->label4->Location = System::Drawing::Point(369, 608);

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

this->label4->Size = System::Drawing::Size(58, 13);

this->label4->TabIndex = 10;

this->label4->Text = L"Скорость:";

//

// trackBar1

//

this->trackBar1->Location = System::Drawing::Point(242, 598);

this->trackBar1->Maximum = 7;

this->trackBar1->Name = L"trackBar1";

this->trackBar1->Size = System::Drawing::Size(121, 45);

this->trackBar1->TabIndex = 11;

this->trackBar1->Scroll += gcnew System::EventHandler(this, &MyForm::trackBar1\_Scroll);

//

// trackBar2

//

this->trackBar2->Location = System::Drawing::Point(433, 598);

this->trackBar2->Maximum = 7;

this->trackBar2->Name = L"trackBar2";

this->trackBar2->Size = System::Drawing::Size(121, 45);

this->trackBar2->TabIndex = 12;

//

// MyForm

//

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

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

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

this->ClientSize = System::Drawing::Size(991, 633);

this->Controls->Add(this->trackBar2);

this->Controls->Add(this->trackBar1);

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

this->Controls->Add(this->numericUpDown1);

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

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

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

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

this->Controls->Add(this->pictureBox1);

this->Controls->Add(this->button1);

this->Name = L"MyForm";

this->Text = L"Пирамида";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->numericUpDown1))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar1))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar2))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

MaximizeBox = false;

}

#pragma endregion

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

Graphics^ gr = pictureBox1->CreateGraphics();

Pen^ pen0 = gcnew Pen(Color::Red, 4);

Pen^ pen1 = gcnew Pen(Color::Orange, 4);

Pen^ pen2 = gcnew Pen(Color::Yellow, 4);

Pen^ pen3 = gcnew Pen(Color::Green, 4);

Pen^ pen4 = gcnew Pen(Color::Blue, 4);

Pen^ pen5 = gcnew Pen(Color::Purple, 4);

Pen^ pen6 = gcnew Pen(Color::Pink, 4);

Pen^ pen7 = gcnew Pen(Color::Black, 4);

SolidBrush^ brush1 = gcnew SolidBrush(Color::Red);

int w = pictureBox1->Width;

int h = pictureBox1->Height;

int stop = 0;

double coef, coef2;

while (stop < numericUpDown1->Value) {

if (trackBar1->Value == 0) coef = 0.25;

if (trackBar1->Value == 1) coef = 0.5;

if (trackBar1->Value == 2) coef = 0.75;

if (trackBar1->Value == 3) coef = 1;

if (trackBar1->Value == 4) coef = 1.25;

if (trackBar1->Value == 5) coef = 1.5;

if (trackBar1->Value == 6) coef = 1.75;

if (trackBar1->Value == 7) coef = 2;

if (trackBar2->Value == 0) coef2 = 0.25;

if (trackBar2->Value == 1) coef2 = 0.5;

if (trackBar2->Value == 2) coef2 = 0.75;

if (trackBar2->Value == 3) coef2 = 1;

if (trackBar2->Value == 4) coef2 = 1.25;

if (trackBar2->Value == 5) coef2 = 1.5;

if (trackBar2->Value == 6) coef2 = 1.75;

if (trackBar2->Value == 7) coef2 = 2;

int ax = -100, bx = -60, cx = 60, dx = 100, ex = 60, fx = -60;

for (int i = 0; i <= 40; ax++, bx += 3, cx++, dx--, ex -= 3, fx--, i++) {

double ay = sqrt((10000 - pow(ax, 2)) / 4);

double by = sqrt((10000 - pow(bx, 2)) / 4);

double cy = sqrt((10000 - pow(cx, 2)) / 4);

double dy = sqrt((10000 - pow(dx, 2)) / 4);

double ey = sqrt((10000 - pow(ex, 2)) / 4);

double fy = sqrt((10000 - pow(fx, 2)) / 4);

int A[] = { w / 2 + ax \* coef, h / 2 + ay \* coef };

int B[] = { w / 2 + bx \* coef, h / 2 + by \* coef };

int C[] = { w / 2 + cx \* coef, h / 2 + cy \* coef };

int D[] = { w / 2 + dx \* coef, h / 2 - dy \* coef };

int E[] = { w / 2 + ex \* coef, h / 2 - ey \* coef };

int F[] = { w / 2 + fx \* coef, h / 2 - fy \* coef };

int K[] = { w / 2, h / 2 - 150 \* coef };

switch (comboBox1->SelectedIndex) {

case 0:

gr->DrawLine(pen0, A[0], A[1], K[0], K[1]);

gr->DrawLine(pen0, B[0], B[1], K[0], K[1]);

gr->DrawLine(pen0, C[0], C[1], K[0], K[1]);

gr->DrawLine(pen0, D[0], D[1], K[0], K[1]);

gr->DrawLine(pen0, E[0], E[1], K[0], K[1]);

gr->DrawLine(pen0, F[0], F[1], K[0], K[1]);

gr->DrawLine(pen0, A[0], A[1], B[0], B[1]);

gr->DrawLine(pen0, B[0], B[1], C[0], C[1]);

gr->DrawLine(pen0, C[0], C[1], D[0], D[1]);

gr->DrawLine(pen0, D[0], D[1], E[0], E[1]);

gr->DrawLine(pen0, E[0], E[1], F[0], F[1]);

gr->DrawLine(pen0, F[0], F[1], A[0], A[1]);

break;

case 1:

gr->DrawLine(pen1, A[0], A[1], K[0], K[1]);

gr->DrawLine(pen1, B[0], B[1], K[0], K[1]);

gr->DrawLine(pen1, C[0], C[1], K[0], K[1]);

gr->DrawLine(pen1, D[0], D[1], K[0], K[1]);

gr->DrawLine(pen1, E[0], E[1], K[0], K[1]);

gr->DrawLine(pen1, F[0], F[1], K[0], K[1]);

gr->DrawLine(pen1, A[0], A[1], B[0], B[1]);

gr->DrawLine(pen1, B[0], B[1], C[0], C[1]);

gr->DrawLine(pen1, C[0], C[1], D[0], D[1]);

gr->DrawLine(pen1, D[0], D[1], E[0], E[1]);

gr->DrawLine(pen1, E[0], E[1], F[0], F[1]);

gr->DrawLine(pen1, F[0], F[1], A[0], A[1]);

break;

case 2:

gr->DrawLine(pen2, A[0], A[1], K[0], K[1]);

gr->DrawLine(pen2, B[0], B[1], K[0], K[1]);

gr->DrawLine(pen2, C[0], C[1], K[0], K[1]);

gr->DrawLine(pen2, D[0], D[1], K[0], K[1]);

gr->DrawLine(pen2, E[0], E[1], K[0], K[1]);

gr->DrawLine(pen2, F[0], F[1], K[0], K[1]);

gr->DrawLine(pen2, A[0], A[1], B[0], B[1]);

gr->DrawLine(pen2, B[0], B[1], C[0], C[1]);

gr->DrawLine(pen2, C[0], C[1], D[0], D[1]);

gr->DrawLine(pen2, D[0], D[1], E[0], E[1]);

gr->DrawLine(pen2, E[0], E[1], F[0], F[1]);

gr->DrawLine(pen2, F[0], F[1], A[0], A[1]);

break;

case 3:

gr->DrawLine(pen3, A[0], A[1], K[0], K[1]);

gr->DrawLine(pen3, B[0], B[1], K[0], K[1]);

gr->DrawLine(pen3, C[0], C[1], K[0], K[1]);

gr->DrawLine(pen3, D[0], D[1], K[0], K[1]);

gr->DrawLine(pen3, E[0], E[1], K[0], K[1]);

gr->DrawLine(pen3, F[0], F[1], K[0], K[1]);

gr->DrawLine(pen3, A[0], A[1], B[0], B[1]);

gr->DrawLine(pen3, B[0], B[1], C[0], C[1]);

gr->DrawLine(pen3, C[0], C[1], D[0], D[1]);

gr->DrawLine(pen3, D[0], D[1], E[0], E[1]);

gr->DrawLine(pen3, E[0], E[1], F[0], F[1]);

gr->DrawLine(pen3, F[0], F[1], A[0], A[1]);

break;

case 4:

gr->DrawLine(pen4, A[0], A[1], K[0], K[1]);

gr->DrawLine(pen4, B[0], B[1], K[0], K[1]);

gr->DrawLine(pen4, C[0], C[1], K[0], K[1]);

gr->DrawLine(pen4, D[0], D[1], K[0], K[1]);

gr->DrawLine(pen4, E[0], E[1], K[0], K[1]);

gr->DrawLine(pen4, F[0], F[1], K[0], K[1]);

gr->DrawLine(pen4, A[0], A[1], B[0], B[1]);

gr->DrawLine(pen4, B[0], B[1], C[0], C[1]);

gr->DrawLine(pen4, C[0], C[1], D[0], D[1]);

gr->DrawLine(pen4, D[0], D[1], E[0], E[1]);

gr->DrawLine(pen4, E[0], E[1], F[0], F[1]);

gr->DrawLine(pen4, F[0], F[1], A[0], A[1]);

break;

case 5:

gr->DrawLine(pen5, A[0], A[1], K[0], K[1]);

gr->DrawLine(pen5, B[0], B[1], K[0], K[1]);

gr->DrawLine(pen5, C[0], C[1], K[0], K[1]);

gr->DrawLine(pen5, D[0], D[1], K[0], K[1]);

gr->DrawLine(pen5, E[0], E[1], K[0], K[1]);

gr->DrawLine(pen5, F[0], F[1], K[0], K[1]);

gr->DrawLine(pen5, A[0], A[1], B[0], B[1]);

gr->DrawLine(pen5, B[0], B[1], C[0], C[1]);

gr->DrawLine(pen5, C[0], C[1], D[0], D[1]);

gr->DrawLine(pen5, D[0], D[1], E[0], E[1]);

gr->DrawLine(pen5, E[0], E[1], F[0], F[1]);

gr->DrawLine(pen5, F[0], F[1], A[0], A[1]);

break;

case 6:

gr->DrawLine(pen6, A[0], A[1], K[0], K[1]);

gr->DrawLine(pen6, B[0], B[1], K[0], K[1]);

gr->DrawLine(pen6, C[0], C[1], K[0], K[1]);

gr->DrawLine(pen6, D[0], D[1], K[0], K[1]);

gr->DrawLine(pen6, E[0], E[1], K[0], K[1]);

gr->DrawLine(pen6, F[0], F[1], K[0], K[1]);

gr->DrawLine(pen6, A[0], A[1], B[0], B[1]);

gr->DrawLine(pen6, B[0], B[1], C[0], C[1]);

gr->DrawLine(pen6, C[0], C[1], D[0], D[1]);

gr->DrawLine(pen6, D[0], D[1], E[0], E[1]);

gr->DrawLine(pen6, E[0], E[1], F[0], F[1]);

gr->DrawLine(pen6, F[0], F[1], A[0], A[1]);

break;

case 7:

gr->DrawLine(pen7, A[0], A[1], K[0], K[1]);

gr->DrawLine(pen7, B[0], B[1], K[0], K[1]);

gr->DrawLine(pen7, C[0], C[1], K[0], K[1]);

gr->DrawLine(pen7, D[0], D[1], K[0], K[1]);

gr->DrawLine(pen7, E[0], E[1], K[0], K[1]);

gr->DrawLine(pen7, F[0], F[1], K[0], K[1]);

gr->DrawLine(pen7, A[0], A[1], B[0], B[1]);

gr->DrawLine(pen7, B[0], B[1], C[0], C[1]);

gr->DrawLine(pen7, C[0], C[1], D[0], D[1]);

gr->DrawLine(pen7, D[0], D[1], E[0], E[1]);

gr->DrawLine(pen7, E[0], E[1], F[0], F[1]);

gr->DrawLine(pen7, F[0], F[1], A[0], A[1]);

break;

}

Sleep(10/coef2);

pictureBox1->Refresh();

}

stop++;

}

}

private: System::Void comboBox1\_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void trackBar1\_Scroll(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void trackBar2\_Scroll(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void numericUpDown1\_ValueChanged(System::Object^ sender, System::EventArgs^ e) {

}

};

}