**СОДЕРЖАНИЕ**

[Введение](#_Toc474724148) 3

1 [Исследовательская часть 5](#_Toc474724149)

 [1.1 Особенности графического режима. 5](#_Toc474724150)

 [1.2 Основные понятия графического режима 6](#_Toc474724151)

[2 Практическая часть 8](#_Toc474724152)

 [2.1 Подключение графических возможностей. 8](#_Toc474724153)

 [2.2 Графические примитивы 10](#_Toc474724154)

 [2.2.1 Точка 10](#_Toc474724155)

 [2.2.2 Отрезок 10](#_Toc474724156)

 [2.2.3 Прямоугольник 11](#_Toc474724157)

 [2.2.4 Эллипс 12](#_Toc474724158)

 [2.2.5 Дуга 13](#_Toc474724159)

 [2.2.6 Сектор 14](#_Toc474724160)

 [2.2.7 Сегмент 15](#_Toc474724161)

 [2.2.8 Полилиния (ломанная) 16](#_Toc474724162)

 [2.3 Пример применения графических возможностей 17](#_Toc474724163)

[Заключение 17](#_Toc474724164)

[Список использованных источников](#_Toc474724165) 21

# ВВЕДЕНИЕ

Ученые утверждают, что человек более семидесяти процентов информации воспринимает именно визуально. А это указывает на то, что компьютерные системы должны быть обеспечены элементами визуализации сведений. Во времена разработки первых языков программирования общего назначения технические средства вычислительной техники не имели таких элементов и в самих языках отсутствовали инструкции для выполнения какой-либо удобной для человека визуализации.

Первые мониторы, были и остаются неотъемлемой частью компьютера, и предназначается для ввода или вывода текстовой информации до конца 70-ых годов на экране мониторов можно было увидеть всего лишь два цвета- черный и белый. Но позже, на основе телевизоров разработали первые дисплеи для компьютеров с композитным видеовыходом. Одновременно с этим в языках программирования стали появляются инструкции, направленные на разрешение проблем визуализации результатов расчетов.

Только в 1981-ом году IBM представила первый персональный компьютер, который состоял из трёх частей: клавиатуры, системного блока и монитора. Появление удобного для пользователя устройства отображения, привело к повышению спроса на ПК и, соответственно, спроса на удобные для человека программы. А создание таких программ возможно только при наличии в языках программирования соответствующих инструкций.

В след за консольными средствами отображения входных данных и результатов появился текстово-терминальный режим. Этот режим предполагал возможность управлять отображением символьные данные в любом, удобном для разработчика, месте экрана.

Следующим шагом в развитии технического и программного обеспечения ПК стало появление мониторов с реализацией графического стандарта CGA (Color Graphics Adapter) для цветной графики. И пусть в первых версиях эти мониторы передавали четыре цвета и давали разрешение всего 320 x 200 пикселей. Но для них в ЯП уже стали присутствовать актуальные компьютерной графики инструкции.

Язык С стал одним из первых языком программирования, которому обучали людей для разработки будущих программ. В связи с чем в будущем появилось огромное количество программ именно на этом языке. Программист пытается создавать программы, которые бы максимально упрощали жизнь человека, именно по этой причине языки программирования совершенствуются, в них появляется что-то новое и перерабатывается что-то старое.

Цель курсовой работы – рассмотреть множество команд-инструкций языка программирования С++ для реализации компьютерной графики.

# 1 Исследовательская часть

# Любой язык программирования предназначен для написания компьютерных программ, которые представляют собой набор правил, позволяющих компьютеру выполнить тот или иной [вычислительный процесс](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81), организовать управление различными объектами, в том числе и монитором.

Язык программирования общего назначения C, разработанный в 1969—1973 годах сотрудником [Bell Labs](https://ru.wikipedia.org/wiki/Bell_Labs) [Деннисом Ритчи](https://ru.wikipedia.org/wiki/%D0%A0%D0%B8%D1%82%D1%87%D0%B8%2C_%D0%94%D0%B5%D0%BD%D0%BD%D0%B8%D1%81), оказал существенное влияние на развитие индустрии программного обеспечения, а его синтаксис стал основой для многих языков программирования, в том числе и на [C++](https://ru.wikipedia.org/wiki/C%2B%2B).

Синтаксис C++ это дальнейшее развитие языка [C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_%28%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%29). Одним из принципов разработки было сохранение совместимости с C. C++ широко используется для разработки программного обеспечения, являясь одним из самых распространенных, наиболее часто используемых языков программирования. Поддерживает такие [парадигмы программирования](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D1%8B_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, [абстракцию данных](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), объявление типов (классов) объектов, виртуальные функции.

## Особенности графического режима

Важно понимать, что организация изображения в графическом режиме имеет важные отличия от формирования в текстовом режиме.

Экран в текстовом режиме можно понимать, как таблицу. у которой в отдельную ячейку можно поместить только один печатный символ. Размеры такой таблицы строго фиксированы: всего 24 строки по 40 позиций. Для символов то же всё строго – их количество и внешний вид зафиксирован в специальной странице постоянной памяти. Задача формирования текстового интерфейса – вывод символов в заданную позицию экрана (клетку таблицы). Данная задача не требует большого количества специальных инструкций и значительного объёма памяти для запоминания изображения, отображаемого на мониторе.

В графическом режиме требуется контролировать каждую точку изображения, а таких точек на первых графических экранах было 320х200, что значительно больше чем «клеток» -позиций в текстовом режиме. Кроме того, само изображение формируется не из набора фиксированных символов, а из графических примитивов, у которых позиция, размер и внешний вид определяются в конкретный момент отображения. Всё это требует наличия специальных программных и технических средств.

 Для реализации возможности формирования графических изображений в программировании обычно языковое «ядро» обогащают графической библиотекой, в которой для «исполнителя» описано - как выполнять построение изображений по конкретным инструкциям-функциям и их параметрам.

##  Основные понятия графического режима

Задачи компьютерной графики — воспроизведение (синтез) изображений в тех случаях, когда исходная информация имеет неизобразительную природу. Для графического изображения на экране применяют систему координат несколько отличительная от математической системы – у экранной системы только одна «четверть», в которой абсцисса и ордината только положительные.

 Точкой начала отсчета обоих осей является верхняя левая точка экрана. Считается, что ось Ох направлена вдоль верхнего края экрана вправо, ось Оу направлена вдоль левого края вниз.

Любое изображение формируется из отдельных точек-пикселей. Средствами специальных функций, «исполнитель» просчитывает какие точки отдельного фрагмента изображения должны на экране отобразиться, а какие – нет. Каждая такая функция отвечает за формирование на экране отдельного типа изображения, обычно описываемого несложным математическим выражением. К таким выражениям относятся выражения описания отрезка, прямоугольника, окружности, эллипса, дуги и сектора. Такие инструкции языка программирования принято называть графическими примитивами.

Для удобства понимания и осмысления последовательности «прорисовки» примитивов используют понятие «перо» - считается, что этим пером как карандашом на бумаге и выполняется формирование изображения. В начальный момент построения перо размещается в точке (0,0), а по выполнению отдельной инструкции рисования – перо в конечной точке прорисовки графического примитива. Существуют отдельные инструкции, которые позволяют определить толщину этого пера и стиль отображения линии (сплошная, пунктирная или отдельными точками и т.п.)

 Изображение на экране – это не только контуры и отдельные точки. Это ещё и закрашенные некоторым цветом замкнутые участки экрана. Подобное закрашивание реализуется специальным алгоритмам. А для пользовательского восприятия этого механизма закрашивания его называют «кистью». В некоторых графических библиотеках для кисти определяют стили закрашивания – сплошная закраска, в полоску, в клетку и т.д.

Для формирования изображения важно не только прорисовывать элементы рисунка на экране, но цветность этих элементов. По этой причине важно понимание формирования кода цвета. В простейших графических системах используется фиксированная таблица цветов: некоторые цвета кодируются заданным цветом. Обычно таких цветов всего шестнадцать, включая белый и чёрный цвет. В более современных графических системах для кодирования цвета используется RGB стандартная схема.

1. **Практическая часть**

# 2.1 Подключение графических возможностей

Для того, чтобы воспользоваться возможностью создавать графическое изображение программными средствами языка С++ необходимо в заголовочных файлах указать имя графической библиотеки. Для разных сред разработки существуют разные файлы графических библиотек. В IDE Microsoft Visual Studio для этой цели применяют «windows.h». По своей сути, эта библиотека реализует функции Windows API, призванные создавать различные оконные процедуры и диалоговые окна в стиле Windows. Но тут же присутствует возможность генерирования графического окна в стиле DOS и реализация базовых функций рисования в нём.

Чтобы в окне (в стиле DOS) реализовать графические возможности нужно в главной функции приложения main() , в начальных строках прописать следующие инструкции

HWND hWnd = GetConsoleWindow();//инициация"полотна"

HDC hDC = GetDC(hWnd); // Получаем контекст рисования

Задать перо, его цвет и стиль можно несколькими приёмами:

SelectObject(hDC, GetStockObject(WHITE\_PEN));//перо белое

или HPEN p; // объект «перо»

p=CreatePen(PS\_SOLID,4,RGB(255,0,0));//стиль, толщина, цвет

SelectObject(hDC, p);// выбор объекта к применению

Для указания стиля пера можно указывать следующие значения:

PS\_SOLID – сплошное рисование;

PS\_DASH - пунктирное рисование;

PS\_DOT – рисование точками (штрихами);

PS\_DASHDOT – чередование пунктиров и точек;

PS\_DASHDOTDOT – чередование пунктира и двух точек;

PS\_NULL - перо невидимо.

Все перечисленные стили, кроме сплошного, могут применяться только для толщены не более 1 пикселя.

Задать кисть, её цвет и стиль также можно несколькими приёмами:

SelectObject(hDC, GetStockObject(BLACK\_BRUSH));

Или HBRUSH b; // объект «кисть»

b=CreateHatchBrush(HS\_SOLID, RGB(0,255,0));//сплошная зеленая

SelectObject(hDC, b);// выбор объекта к применению

Для стилей кисти можно применить несколько стилей. Соответствие кода стиля и его внешний вид представлено в таблице 1.

Таблица 1- Соответствие стилей кисти и внешних видов

|  |  |  |  |
| --- | --- | --- | --- |
| Стиль штриховки | Внешний вид | Стиль штриховки | Внешний вид |
| HS\_BDIAGONAL | http://www.frolov-lib.ru/books/bsp/v14/img00036.gif | HS\_FDIAGONAL | http://www.frolov-lib.ru/books/bsp/v14/img00039.gif |
| HS\_CROSS | http://www.frolov-lib.ru/books/bsp/v14/img00037.gif | HS\_HORIZONTAL | http://www.frolov-lib.ru/books/bsp/v14/img00040.gif |
| HS\_DIAGCROSS | http://www.frolov-lib.ru/books/bsp/v14/img00038.gif | HS\_VERTICAL | http://www.frolov-lib.ru/books/bsp/v14/img00041.gif |

А по завершению работы главной функции требуется выполнить очистку памяти от объекта графического окна и его примитивах – элементах рисунка:

DeleteObject(p); // Удалили созданное перо

DeleteObject(b); // Удалили созданную кисть

ReleaseDC(hWnd, hDC);//"Освобождаем" контекст

#  Графические примитивы

Графические примитивы это заранее определенные элементы, которые можно поместить в  чертеж при помощи одной команды. Каждый графический примитив формируется на основании геометрического описания объекта.

Первый параметр всех функции, которые выполняют рисование, обязательно указывает идентификатор того «холста» -контекста отображения, на котором будет выполнятся рисование.

# Точка

Функция рисования точки SetPixel устанавливает цвет точки с заданными координатами:

SetPixel( hdc, XPos, YPos, RGB(0,255,0)); //точка

У точки не бывает других размеров кроме единичного – размер пикселя. Отдельную точку разглядеть на пустом экране довольно сложно.

# Отрезок

Для рисования отдельного отрезка от точки (x1, y1) до точки (x2, y2) в исследуемой системе нет самостоятельной функции. Но выполнение такого построения возможно следующими двумя шагами: установить начальную точку, прорисовать отрезок в конечную точку.

Для установки пера в начальную точку применяют команду

MoveToEx(hdc, x1, y1, NULL);

А для рисования от этой точки отрезка применяют инструкцию:

LineTo( hdc, x2, y2);

Результат работы такой «связки» представлен на рисунке 1.



 Рисунок 1-Отрезок

#  Прямоугольник

Для рисования прямоугольника, со сторонами параллельными сторонам экрана (осям системы рисования) требуется воспользоваться инструкцией.

Rectangle (hdc, x1, y1, x2, y2);

Результат выполнения такой инструкции показан на рисунке 2



Рисунок 2-Прямоугольник

# Эллипс

Для рисования эллипсов и кругов применяется одна и та же инструкция. Исполнитель по этой инструкции рисует округлую фигуру, помещенную (вписанную) в заданный угловыми диагональными точками прямоугольник. Если эти почки характеризуют квадрат, то в результате рисуется круг, а в остальных случаях – эллипс.

Ellipse (hdc, x1, y1, x2, y2);

В зависимости от значений (х1, y1) и (x2, y2) можно получить эллипс, вытянутый горизонтально или вертикально. Возможные варианты отображения эллипсов представлены на рисунке 3.



Рисунок 3- Результаты построения разных видов эллипсов

# Дуга

Возможности рисования кривых линий при помощи стандартных функций ограничены – есть только одна функция, позволяющая рисовать дугу эллипса или окружности:

 Arc (hdc, x1, y1, x2, y2, xStart, yStart, xEnd, yEnd);

Для объяснения параметров обратимся к рисунку 4.



Рисунок 4- Базовые точки для рисования дуги эллипса

Параметры (nxLeft,nyTop) и (nxRight,nyBottom) задают координаты, соответственно, верхнего левого и правого нижнего углов воображаемого прямоугольника, в который вписан эллипс.

Начало дуги эллипса определяется пересечением эллипса с воображаемой прямой линией, проведенной из центра эллипса (xC,yC) в точку (xStart,yStart).

Конец дуги определяется аналогично - как пересечение эллипса с воображаемой прямой линии, проведенной из центра эллипса в точку (xEnd,yEnd).

Дуга рисуется в направлении против часовой стрелки.

Примеры результатов выполнения инструкций с разными параметрами представлены на рисунке 5.



Рисунок 5- Результаты построения различных дуг

# Сектор

Для прорисовки сектора требуется указывать те же параметры что и для рисования дуги, только дополнительно прорисуются отрезки, соединяющие центр фигуры с концами дуги. Что бы это реализовать, нужно применить инструкцию

Pie (hdc, x1, y1, x2, y2, xStart, yStart, xEnd, yEnd);

Примеры результатов выполнения инструкций с разными параметрами представлены на рисунке 6.



Рисунок 6- Результаты построения различный сегментов

# Сегмент

Сегмент – это тоже часть эллипса, но от сектора он отличается тем, что края дуги соединяются единственным отрезком, который не обязательно проходит через центр фигуры. Для рисования сегмента нужно воспользоваться инструкцией.

Chord (hdc, x1, y1, x2, y2, xStart, yStart, xEnd, yEnd);

Параметры в этой инструкции имеют те же назначения, что и при построении дуги и сектора. Примеры результатов выполнения инструкций с разными параметрами представлены на рисунке 7.



Рисунок 7- Результат построения различных сегментов

# Полилиния (ломанная)

Функция, предназначенная для рисования ломаных линий, основывается на массив структур типа «точка», в котором должны находится координаты начала ломаной линии, координаты точек излома и координаты конца ломаной линии, а также размер этого массива. Для выполнения этой функции применяют инструкцию Polyline (hdc, \*lppt, Points); где lppt является указателем на массив, а Points определяет количество точек из этого массива. Функция Polyline не использует текущую позицию пера и не изменяет ее.

Если ломаная линия не замкнута, ее последняя точка не рисуется. Пояснение к приёму отрисовки полилинии представлено на рисунке 8-9.



Рисунок 8- Пояснение к рисованию полилинии



Рисунок 9- Пример результат отображения полилинии

#  Пример применения графических возможностей

# Задание на построение

# Для примера использования компьютерной графики мы возьмем задание для программы КОМПАС-3D.

#  Мы создадим чертеж. Для построения нам необходимо создать сначала лист, на котором будет происходить работа. Делается это так: Находим вкладку «Файл», затем выбираем пункт «создать», после чего выбираем «чертеж».

# Перед нами создался лист, на котором в дальнейшем будет находится наша работы. Начальная стадия работы представлена на рисунке 10.

# https://im0-tub-ru.yandex.net/i?id=b9129191ff63c2652bb842426b2dde67-l&n=13

# Рисунок 10- Начала работы в КОМПАС-3D

Далее мы находим пункт «Инструмены» и в открывшемся окне выбираем инструмент «Отрезок».

После того, как мы выбрали отрезок, нам необходимо выбрать его местоположение, мы можем сделать это двумя способами: 1- Выбрать место начала, нажать левую клавишу мыши и протянуть отрезок на нужное расстояние, после чего повторно нажать левую клавишу мыши, чтобы зафиксировать наш отрезок; 2- открыть вкладку отрезок в нижнем левом углу и задать его кардинаты, после чего использовать функцию «создать объект». Пример выполнения показан на рисунке 11.



Рисунок 11- Пример работы в КОМПАС-3D

Программа КОМПАС-3D предлагает пользователю огромное количество разных инструментов, с помощью которых можно создавать чертежи различной сложности, в которых будут использоваться различные инструменты.

Создадим один из таких чертежей, пример такого чертежа показан на рисунке 12.



Рисунок 12- Чертеж шайбы

Для построения данного чертежа мы использовали такие инструменты как: Окружность, диаметральный размер, прямоугольник, отрезок, штриховка, линейный размер.



# ЗАКЛЮЧЕНИЕ

Основные результаты курсовой работы состоят в следующем:

1 Узнали о применении в языках программирования возможностей реализовывать построение графических изображений, позволяющих делать программы с более приятным для пользователя интерфейсом, отображающие результаты вычислений в наглядной форме.

2 Узнали о возможности реализации графики в консольных приложениях средствами языка С++, освоили приёмы формирования графических примитивов на основе ключевых команда С++ и показали реализацию компьютерной графики в программе КОМПАС-3D.

**СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ**

1. Фролов А. Графический интерфейс GDI в Microsoft Windows/ А.Фролов, Г. Фролов.- Диалог-МИФИ, 1994. -288 c.
2. История создания и развития компьютерных мониторов - Web-3.Создаём вместе. [режим доступа] URL: <http://comp.web-3.ru/accessories/monitors/?act=full&id_article=5849> [18 марта 2017].
3. Компьютерное периферийное оборудование: история развития мониторов. Школьная коллекция. [режим доступа] URL: <http://files.school-collection.edu.ru/dlrstore/39131514-5991-11da-8314-0800200c9a66/index.htm#7> [18 марта 2017].
4. Язык программирования. Википедия - свободная энциклопедия. [режим доступа] https://ru.wikipedia.org/wiki/язык\_программирования. [18 марта 2017]
5. Как рисовать в Win32 API. URL: <http://radiofront.narod.ru/htm/prog/htm/winda/api/paint.html> [18 марта 2017].
6. Перемитина Т.О. Компьютерная графика учебное пособие/ Томск «Эль Контент», 2012. – 142 с.

### И.П. Конакова, И.И. Пирогова «Инженерная и компьютерная графика» / университет - УрФУ, 2014 год- 90 с.

1. Компьютерная графика. URL: <http://grafika.me> [18 марта 2017]