**Лекция 7.**

**Отображение графики**

Библиотека MFC предоставляет разработчикам все необходимые классы, которые инкапсулируют соответствующие графические объекты Windows . Кроме того, библиотека имеет в своем составе дополнительные классы, не имеющие непосредственного отношения к графике, но значительно облегчающие решение ряда задач (классы CPoint, CSize, CRecf, CRectTracker). Все классы имеют один и тот же базовый класс, который позволяет унифицировать работу с графическими объектами как таковыми (CGdiObject). Вот краткий обзор имеющихся классов.

CGdiObject - базовый класс для всех классов, обеспечивающих интерфейс с графическими объектами Windows. Хотя этот класс и не является абстрактным, нет никакого смысла в непосредственном создании его объектов, т. к. он не ассоциирован ни с каким графическим объектом Windows.

CPen - инкапсулирует объект Windows "карандаш", который может быть назначен для контекста устройства и использоваться для определения типа и цвета линий или границ фигур.

CBrush - инкапсулирует объект Windows "кисть", который может быть назначен для контекста устройства и использоваться для определения типа и цвета заливки внутренних областей замкнутых фигур.

CFont — инкапсулирует объект Windows "шрифт", который может быть назначен для контекста устройства и использоваться при операциях вывода текстовой информации.

CBitmap - инкапсулирует объект Windows "битовый массив", который обеспечивает приложение рядом функций. Используется в том числе для работы с контекстами устройств, расположенными в памяти.

CPalette - инкапсулирует объект Windows "цветовая палитра" (или просто "палитра"), который обеспечивает интерфейс между приложением и цветным устройством вывода, таким как экран дисплея. Этот интерфейс позволяет приложению в полной мере использовать возможности устройства вывода, не нарушая при этом работу (графический вывод) других приложений. "Палитра"

служит в Windows для определения используемых приложением цветов из набора доступных в системе.

CRgn инкапсулирует объект Windows "регион", который хотя и относится к графическим объектам, на самом деле является скорее дополнительным объектом или, точнее, средством управления графическим выводом, определяя сложные (не только прямоугольные) области.

**Рисование линий**

**Текущая позиция**

В контексте устройства вы рисуете сущности вроде линий, окружностей и текста относительно текущей позиции.

Текущая позиция это точка в клиентской области, которая была установлена ранее, после рисования предыдущей сущности, либо установлена явным вызовом функции, которая для этого предназначена.

*pDC->MoveTo (50, 50) ;* // Установить текущую позицию в точку 50, 50

В этом операторе происходит вызов метода Move То () объекта СDC, на который указывает pDC. Этот метод просто устанавливает текущую позицию в координаты х и у, определенные в аргументах. Режимом отображения по умолчанию является ММ\_ТЕХТ, так что координаты выражаются в пик­селях, и текущая позиция будет установлена в точку, отстоящую на 50 пикселей вправо от левого края окна и на 50 пикселей вниз от верхнего края клиентской области.

Класс СDC переопределяет функцию Move То () для обеспечения гибкости в указа­нии позиции, которую вы хотите указать в качестве текущей. Существуют две версии этой функции, объявленные в классе СDC следующим образом:

*CPoint MoveTo(int х, int у);* // Перейти в позицию х, у

*CPoint MoveTo(CPoint aPoint);* // Перейти в позицию, определенную aPoint

Первая версия принимает координаты х и у как отдельные аргументы. Вторая принимает один аргумент типа CPoint, который представляет собой объект класса CPoint.

Класс CPoint имеет поля х и у типа LONG, и при­менение объектов CPoint обладает тем преимуществом, что в классе также определе­ны методы для операций с объектами типа CPoint.

Возвращаемым значением функции Move То () является объект CPoint, который специфицирует текущую позицию, какой она была перед перемещением. Можно подумать, что это несколько странно, но представьте ситуацию, когда нужно пе­рейти в новую позицию, нарисовать что-либо, а затем вернуться обратно. Текущей позиция перед перемещением может быть неизвестна, и после того, как перемещение про­изойдет, она будет утеряна, так что вам предоставляется возможность вернуться в ис­ходную позицию, когда это необходимо.

Часто за вызовами Move То () следуют вызовы функции Line То (), которая рисует линию в клиентской области от текущей позиции в точку, специфици­рованную ее аргументами, как проиллюстрировано на рис. 11.2.1.

Класс СDC также определяет две версии функции Line То () со следующими про­тотипами:

*BOOL LineTo(int х, int у);* // Рисовать линию до позиции х, у

*BOOL LineTo(CPoint aPoint);* // Рисовать линию до позиции, определенной aPoint

Это предлагает вам такую же гибкость в указании аргумента, как и Move То (). Вы мо­жете использовать объект С Point в качестве аргумента второй версии функции. Функ­ция возвращает TRUE, если линия была нарисована, и FALSE — в противном случае.

Когда выполняется функция LineTo (), текущая позиция изменяется в точку, ука­занную в конце линии. Это позволяет рисовать серии соединенных линий простыми последовательными вызовами LineTo () для каждой из них. Взгляните на следующий фрагмент кода:

……

*pDC->MoveTo (50, 50);* // Установить текущую позицию

*pDC->LineTo (50, 200);* // Нарисовать вертикальную линию вниз на 150 единиц

*pDC->LineTo (150,200);* // Нарисовать горизонтальную линию вправо на 100 единиц

*pDC->LineTo (150,50);* // Нарисовать вертикальную линию вверх на 150 единиц

*pDC->LineTo (50,50) ;* // Нарисовать горизонтальную линию влево на 100 единиц

…….

Если вы включите этот фрагмент в программу то в окне четыре вызова функции LineTo () обеспечат рисование прямоугольника против часовой стрелки, начиная с верхнего левого угла. Первый вызов использует текущую позицию,

установленную функцией Move То (); последующие вызовы применяют текущую позицию, установленную предыдущим вызовом функции LineTo (). Можно использовать это для рисования любой фигуры, состоящей из последовательности линий, каждая из которых соединена с предыдущей. Конечно, можно также в любой момент использовать MoveTo () для изменения текущей позиции.

**Рисование эллипсов и кривых**

Существует выбор среди нескольких методов класса СDC для рисования окружностей, но все они предназначены для рисования эллипсов.

Как известно из школьного курса геометрии, окружность — это частный случай эллипса, у кото­рого большая и малая оси равны. Поэтому можно использовать метод Ellipse () для рисования окружности. Как и все другие замкнутые фигуры, поддер­живаемые классом СDC, метод Ellipse () заполняет свою внутреннюю часть уста­новленным вами цветом. Внутренний цвет определяется кистью (brush), выбранной в контексте устройства. Текущая кисть контекста устройства определяет способ за­полнения замкнутых фигур.

Библиотека MFC предлагает класс CBrush, который служит для определения ки­сти. Можно установить цвет объекта CBrush, а также определить шаблон заполнения замкнутых фигур. Если вы хотите нарисовать незакрашенную замкнутую фигуру, мож­но использовать пустую кисть, которая оставит внутреннюю часть фигуры пустой. Чуть позже мы еще вернемся к разговору о кистях.

Другой способ рисования незакрашенных окружностей предполагает применение функции Arc (), которая не работает с кистями. Преимущество ее состоит в том, что вы можете рисовать любые дуги и эллипсы, а не только замкнутые.

Доступны две вер­сии этой функции в классе СDC:

*BOOL Arc(int xl, int yl, int x2, int y2, int x3, int y3, int x4, int y4);*

*BOOL Arc(CRect\* pRect, CPoint StartPt, CPoint EndPt);*

В первой версии (xl, yl) и (x2, y2) определяют верхний левый и нижний пра­вый углы прямоугольника, в который вписана полная кривая. Если вы укажете в этих координатах углы квадрата, то нарисованная кривая будет сегментом

окружности. Точки (хЗ, уЗ) и (х4, у4) определяют начальную и конечную точки рисуемого сег­мента. Сегмент рисуется против часовой стрелки. Если вы определите координаты (хЗ, уЗ) идентичными (х4, у4), то получите полную замкнутую кривую.

Во второй версии Arc () описанный прямоугольник задается объектом CRect, и ука­затель на этот объект передается в первом аргументе. Функция также принимает ука­затель на объект класса CRect, имеющий четыре общедоступных поля: left, top, right и bottom. Они соответствуют координатам х и у левой верхней и правой нижней точек соответственно. Класс также предлагает множество методов, оперирующих объектами CRect, часть из которых мы используем чуть позже.

Объекты CPoint с именами StartPt и EndPt во второй версии Arc () определяют начальную и конечную точки дуги, которую нужно нарисовать.

Рассмотрим пример кода, использующего обе версии функции Arc ().

*pDC->Arc(50,50,150,150,100,50,150,100);* // Нарисовать 1-ю (большую) окружность

// Определить ограничивающий прямоугольник для 2-й (меньшей) окружности *CRect\* pRect = new CRect(250,50,300,100);*

*CPoint Start(275,100);* // Начальная точка дуги

*CPoint End(250,75);* // Конечная точка дуги

*pDC->Arc(pRect,Start, End);* // Нарисовать вторую окружность

*delete pRect;*

Функция Arc () не требует установки текущей позиции, поскольку положение и размер дуги полностью опре­деляются передаваемыми ей аргументами. Текущая позиция при рисовании дуги не важна — она остается на прежнем месте, где и была до начала рисования дуги. Хотя координаты могут быть до 32 768, максимальная ширина и высота прямоугольника, ограничивающего фигуру, составляет 32 767, потому что это максимальное положи­тельное значение, которое может быть представлено 16-битным целым числом.

Основные (хотя и не все) функции-члены CDC, предназначенные для рисования отрезков и кривых, приведены в таблице .

Функции-члены CDC для рисования отрезков и кривых

|  |  |
| --- | --- |
| **Функция** | **Назначение** |
| MoveTo | Задает текущую позицию |
| LineTo | Рисует отрезок из текущей позиции в заданную точку и смещает в нее текущую позицию |
| Polyline | Последовательно соединяет набор точек отрезками |
| PolylineTo | Соединяет набор точек отрезками прямых, начиная с текущей позиции. Текущая позиция смещается в последнюю точку набора. |
| Arc | Рисует дугу |
| ArcTo | Рисует дугу и смещает текущую позицию в конец дуги |
| PolyBezier | Рисует один или несколько сплайнов Безье |
| PolyBezierTo | Рисует один или несколько сплайнов Безье и помещает текущую позицию в конец последнего сплайна |
| PolyDraw | Рисует набор отрезков и сплайнов Безье через набор точек и смещает текущую позицию в конец последнего отрезка или сплайна |

В GDI есть функции для рисования более сложных примитивов, чем отрезки и кривые. Некоторые из них перечислены в таблице.

Функции-члены CDC для рисования замкнутых фигур

|  |  |
| --- | --- |
| **Функция** | **Описание** |
| Chord | Замкнутая фигура, образованная пересечением эллипса и отрезка |
| Ellipse | Эллипс или окружность |
| Pie | Сектор круговой диаграммы |
| Polygon | Многоугольник |
| Rectangle | Прямоугольник |
| RoundRect | Прямоугольник с закругленными углами |