




Лекция №6

Кросс-платформенное программирование

- 
- Рисование на холсте, класс wxDC
 - Масштабирование изображения

wxDC

- Класс wxDC представляет собой «Контекст устройства», на которое выводится графика и текст.
- Класс wxDC предназначен для работы с различными устройствами вывода и предлагает общие абстрактные API для рисования и вывода текста.

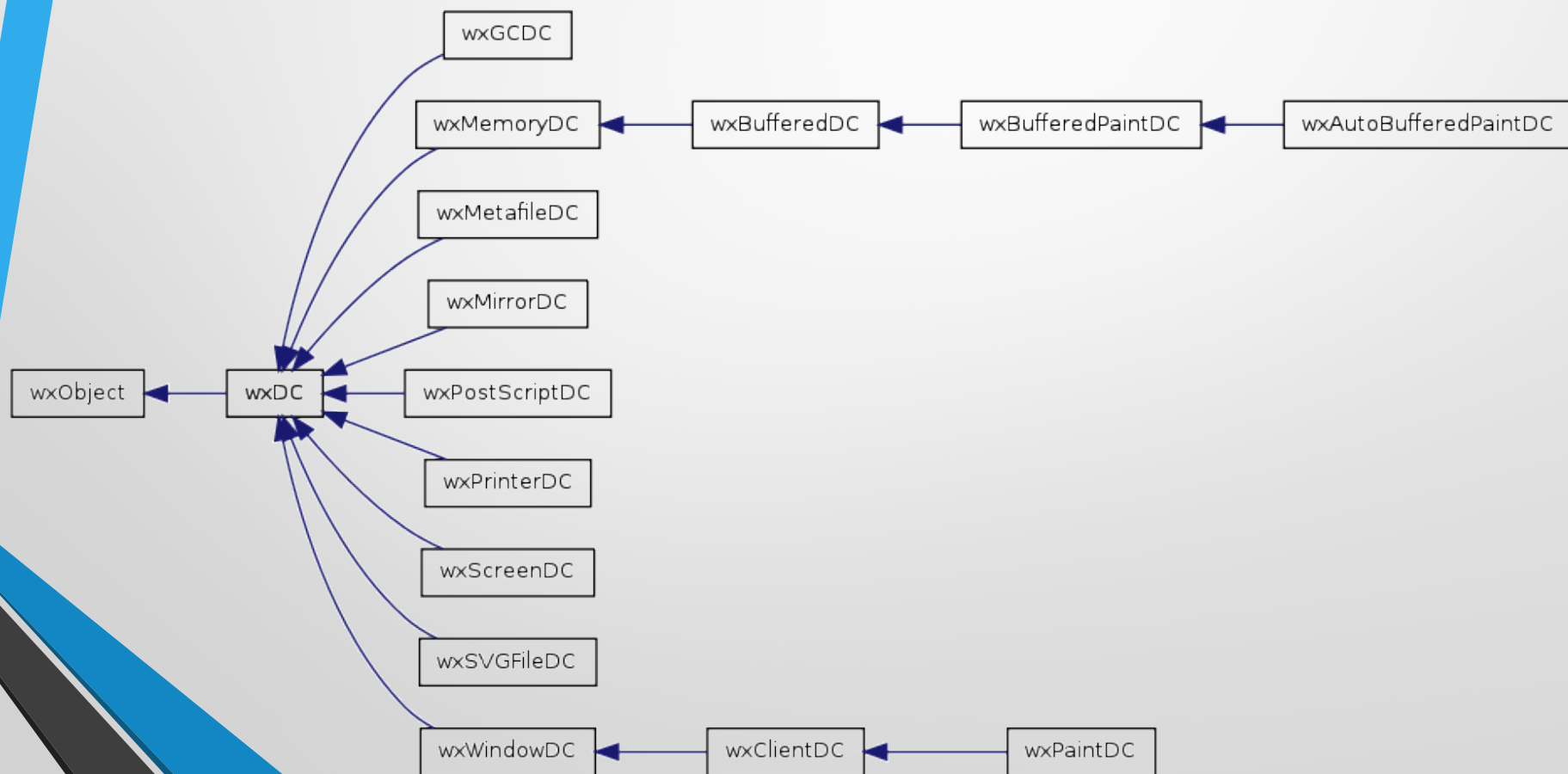
wxDC

- wxDC является абстрактным базовым классом и не может быть создан непосредственно. Поэтому для работы с графикой используются следующие классы: wxPaintDC , wxClientDC , wxWindowDC , wxScreenDC , wxMemoryDC или wxPrinterDC .
- Класс wxPaintDC (библиотека wx/dcclient.h) используется, если необходим вывод на экран в обработчике события EVT_PAINT() (OnPaint), для рисования вне этого события используется класс wxClientDC .

wxDC

- wxWindowDC (библиотека wx/dcclient.h) используется для рисования на всей области окна, как на клиентской области, так и на заголовке окна.
- wxScreenDC (библиотека wx/dcclient.h) используется для рисования на всем экране как таковом.
- wxMemoryDC (библиотека wx/dcmemory.h) предназначен для вывода графики на растровое изображение в памяти.
- wxPrinterDC (библиотека wx/dcprint.h) предназначен для вывода графики и текста на принтер

Диаграмма наследования wxDC



Некоторые методы wxDC

- `void Clear()` -- Очищает холст заливая его текущей кистью
- `void DrawArc(wxCoord x1, wxCoord y1, wxCoord x2, wxCoord y2, wxCoord xc, wxCoord yc)` -- Рисует дугу окружности, с центром (xc, yc) , с началом в точке $(x1, y1)$ и окончанием $(x2, y2)$. Дуга рисуется против часовой стрелки от начальной до конечной точки.
- `void DrawCircle(wxCoord x, wxCoord y, wxCoord radius)` -
- Рисует окружность с центром (x, y) и заданным радиусом
- `void DrawEllipse(wxCoord x, wxCoord y, wxCoord width, wxCoord height)` -- Рисует эллипс, содержащиеся в прямоугольник, заданный координатами верхнего левого угла и его шириной и высотой.

Некоторые методы wxDC

- `void DrawEllipticArc(wxCoord x, wxCoord y, wxCoord width, wxCoord height, double start, double end)` – Рисует дугу эллипса. x и y координаты левого верхнего угла прямоугольника, содержащего эллипс. *width* и *height* задают ширину и высоту прямоугольника, содержащего эллипс. *start*, *end* начало и конец дуги относительно трех-часовой позиции от центра прямоугольника. Углы задаются в градусах
- `void DrawLine(wxCoord x1, wxCoord y1, wxCoord x2, wxCoord y2)` – Рисует линию из т. (x_1, y_1) в т. (x_2, y_2)
- `void DrawPolygon(int n, wxPoint points[], wxCoord xoffset = 0, wxCoord yoffset = 0, int fill_style = wxODDEVEN_RULE)` – Рисует закрашенный многоугольник с помощью массива точек размером n

Некоторые методы wxDC

- `void DrawPoint(wxCoord x, wxCoord y)` – Рисует точку с координатами (x, y)
- `void DrawRectangle(wxCoord x, wxCoord y, wxCoord width, wxCoord height)` -- Рисует прямоугольни из точки (x, y) шириной *width*, высотой *height*
- `void DrawRotatedText(const wxString& text, wxCoord x, wxCoord y, double angle)` -- Рисует текст повернутым на *angel* градусов.
- `void DrawSpline(wxCoord x1, wxCoord y1, wxCoord x2, wxCoord y2, wxCoord x3, wxCoord y3)` -- Рисует сплайн через 3 точки

Некоторые методы wxDC

- `void DrawText(const wxString& text, wxCoord x, wxCoord y)` - Выводит текст в указанной точке.
- `bool FloodFill(wxCoord x, wxCoord y, const wxColour& colour, int style=wxFLOOD_SURFACE)` - Выполняет заливку контекста устройства, начиная с указанной точки, используя текущий цвет кисти, и с помощью стиля: `wxFLOOD_SURFACE`: заполнение происходит, пока не встретится цвета отличный от заданного цвета. `wxFLOOD_BORDER`: область, которая будет закрашена ограничена указанным цветом.
- `wxBrush& GetBrush()`
- `void SetBrush(const wxBrush& brush)`- Возвращает/устанавливает текущую кисть

```
wxPaintDC dc(this);

wxPoint lines[] = { wxPoint(20, 260),
wxPoint(100, 260), wxPoint(20, 210), wxPoint(100,
210) };

wxPoint polygon[]={wxPoint(130, 140),
wxPoint(180, 170), wxPoint(180, 140),
wxPoint(220, 110), wxPoint(140, 100) };

wxPoint splines[] = { wxPoint(240, 170),
wxPoint(280, 170), wxPoint(285, 110),
wxPoint(325, 110) };

dc.DrawEllipse(20, 20, 90, 60);

dc.DrawRoundedRectangle(130, 20, 90, 60, 10);

dc.DrawArc(240, 40, 340, 40, 290,
20);

dc.DrawPolygon(4, polygon);

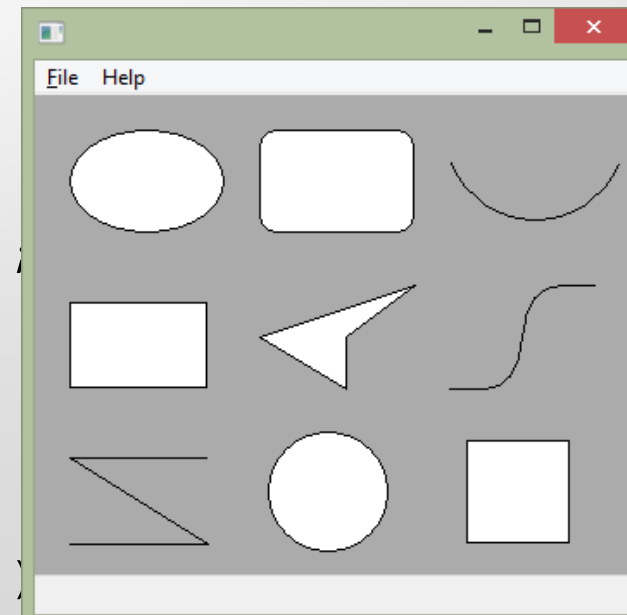
dc.DrawRectangle(20, 120, 80, 50);

dc.DrawSpline(4, splines);

dc.DrawLine(4, lines);

dc.DrawCircle(170, 230, 35);

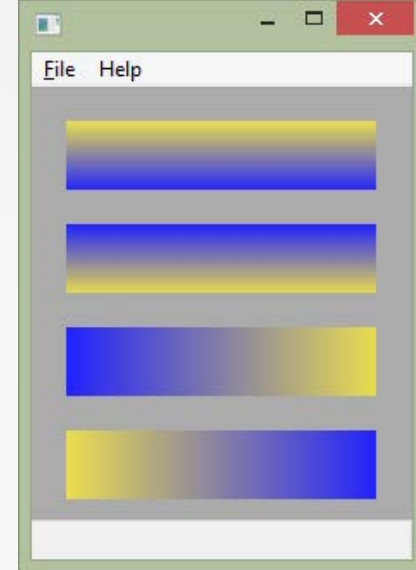
dc.DrawRectangle(250, 200, 60, 60);
```



Некоторые методы wxDC: Градиенты

- `void wxDC::GradientFillConcentric (const wxRect & rect, const wxColour & initialColour, const wxColour & destColour)` - Заполните области, заданной прямоугольником радиальным градиентом, начиная с `initialColour` в центре круга и переходящих в `destColour` на краях окружности.
- `void wxDC::GradientFillLinear (const wxRect & rect, const wxColour & initialColour, const wxColour & destColour, wxDirection nDirection = wxRIGHT)` - Заполните области, заданной прямоугольником линейным градиентом

```
wxPaintDC dc(this);  
wxColour col1, col2;  
col1.Set(wxT("#2222FE"));  
col2.Set(wxT("#ebde4f"));  
dc.GradientFillLinear(wxRect(20, 20, 180,  
40), col1, col2, wxNORTH);  
dc.GradientFillLinear(wxRect(20, 80, 180,  
40), col1, col2, wxSOUTH);  
dc.GradientFillLinear(wxRect(20, 140,  
180, 40), col1, col2, wxEAST);  
dc.GradientFillLinear(wxRect(20, 200,  
180, 40), col1, col2, wxWEST);
```



Перо является элементарным графическим объектом. Оно используется для рисования линий, кривых и контуров прямоугольников, эллипсов, многоугольников и других объектов.

```
wxPen(const wxColour& colour, int width = 1,  
int style = wxSOLID)
```

Конструктор `wxPen` имеет три параметра. Цвет, толщину пера и стиль. Ниже приведен перечень возможных стилей пера.

- ✓ `wxDOT` (пунктир)
- ✓ `wxSOLID` (сплошной)
- ✓ `wxLONG_DASH` (длинное тире)
- ✓ `wxSHORT_DASH` (короткое тире)
- ✓ `wxDOT_DASH` (точка тире)
- ✓ `wxTRANSPARENT` (прозрачный)

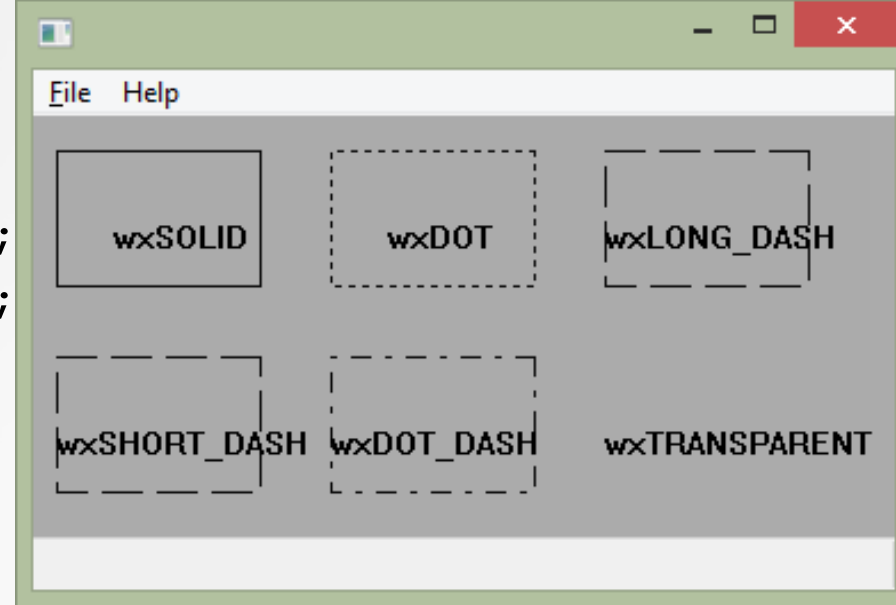
```
wxPaintDC dc(this);
```

```
wxColour col1, col2;  
col1.Set(wxT("#0c0c0c"));  
col2.Set(wxT("#000000"));  
wxBrush brush(  
    wxColour(255, 255, 255),  
    wxTRANSPARENT);  
dc.SetBrush(brush);
```

```
dc.SetPen(wxPen(col1, 1, wxSOLID));  
dc.DrawRectangle(10, 15, 90, 60);  
dc.DrawText(_T("wxSOLID"), 10+25, 15+30);
```

```
dc.SetPen(wxPen(col1, 1, wxDOT));  
dc.DrawRectangle(130, 15, 90, 60);  
dc.DrawText(_T("wxDOT"), 130+25, 15+30);
```

```
dc.SetPen(wxPen(col1, 1, wxLONG_DASH));  
dc.DrawRectangle(250, 15, 90, 60);  
dc.DrawText(_T("wxLONG_DASH"), 250, 15+30);
```



- Перо является также элементарным графическим объектом, отвечающим за закраску фигур

```
wxBrush (const wxColour &colour,  
wxBrushStyle style=wxBRUSHSTYLE_SOLID)
```

Конструктор имеет два параметра: цвет и стиль. Основные стили:

`wxBDIAGONAL_HATCH` -- Диагональная штриховка

`wxCROSSDIAG_HATCH` -- Диагональная крест-
накрест

`wxFDIAGONAL_HATCH,` -- Обратная диагональная

`wxCROSS_HATCH,` -- Крест-накрест прямая

`wxHORIZONTAL_HATCH,` -- Горизонтальными
линиями

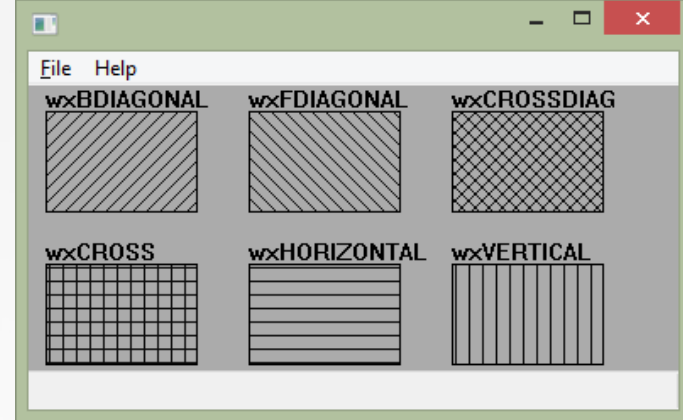
`wxVERTICAL_HATCH` -- Вертикальными линиями


```
wxPaintDC dc(this);  
wxColour coll;  
coll.Set(wxT( "#0c0c0c" ));
```

```
dc.SetBrush(wxBrush(coll,wxBDIAGONAL_HATCH));  
dc.DrawRectangle(10, 15, 90, 60);  
dc.DrawText(_( "wxBDIAGONAL" ), 10,0);
```

```
dc.SetBrush(wxBrush(coll,wxFDIAGONAL_HATCH));  
dc.DrawRectangle(130, 15, 90, 60);  
dc.DrawText(_( "wxFDIAGONAL" ), 130,0);
```

```
dc.SetBrush(wxBrush(coll,wxCROSSDIAG_HATCH));  
dc.DrawRectangle(250, 15, 90, 60);  
dc.DrawText(_( "wxCROSSDIAG" ), 250,0);
```



Масштабирование

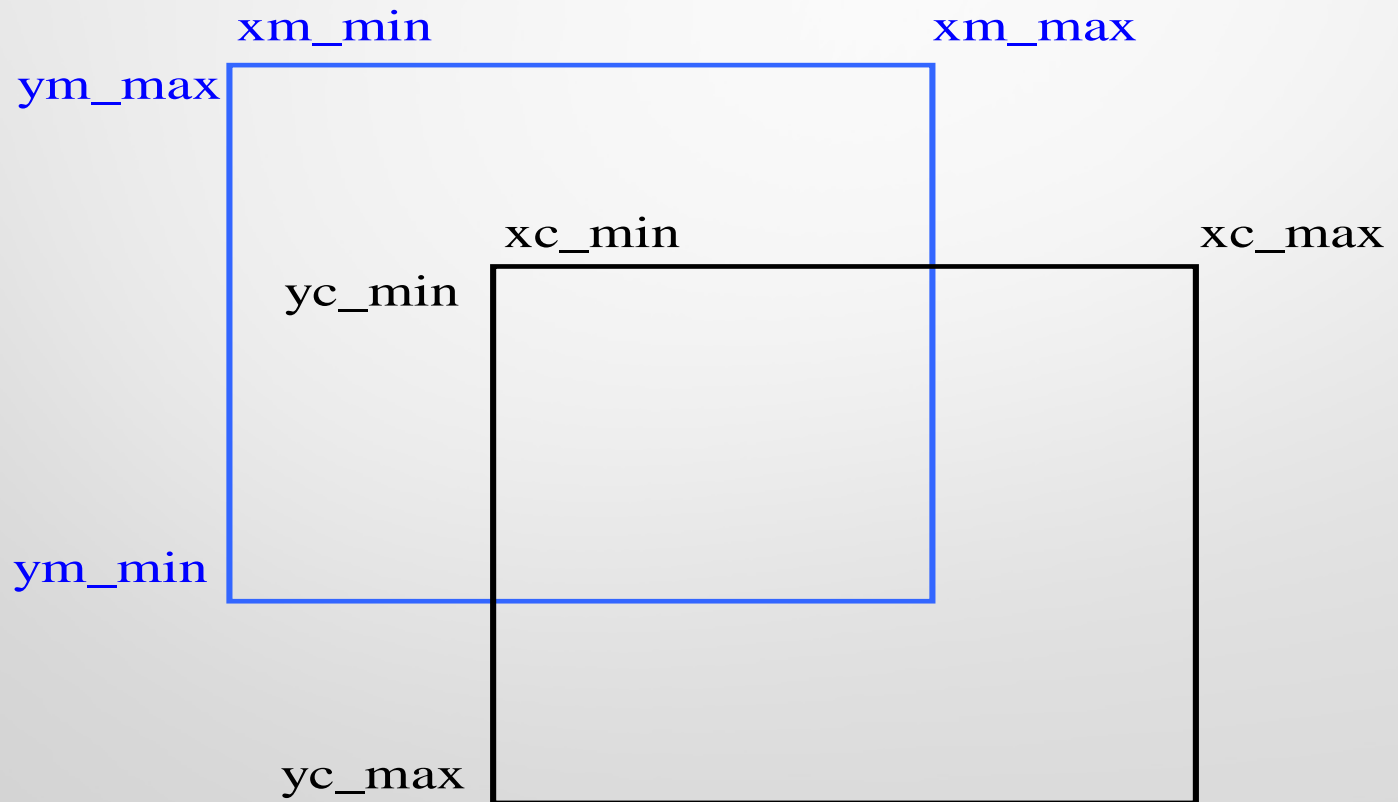
При выводе изображений довольно часто возникает проблема их корректного масштабирования, т.е. перевода из системы координат модели в систему координат «холста». Поскольку прямые линии при масштабировании должны оставаться прямыми, то оно осуществляется путем линейного преобразования координат:

$$x_c = A \cdot x_m + B$$

$$y_c = C \cdot y_m + D$$

Масштабирование

Коэффициенты масштабирования находятся исходя из известных преобразований для граничных точек.



Масштабирование

$$x_{c \min} = A \cdot x_{m \min} + B$$

$$x_{c \max} = A \cdot x_{m \max} + B$$

$$y_{c \min} = C \cdot y_{m \max} + D$$

$$y_{c \max} = C \cdot y_{m \min} + D$$

$$A = \frac{x_{c \max} - x_{c \min}}{x_{m \max} - x_{m \min}}$$

$$B = x_{c \min} - A \cdot x_{m \min}$$

$$C = \frac{y_{c \min} - y_{c \max}}{y_{m \max} - y_{m \min}}$$

$$D = y_{c \min} - C \cdot y_{m \max}$$