**Лабораторная работа №3.2 (MFC)**

**1) Логические единицы в системах, изотропный и анизотропный режимы отображения**

**ФИЗИЧЕСКИЕ И ЛОГИЧЕСКИЕКООРДИНАТЫ**

**Основные определения**

**Физические координаты**, как это следует из названия, имеют непосредственное отношение к физическому устройству вывода. В качестве единицы измерения длины в системе физических координат всегда используется пиксел. Если устройством вывода является экран монитора, физические координаты обычно называют экранными координатами.

**Логические координаты** передаются функциям GDI, выполняющим рисование фигур или вывод текста. Используемые единицы измерения зависят от режима отображения.

*При отображении GDI преобразует логические координаты в физические.* Способ преобразования зависит от режима отображения и других атрибутов контекста отображения, таких как расположение начала системы координат для окна, расположение начала системы физических координат, масштаб осей для окна и масштаб осей физических координат.

**GDI** (Graphics Device Interface, **Graphical Device Interface**) — один из трёх основных компонентов или «подсистем», вместе с ядром и Windows API, составляющих пользовательский интерфейс (оконный менеджер GDI) Microsoft Windows.

GDI — это интерфейс Windows для представления графических объектов и передачи их на устройства отображения, такие, как мониторы и принтеры.

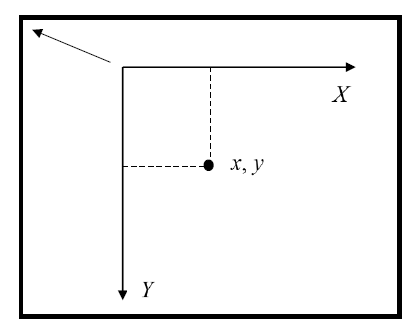
GDI отвечает за отрисовку линий и кривых, отображение шрифтов и обработку палитры. Он не отвечает за отрисовку окон, меню и т. п., эта задача закреплена за пользовательской подсистемой, располагающейся в user32.dll и основывающейся на GDI. GDI выполняет те же функции, что и QuickDraw в Mac OS.

Одно из преимуществ использования GDI вместо прямого доступа к оборудованию — это унификация работы с различными устройствами. Используя GDI, можно одними и теми же функциями рисовать на разных устройствах, таких, как экран или принтер, получая на них практически одинаковые изображения. Эта возможность лежит в центре всех WYSIWYG-приложений для Windows.

**Физическая система координат**

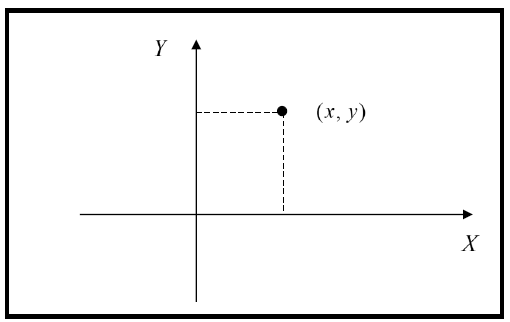
На рис. показана физическая система координат для экрана видеомонитора.

Начало этой системы координат располагается в левом верхнем углу экрана. Ось X направлена слева направо, ось Y – сверху вниз. В качестве единицы длины в данной системе координат используется пиксел.



**Логическая система координат**

Приложения Windows могут использовать одну из нескольких логических координат, устанавливая соответствующий режим отображения в контексте отображения. При этом можно использовать любое направление координатных осей и любое расположение начала координат. Например, возможна система координат, в которой задаются положительные и отрицательные координаты по любой оси.



Для установки режима отображения, непосредственно определяющего направление осей и размер логической единицы системы координат, используется функция **SetMapMode**:

*int dc.SetMapMode(int nMapMode) // MFC*

Параметр nMapMode может принимать одно из следующих значений, представленных в таблице:

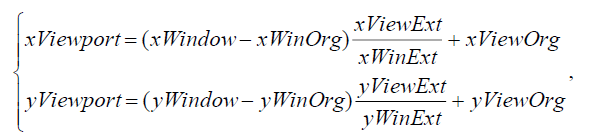
**Значения параметра nMapMode**

|  |  |  |  |
| --- | --- | --- | --- |
| Режим отображения | Направление оси X | Направление  оси Y | Размер одной логической единицы |
| MM\_TEXT | Вправо | Вниз | 1 пиксел |
| MM\_LOMETRIC | Вправо | Вверх | 0,1 мм |
| MM\_HIMETRIC | Вправо | Вверх | 0,01 мм |
| MM\_LOENGLISH | Вправо | Вверх | 0,01 дюйм |
| MM\_HIENGLISH | Вправо | Вверх | 0,001 дюйм |
| MM\_TWIPS | Вправо | Вверх | 1/1440 дюйма |
| MM\_ISOTROPIC | Можно выбирать | Можно выбирать | Произвольный, одинаковый для осей X и Y |
| MM\_ANISOTROPIC | Можно выбирать | Можно выбирать | Произвольный, может быть разный для осей X и Y |

Преобразование пространства страницы в пространство устройства определяет режим отображения для всего вывод графических данных, связанных с частным контекстом устройства (DC). Режим отображения - это преобразование масштабирования, которое определяет размер единиц, используемых для операции рисования. Режим отображения (mapping mode) может также выполнить перенос.

**Преобразование координат**

Приложение, вызывая для рисования функции GDI, указывает логические координаты. Перед выводом GDI преобразует их в физические с использованием следующих формул:



где

xWindow, yWindow – логические координаты по оси X и Y соответственно,

yViewport, xViewport – физические (экранные) координаты по оси X и Y

соответственно,

xWinOrg, yWinOrg – определяют расположение начала логической си-

стемы координат, по умолчанию xWinOrg = 0, yWinOrg = 0 ,

xViewOrg, yViewOrg – определяют расположение начала физической си-

стемы координат, по умолчанию xViewOrg = 0, yViewOrg = 0,

xViewExt, xWinExt – задают масштаб, который используется в процессе

преобразования координат по оси x ,

yViewExt, yWinExt – задают масштаб, который используется в процессе

преобразования координат по оси y .

Эти масштабы зависят от установленного режима отображения. Приложения могут изменить его только в режимах MM\_ISOTROPIC и

MM\_ANISOTROPIC, для остальных режимов отображения используются фиксированные значения.

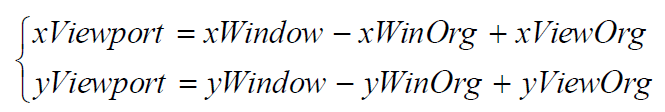
Таким образом, логические координаты (xWindow, yWindow) преобразуются в физические координаты (xViewport, yViewport).

**Режимы отображения**

После рассмотрения физических и логических координат, а также их преобразований, займемся подробным описанием каждого режима отображения.

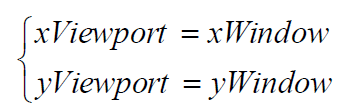
**Режим MM\_TEXT**

Режим отображения MM\_TEXT устанавливается в контексте отображения по умолчанию. Для этого режима формулы преобразования координат упрощаются:

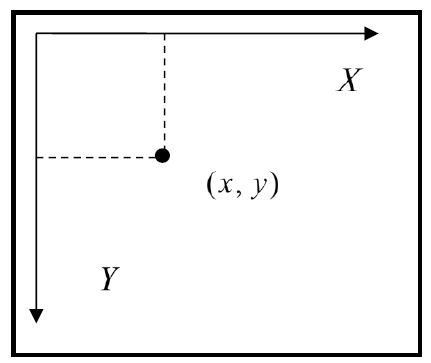


Так как по умолчанию xViewOrg = 0, yViewOrg = 0 , xWinOrg = 0 ,

yWinOrg = 0 , то



Соответствующая система координат представлена на рис. (начало системы координат расположено точно в левом верхнем углу внутренней области окна, рисунок иллюстрирует только направление координатных осей).



Так как в формуле преобразования не присутствуют переменные xViewExt , xWinExt и yViewExt , yWinExt , в данном режиме преобразования невозможно изменить масштаб осей координат. Поэтому логическая единица длины в режиме отображения MM\_TEXT равна физической, т. е. одному пикселу.

**Метрические режимы отображения**

К метрическим режимам отображения относятся режимы MM\_LOMETRIC,

MM\_HIMETRIC, MM\_LOENGLISH, MM\_HIENGLISH и MM\_TWIPS.Эти режимы позволяют использовать привычные единицы измерения, такие как миллиметры и дюймы.

В метрических режимах отображения используются полные формулы преобразования координат, приведенные выше. В этих формулах приложение может изменять переменные, определяющие смещение начала физической или логической системы координат ( xViewOrg , yViewOrg ) и ( xWinOrg , yWinOrg ).

Приложение не может изменить значения переменных xViewExt , xWinExt и yViewExt , yWinExt , от которых зависит масштаб по осям координат. Отношения xViewExt/xWinExt и yViewExt/yWinExt имеют фиксированное значение для каждого из метрических режимов отображения.

Заметим, что для этих режимов отношение yViewExt/yWinExt имеет отрицательный знак, в результате чего ось Y оказывается направленной снизу вверх.

Остановимся подробнее на описании режимов **MM\_ISOTROPIC** и **MM\_ANISOTROPIC**.

Режимы отображения MM\_ISOTROPIC (**изотропный**) и MM\_ANISOTROPIC (**анизотропный**) допускают изменение направления осей X и Y, а также изменение масштаба осей координат. В изотропном режиме отображения MM\_ISOTROPIC масштаб вдоль осей X и Y всегда одинаковый (т. е. для обеих осей используются одинаковые логические единицы длины). Анизотропный режим MM\_ANISOTROPIC предполагает использование разных масштабов для разных осей (хотя можно использовать и одинаковые масштабы).

Для изменения ориентации и масштаба осей можно воспользоваться функциями SetViewportExt и SetWindowExt, которые являются методами класса CDC из библиотеки MFC:

*virtual CSize SetWindowExt(int cx, int cy),*

*virtual CSize SetViewportExt(int cx, int cy).*

Функция SetWindowExt устанавливает для формулы преобразования координат значения переменных xWinExt и yWinExt .

Функция SetViewportExt устанавливает для формулы преобразования координат значения переменных xViewExt и yViewExt . Функция **SetViewportExt** должна использоваться **после функции SetWindowExt**.

**Изотропный режим** отображения удобно использовать в тех случаях, когда надо сохранить установленное отношение масштабов осей X и Y при любом изменении размеров окна, в которое выводится изображение.

**Анизотропный режим** удобен в тех случаях, когда изображение должно занимать всю внутреннюю поверхность окна при любом изменении размеров окна. Соотношение масштабов при этом не сохраняется.

**2) Кисть, перо, как задается**

Для рисования графики GDI предлагает два инструмента кисти и перья. Главное отличие в том, что перо позволяет рисовать линии заданной толщины, а кисть заливает замкнутую фигуру. По-русски, закрашивает. Кисти можно делать любые, сплошные, трафаретные и даже из растровых файлов.

**Перо**

Когда вы объявляете переменную CDC , она создает и выбирает перо, которое может рисовать черную линию шириной в 1 пиксель. Если вам нужно более совершенное перо, MFC предоставляет класс CPen . Поэтому первым шагом в создании пера является объявление переменной типа CPen, что можно сделать с помощью конструктора по умолчанию следующим образом:

*CPen NewPen;*

Для создания пера необходимо указать желаемые характеристики. Это можно сделать с помощью другого конструктора CPen , объявленного следующим образом:

*CPen (int nPenStyle, int nWidth, COLORREF crColor);*

Параметры  
**nPenStyle**  
Определяет перьевой стиль. Этот параметр в первой версии конструктора может быть одно из следующих значений:

* **PS\_SOLID** - Создает сплошное перо.
* **PS\_DASH** - Создает пунктирное перо. Допустимый только, когда перьевая ширина 1 или меньше, в модулях устройства.
* **PS\_DOT** - Создает точечное перо. Допустимый только, когда перьевая ширина 1 или меньше, в модулях устройства.
* **PS\_DASHDOT** - Создает перо с чередованием, черточки и точки. Допустимый только, когда перьевая ширина 1 или меньше, в модулях устройства.
* **PS\_DASHDOTDOT** - Создает перо с чередованием, черточки и двойные точки. Допустимый только, когда перьевая ширина 1 или меньше, в модулях устройства.
* **PS\_NULL** - Создает пустое перо.
* **PS\_INSIDEFRAME** - Создает перо, которое выводит линию внутри рамки закрытых форм, произведенных Windows GDI функциями вывода, которые определяют ограничительный прямоугольник (например, Ellipse, Rectangle, RoundRect, Pie, и Chord функции. Когда этот стиль используется с Windows GDI функции вывода, которые не определяют ограничительный прямоугольник (например, LineTo функция), область рисунка пера не ограничена рамкой.

**nWidth**  
Определяет ширину пера.

**crColor**  
Содержит **RGB** цвет для пера.

**Кисть**

Надо сделать класс пера для контура. Параметры описывать не буду, не о том речь.

*CPen pen(PS\_SOLID, 5, RGB(0,0,255));*

Выберем перо и сохраним старое

*CPen\* oldPen = pDC->SelectObject(&pen);*

Теперь класс кисти.

*CBrush brush(HS\_BDIAGONAL, RGB(255,0,0));*

У него несколько конструкторов в зависимости от того, чего Вам надо. А чего Вам надо? Мне штриховку лично.

*CBrush( ); // просто так ;-)*

*CBrush( COLORREF crColor ); // сплошным цветом*

*throw( CResourceException );*

*CBrush( int nIndex, COLORREF crColor ); // со штриховкой*

*throw( CResourceException );*

*CBrush( CBitmap\* pBitmap ); // растровой картинкой*

*throw( CResourceException );*

Как видите я выбрал третий вариант со штриховкой. Все варианты шриховок смотрите в подробностях.

Выбираем для рисования кисть.

*CBrush\* oldBrush =pDC->SelectObject(&brush);*

Пора рисовать

*pDC->Rectangle(10,10,200,200);*

Теперь надо все вернуть на место.

*pDC->SelectObject(oldPen);*

*pDC->SelectObject(oldBrush);*

Можно собирать и запускать. А вот и результат.

