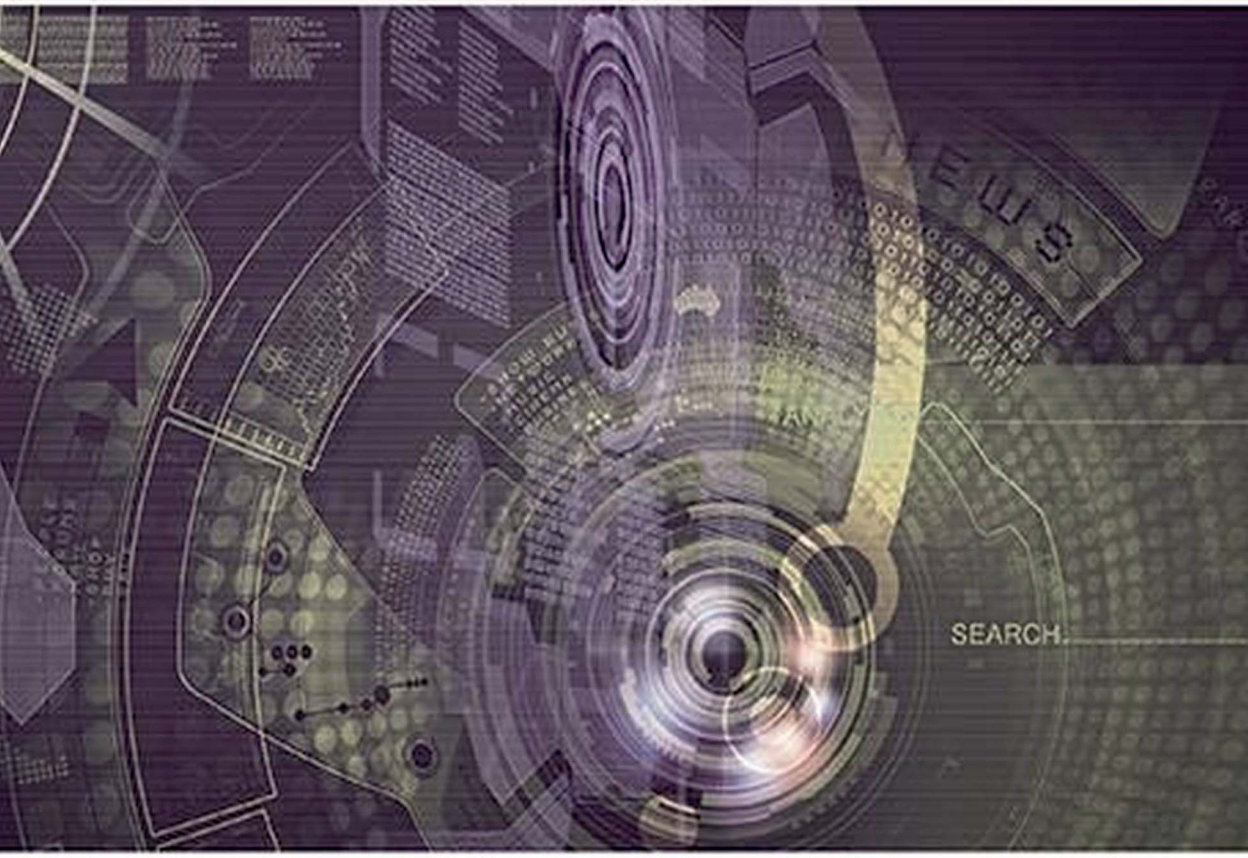
УЧЕБНОЕ ПОСОБИЕ
МГТУ им. Н.Э. БАУМАНА

Ю.Е. Алексеев, А.В. Куров

КОМПЬЮТЕРНАЯ ГРАФИКА В СРЕДЕ MS VS C++



BAUMANPRESS
ИЗДАТЕЛЬСТВО
МГТУ им. Н.Э. БАУМАНА

УДК 681.3.06(075.8)
ББК 32.973.018
А47

Издание доступно в электронном виде на портале *ebooks.bmstu.ru*
по адресу: http://ebooks.bmstu.ru/catalog/199/book1614.html

Факультет «Информатика и системы управления»
Кафедра «Программное обеспечение ЭВМ и информационные технологии»

Рекомендовано Редакционно-издательским советом
МГТУ им. Н.Э. Баумана в качестве учебного пособия

Рецензент канд. техн. наук доцент *С.М. Авдеева*

Алексеев, Ю. Е.

А47 Компьютерная графика в среде MS VS C++ : учебное пособие / Ю. Е. Алексеев, А. В. Куров. — Москва : Издательство МГТУ им. Н. Э. Баумана, 2017. — 98, [2] с. : ил.

ISBN 978-5-7038-4715-2

Рассмотрено создание графических приложений в среде визуального программирования в режиме Common Language Runtime (CLR) — общеязыковой среде выполнения.
Для студентов 1-го курса машино- и приборостроительных специальностей.

УДК 681.3.06(075.8)
ББК 32.973.018

ISBN 978-5-7038-4715-2

1. Основы компьютерной графики

Компьютерная графика — совокупность методов и средств для преобразования данных в графическую форму представления и обратно с помощью ЭВМ.

Компьютерная графика находит самое широкое применение в различных отраслях науки и техники, промышленности, экономике, учебном процессе, органах управления, быту.

Среди систем компьютерной графики выделяют прежде всего следующие:

- иллюстративная графика (создание изображений, играющих роль иллюстративного материала — рисунки, схемы, эскизы, карты). Системы иллюстративной графики должны реализовывать функции, позволяющие «резать», «стирать», «склеивать» различные части изображения, хранить в библиотеке ранее сформированные изображения и вставлять их во вновь создаваемые рисунки, использовать трафареты, выводить изображения различным цветом, осуществлять закраску объектов изображения;
- деловая графика (позволяет в наглядной форме отображать данные, хранимые в таблицах или базах данных, в виде графиков, диаграмм, гистограмм);
- инженерная графика (автоматизация чертежных и конструкторских работ);
- научная графика. Задачи научной графики во многом определяются конкретной научной областью. В географии системы компьютерной графики должны обеспечивать создание и обработку географических и рельефных карт, карт погоды. В математике и химии средства научной графики предоставляют возможность использовать специальную нотацию (формулы) при подготовке документации. В этом случае должны обеспечиваться ввод символов с клавиатуры, генерация представления формул, преобразование формул для подключения системы аналитических преобразований.

Компьютерная графика применяется также в издательском деле, при моделировании, создании тренажеров, в управлении техническими системами и т. д.

В зависимости от направления преобразования данных, способа их визуального представления и типа объектов визуализации выделяют три основные задачи, решаемые средствами компьютерной графики: синтез изображения, анализ изображения, обработка изображения.

При синтезе изображения осуществляется его генерация и вывод на конкретное графическое устройство. При этом используются модели выводимых объектов, в отношении которых применяются операции преобразования (перенос, масштабирование, поворот, проецирование, отсечение). Перед построением изображения выполняются также удаление невидимых линий и поверхностей, закрашивание и затенение объектов сцены.

В процессе анализа изображения решается задача распознавания и выделения элементарных объектов по их абстрактным описаниям.

Обработка изображения предназначена для изменения визуального представления картины с целью улучшения ее качества.

Для построения изображений необходимо иметь специальные технические и программные средства, называемые средствами компьютерной графики. При использовании графического дисплея на основе электронно-лучевой трубки, плазменной панели или жидкокристаллической матрицы поверхность, на которой формируется изображение, представляет собой совокупность отдельных светящихся элементов, т. е. имеет дискретную природу, в то время как выводимые геометрические объекты обладают непрерывным характером.

Таким образом, современная графика является растровой. Растр представляет собой матрицу отдельных пикселей (точек), упорядоченных по строкам и столбцам. Пиксель — наименьший элемент изображения, которому можно индивидуально назначить цвет или степень яркости. Для хранения информации о выводимом изображении требуется достаточно большой объем памяти, поскольку современные дисплеи могут иметь от 600 до 2048 строк по 800—2048 пикселей в каждой строке, а цвет каждого пикселя кодируется тремя байтами.

В связи с тем, что изображение на поверхности дисплея формируется в виде совокупности отдельных светящихся точек, для построения сколь угодно сложного графического изображения достаточно знать цвет каждой точки экрана, т. е. изображение фактически представляет собой мозаику. Однако в силу разной природы исходных и визуализируемых объектов прежде всего требуются разработка и применение соответствующих алгоритмов построения отрезков, окружностей, других кривых, заполнения областей на поверхности экрана.

Другая проблема — это ступенчатость (или лестничный эффект) получаемых изображений. Этот эффект легко видеть на примере изображения простейшего объекта — наклонного отрезка. Оно выглядит именно как совокупность отдельных отрезков-ступенек (в виде лестницы). Только у изображений горизонтальных, вертикальных и наклоненных под углом в 45° отрезков отсутствует лестничный эффект. Принципиально преодолеть этот эффект нельзя, однако применяют различные способы сглаживания изображения для создания у наблюдателя иллюзии отсутствия лестничного эффекта. В связи с бурным прогрессом в области технических средств основным способом уменьшения лестничного эффекта является увеличение разрешающей способности, при этом отдельные точки экрана становятся настолько малыми и настолько близко расположенными друг к другу, что человеческий глаз практически их уже не различает.

Изображение, подлежащее выводу на экран дисплея, может быть представлено совокупностью простейших геометрических фигур (отрезков, прямоугольников, окружностей, эллипсов и т. д.), для высвечивания которых имеются, как правило, готовые подпрограммы (методы классов), входящие в библиотеки подпрограмм современных сред программирования. Однако для правильного использования готовых методов классов, обеспечивающих построение геометрических фигур, следует знать особенности этих средств. Так, готовые средства часто не позволяют вывести геометрический объект

произвольной ориентации на плоскости. В связи с этим следует рассмотреть такое понятие, как параметрическое число геометрического объекта. Параметрическим числом объекта называется минимальное количество параметров, задающих этот геометрический объект. Например, параметрическое число отрезка — четыре, прямоугольника, эллипса — пять.

Однако при использовании методов классов требуется задавать меньшее количество параметров. Это свидетельствует о том, что данные методы обеспечивают вывод фигур частного положения. Например, при построении прямоугольника требуется задать четыре параметра, в этом случае получается прямоугольник со сторонами, параллельными координатным осям, а при построении эллипса — эллипс с осями, также параллельными координатным осям. Данные ограничения следует принимать во внимание при написании программ, в частности, когда строится преобразованное (повернутое) изображение, так как при этом геометрические фигуры приобретают произвольную ориентацию на плоскости. В этом случае необходимо самостоятельно написать фрагмент программы, обеспечивающей построение требуемого изображения. Вывод прямоугольника удобно осуществлять, зная координаты его четырех вершин, а построение кривых обычно выполняется по точкам, принадлежащим кривой, которые соединяются отрезками прямых.

В последнем случае необходимо правильно выбирать количество точек кривой. Поскольку при рисовании кривой осуществляется кусочно-линейная аппроксимация, то малое количество аппроксимирующих отрезков приведет к плохому качеству изображения (кривая явно будет выглядеть в виде ломаной), а большое количество точек — к излишним вычислениям, так как после округления координат получаемых точек будем получать одни и те же точки.

Наилучший результат достигается в том случае, если шаг изменения аргумента при аппроксимации кривой выбирается согласно следующему правилу: при достаточно большом радиусе кривизны кривой две соседние точки кривой должны определяться таким образом, чтобы значение угла (выраженное в радианах), образованного радиусами, проведенными в рассматриваемые точки, было не менее $1/R$, где R — радиус кривизны кривой.

2. Графические возможности среды CLR MS VS

Среда визуального программирования MS VS в полной мере предоставляет пользователю возможность разрабатывать программы, с помощью которых получают графические изображения: схемы, чертежи, текст и иллюстрации. Богатство графических средств Windows связано с так называемым дескриптором контекста графического устройства DC (Device Context). Контекст устройства — это структура данных, определенная Windows и содержащая информацию, которая позволяет Windows транслировать запросы на вывод, поступающие в форме независимых от устройства вызовов функций GDI (Graphical Device Interface), в действия физического вывода на конкретном устройстве.

Вывод на экран в операционной системе Windows требует использования интерфейса графических устройств GDI. Интерфейс GDI позволяет программировать графический вывод независимо от оборудования, на котором он отображается. Это означает, что программа может работать на разных ЭВМ с разными дисплеями. Интерфейс поддерживает также принтеры и плоттеры.

В MS VS имеются классы, упрощающие применение графических инструментов Windows. Основным классом при использовании графики является `Graphics`. Напрямую сформировать указатель на этот класс, чтобы использовать его члены, нельзя. Необходимо сначала получить ссылку на графический объект с помощью специального метода формы `CreateGraphics()`. Это можно сделать, написав следующий оператор:

```
Graphics ^im=this->CreateGraphics();
```

Указанный метод сформирует ссылку на графический объект. Формирование ссылки на класс производится с помощью утилиты `gcnew`, которая вызывает конструктор класса, создающий в выделенной утилитой памяти экземпляр класса, с которым и происходит дальнейшая работа.

При построении графических изображений используются три основных инструмента — перо (карандаш) `Pen`, кисть `Brush` и шрифт `Font`.

2.1. Перо (PEN)

Объект «перо» (`Pen`) предназначен для рисования линии заданной ширины и указанного стиля.

Рассмотрим основные свойства пера. Свойство `Brush` позволяет перу рисовать заполненные линии и кривые. Свойство `Color` позволяет получить или установить цвет пера. В приложении 1 указаны стандартные цвета, их названия и значения в шестнадцатеричной системе счисления при использовании аддитивной `RGB`-модели представления цвета.

Свойство `CustomEndCap` получает или задает настраиваемое окончание линий, нарисованных с помощью объекта `Pen`.

Свойство `CustomStartCap` получает или задает настраиваемое начало линий, нарисованных с помощью объекта «перо». Свойство `DashCap` получает или задает стиль окончания пунктиров, ограничивающих пунктирные линии, нарисованные с помощью объекта `Pen`. Свойство может принимать одно из следующих значений: `Flat` — квадратное завершение для обоих концов каждого штриха; `Round` — круглое завершение для обоих концов каждого штриха; `Triangle` — треугольное завершение для обоих концов каждого штриха.

Свойство `DashOffset` определяет расстояние от начала линии до начала штрихового шаблона.

Свойство `DashStyle` получает или задает стиль, используемый для пунктирных линий, нарисованных с помощью объекта `Pen`. Свойство может принимать одно из следующих значений: `Solid` — сплошная линия; `Dash` — линия,

состоящая из штрихов; Dot — линия, состоящая из точек; DashDot — штрихпунктирная линия; DashDotDot — линия, состоящая из повторяющегося шаблона штрих — две точки; Custom — пользовательский тип пунктирных линий.

Свойство EndCap получает или задает стиль окончания линий, нарисованных с помощью объекта Pen. Это свойство может принимать одно из следующих значений: Flat — плоское завершение отрезка; Square — квадратное завершение отрезка; Round — круглое завершение отрезка; Triangle — треугольное завершение отрезка; NoAnchor — отсутствие маркера; SquareAnchor — квадратный маркер завершения отрезка; RoundAnchor — круглый маркер; DiamondAnchor — маркер в форме ромба; ArrowAnchor — маркер в форме стрелки; Custom — пользовательское завершение отрезка; AnchorMask — задание маски, используемой для проверки того, является ли завершение отрезка маркером.

Свойство PenType получает или задает стиль линий (тип заполнения, который объект Pen использует для заполнения линий), нарисованных с помощью объекта Pen. Это свойство может принимать следующие значения: SolidColor — сплошное заполнение, HatchFill — заполнение штриховкой, TextureFill — заполнение с текстурой точечного рисунка, PathGradient — градиентное заполнение, LinearGradient — линейное градиентное заполнение.

Свойство StartCap получает или задает стиль начала линий, рисуемых с помощью объекта Pen. Это свойство может принимать те же значения, что и свойство EndCap.

Свойство Width получает или устанавливает ширину пера Pen в единицах объекта Graphics, используемого для рисования.

Pen можно создать с помощью одного из следующих конструкторов:

- Pen(Brush^ brush) — инициализирует новый экземпляр класса Pen с указанным объектом кистью brush;
- Pen(Color color) — инициализирует новый экземпляр класса Pen с указанным цветом color;
- Pen(Brush^ brush, float width) — инициализирует новый экземпляр класса Pen с заданными свойствами кистью brush и толщиной width;
- Pen(Color color, float width) — инициализирует новый экземпляр класса Pen с указанными свойствами цветом color и шириной width.

2.2. Цвет (COLOR)

При написании графических приложений требуется назначать цвет используемым инструментам, таким как перо или кисть, поэтому часто удобно предварительно создать объект Color. Это можно сделать следующим образом:

```
Color ^col=gcnew Color();
```

Объекту можно присваивать один из стандартных цветов, указанных в приложении 1. Можно также использовать один из вариантов перегружаемого метода FromArgb. Этот метод создает структуру Color из указанных 8-разрядных значений компонентов ARGB (альфа, красный, зеленый и синий). Альфа-компонент (или альфа-канал) задает прозрачность, причем наибольшее значение 255 соответствует полностью непрозрачному изображению, а наименьшее значение 0 соответствует полностью прозрачному изображению.

Метод FromArgb(Int32) создает цвет, зависящий от порядка байтов 32-разрядного значения ARGB, представленного в виде AARRGGBB. Самый старший байт (MSB), представленный в виде AA, является значением альфа-компонента. Второй, третий и четвертый байты, представленные в виде RR, GG и BB, являются соответственно красным, зеленым и синим компонентами цвета.

Метод FromArgb(Int32, Color) создает структуру Color из указанной структуры Color, но с новым определенным значением альфа (для имеющегося цвета изменяется значение альфа-компонента). Этот метод позволяет передать 32-разрядное значение для значения альфа, однако оно ограничено восемью разрядами.

Метод FromArgb(Int32, Int32, Int32) создает структуру Color из указанных 8-разрядных значений цветов (красный, зеленый, синий). Значение альфа-компонента неявно определено как 255 (полностью непрозрачно). Этот метод позволяет передать 32-разрядное значение для каждого компонента цвета, однако значение каждого из них ограничено интервалом значений от 0 до 255.

Метод FromArgb(Int32, Int32, Int32, Int32) создает структуру Color из четырех значений компонентов ARGB (альфа, красный, зеленый и синий). Этот метод позволяет передать 32-разрядное значение для каждого компонента, однако значение каждого из них ограничено интервалом значений от 0 до 255.

2.3. Кисть (BRUSH)

Объект кисть (Brush) заполняет (закрашивает) область своими выходными данными. Различные кисти имеют разные типы вывода. Некоторые кисти закрашивают область сплошным цветом, другие — градиентом, узором, изображением или рисунком. В следующем списке описаны доступные типы кистей:

- SolidBrush — определяет кисть одного цвета, используется для заливки графических фигур, таких как прямоугольники, эллипсы, круги, многоугольники и контуры;
- SolidColorBrush — заполняет область сплошным цветом Color;
- LinearGradientBrush — заполняет область цветом с линейным градиентом;
- RadialGradientBrush — заполняет область цветом с радиальным градиентом;

• ImageBrush — заполняет область изображением (представленным объектом ImageSource);

• DrawingBrush — заполняет область с помощью рисунка Drawing; этот рисунок может включать векторные и растровые объекты;

• VisualBrush — заполняет область с помощью объекта Visual; объект VisualBrush позволяет дублировать содержимое из одной части приложения в другую область; это может быть полезным при создании эффектов отражения или увеличении части экрана.

Наиболее часто используемой является сплошная кисть, которую можно создать с помощью конструктора SolidBrush(Color color), требующего задания одного параметра — цвета кисти. Это можно сделать, например, следующим образом:

```
SolidBrush ^sb=gcnew SolidBrush(Color::Brown);
```

Кисть SolidColorBrush может быть создана одним из двух следующих конструкторов:

• SolidColorBrush() — инициализирует новый экземпляр класса без цвета и без анимации;

• SolidColorBrush(Color color) — инициализирует новый экземпляр класса SolidColorBrush с указанным значением цвета.

Для заполнения области штриховкой разного вида используется HatchBrush — прямоугольная кисть, заполняющая область штриховкой задаваемого вида (стиля) и задаваемым основным цветом (цветом рисования) и цветом фона. Основной цвет задает цвет линий, цвет фона определяет цвет интервалов между линиями. Имеется 54 уникальных типа штриховки, которые перечислены в приложении 2.

При использовании методов рисования (вывода изображения) часто используются заранее определенные типы данных:

• Point — структура, представляющая собой упорядоченную пару целых чисел — координат X и Y, определяющую точку на двумерной плоскости;

• PointF — структура, представляющая собой упорядоченную пару координат X и Y с плавающей запятой (float), определяющую точку на двумерной плоскости;

• Rectangle — структура, содержащая набор из четырех целых чисел, определяющих расположение и размер прямоугольника. Прямоугольник определяется положением левого верхнего угла, шириной и высотой;

• RectangleF — структура, содержащая набор из четырех действительных чисел, определяющих расположение и размер прямоугольника. Прямоугольник определяется положением левого верхнего угла, шириной и высотой;

• Size — структура, представляющая упорядоченную пару целых чисел, обычно ширину и высоту прямоугольника;

• SizeF — структура, представляющая упорядоченную пару чисел с плавающей запятой, обычно ширину и высоту прямоугольника.

Оглавление

Учебное издание

Алексеев Юрий Евтихович
Куров Андрей Владимирович

Компьютерная графика в среде MS VS C++

Редактор *О.М. Королева*
Художник *Я.М. Ильина*
Корректор *Л.И. Ильина*
Компьютерная графика *М.В. Пинегиной*
Компьютерная верстка *А.Ю. Ураловой*

Оригинал-макет подготовлен в Издательстве МГТУ им. Н.Э. Баумана.

В оформлении использованы шрифты Студии Артемия Лебедева.

Подписано в печать 30.04.2017. Формат 70×100/16.
Усл. печ. л. 8,125. Тираж 50 экз. Изд. № 198-2016. Заказ

Издательство МГТУ им. Н.Э. Баумана.
105005, Москва, 2-я Бауманская ул., д. 5, стр. 1.
press@bmstu.ru
www.baumanpress.ru

Отпечатано в типографии МГТУ им. Н.Э. Баумана.
105005, Москва, 2-я Бауманская ул., д. 5, стр. 1.
baumanprint@gmail.com