Лабораторная работа №8 (2-й семестр)

> Сначала учите науку программирования и всю теорию. Далее выработаете свой программистский стиль. Затем забудьте все и просто программируйте.
>
> — George Carrette

Теоретические сведения

Для рисования на форме и ее компонентах используются специально написанные для этого классы. В C++ Builder для этих целей используется свойсво Canvas («холст»), а в библиотеке wxWidgets классы: wxPaintDC, wxClientDC, wxWindowDC, wxScreenDC, wxMemoryDC или wxPrinterDC, для которых базовым абстрактным классом является wxDC. Основные методы класса wxDC представлены в табл. 1.

Табл. 1 — методы класса wxDC

Свойство	**Описание**
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.
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. (*x1,y1*) в т.(*x2,y2*)
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*
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.
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 цвет отличный от заданного. wxFLOOD_BORDER: область, которая будет закрашена указанным цветом.
wxBrush& GetBrush() void SetBrush(const wxBrush& brush)	Возвращает/устанавливает текущую кисть (см. табл. 2)
wxFont& GetFont() void SetFont(const wxFont& font)	Возвращает/устанавливает текущий шрифт
wxPen& GetPen() void SetPen(const wxPen& pen)	Возвращает/устанавливает текущее перо

WxBrush - обьект, который отвечает за стиль закраски всех фигур. Основные методы работы с классом приведены в табл. 2. Аналогичным образом за цвет линий отвечает обеъект wxPen, к оснвным свойстам которого можно отнести цвет и тощину линии.

Табл. 2 — Методы объекта wxBrush

Свойство	**Описание**
wxBrush(const wxColour& colour, int style = wxSOLID)	Констуктор. *Colour* – цвет, *style* – стиль заливки (см. табл. 3)
void SetColour(wxColour& colour) wxColour& GetColour()	Возвращает/устанавливает цвет заливки
void SetStyle(int style) int GetStyle()	Возвращает/устанавливает стиль заливки согласно табл. 3

Табл. 3 — Стили заливки класса wxBrush

Свойство	**Описание**
wxTRANSPARENT	Нет заливки
wxSOLID	Сплошная заливка
wxSTIPPLE	Использует точечный рисунок как пунктир
wxBDIAGONAL_HATCH	Обратная диагональная штриховка
wxCROSSDIAG_HATCH	Диагональная крест-накрест штриховка
wxFDIAGONAL_HATCH	Прямая диагональная штриховка
wxCROSS_HATCH	Крест-накрест
wxHORIZONTAL_HATCH	Горизонтальная штриховка
wxVERTICAL_HATCH	Вертикальная штриховка

Следует отметить, что по умолчанию, начало системы координат находится в левом верхнем углу. Ось абсцисс направлена вправо, а ось ординат — вниз.

Событие OnPaint. У большинства компонентов, присутствует событие OnPaint. Это событие вызывается, когда есть необходимость в перерисовке изображения на компоненте: при изменении размеров компонента; когда окно (форма) с компонентом перекрывается другими окнами, а затем вновь становится полностью видимым; и т. п. Соответственно, рисова-

ние в таких компонентах нужно производить при обработке этого события.

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:

$$\begin{aligned} x_c &= A \cdot x_m + B \\ y_c &= C \cdot y_m + D \end{aligned} \tag{1}$$

где: (x_m, y_m) — координаты точки с системе координат модели; (x_c, y_c) — координаты точки в системе координат «холста»; *A, B, C, D* — коэффициенты масштабирования.

Коэффициенты масштабирования могут быть найдены исходя из известных преобразований для граничных точек (рис. 1):

$$\begin{aligned} A &= (x_{cmax} - x_{cmin}) / (x_{mmax} - x_{mmin}) \\ D &= x_{cmin} - A \cdot x_{mmin} \\ C &= (y_{cmin} - x_{cmax}) / (y_{mmax} - y_{mmin}) \\ D &= y_{cmin} - C \cdot y_{mmax} \end{aligned} \tag{2}$$

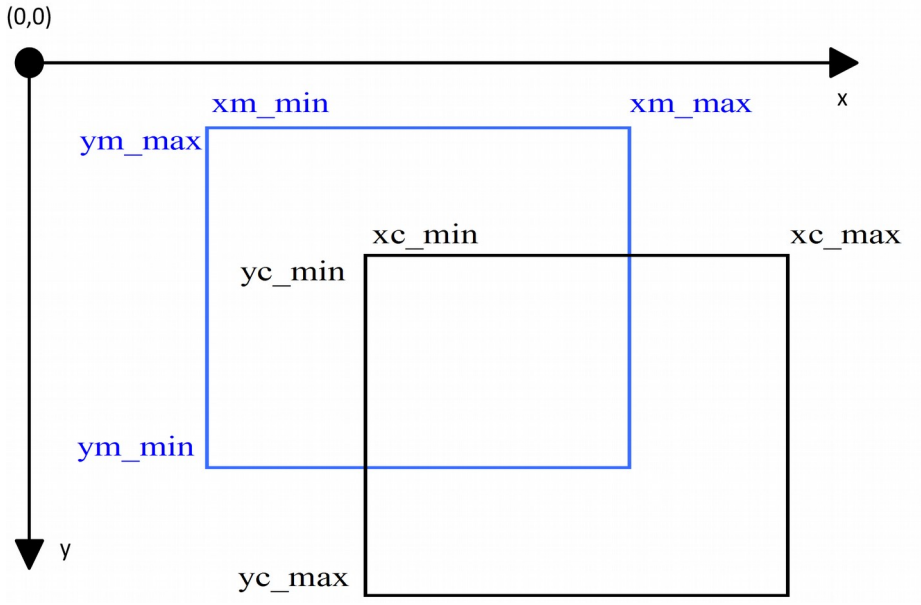

Рис. 1 — Схема взаимосвязи между системами координат модели и «холста»

Например, основной код приложения, рисующего треугольник на компоненте, так чтобы он всегда был вписан в прямоугольную область (10,10, width-10, height-10) на «холсте» компонента, выглядит так:

Файл: *Unit1.cpp*

```
...
#include <wx/dcclient.h>
...
const int xm_min=0, xm_max=1, ym_min=0, ym_max=1;

double a,b,c,d;

int _x(double x)
```

```
{
    return a*x + b;
}
int _y(double y)
{
    return c*y + d;
}
void yyyyFrame::OnPanel1Paint(wxPaintEvent& event)
{
    int h,w;
    wxPaintDC dc(Panel1);
    dc.GetSize(&w, &h);

    int xc_min = 10;
    int xc_max = w - 10;
    int yc_min = 10;
    int yc_max = h - 10;
    a = (xc_max - xc_min) / (xm_max - xm_min);
    b =  xc_min - a * xm_min;
    c = (yc_min - yc_max) / (ym_max - ym_min);
    d =  yc_min - c * ym_max;

    dc.SetPen(wxPen(*wxBLUE, 2));
    wxBrush brush(*wxRED, wxBDIAGONAL_HATCH  );
    brush.SetStyle(wxBDIAGONAL_HATCH);
    dc.SetBrush(brush);
    dc.DrawLine(_x(0),_y(0),_x(0.5),_y(1));
    dc.DrawLine(_x(0.5),_y(1),_x(1),_y(0));
    dc.DrawLine(_x(1),_y(0),_x(0),_y(0));

    dc.FloodFill(_x(0.5),_y(0.5),*wxBLUE, wxFLOOD_BORDER);

   }
```

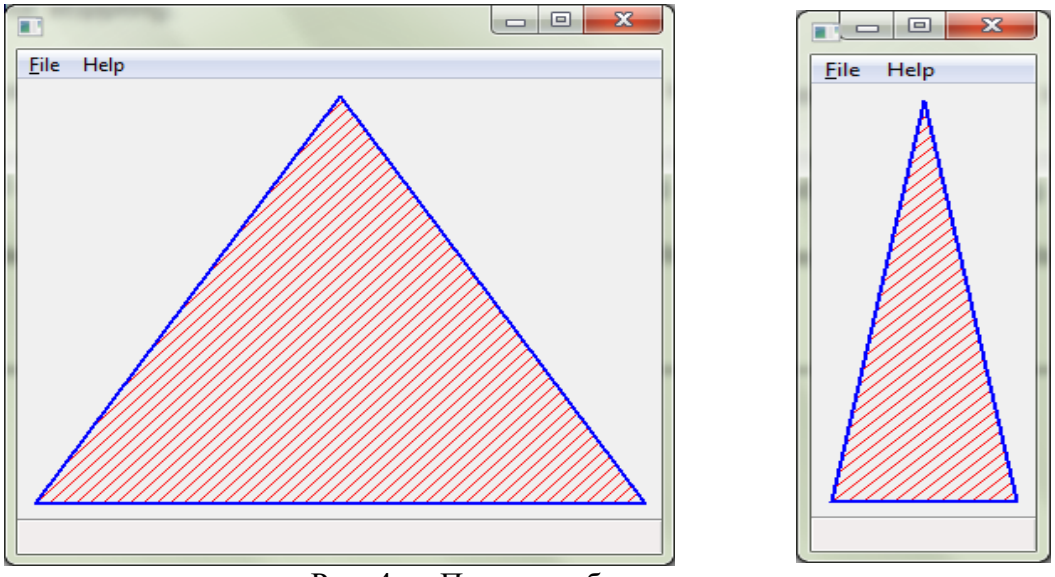

Рис. 4 — Пример работы программы