## Лабораторная работа 4

### *Инструменты рисования Builder C++*

### Подготовка к лабораторной работе

Для выполнения лабораторной работы необходимо ознакомиться (по литературе или в сети) и сделать краткое описание следующих функций и параметров, а также используемых событий.

1. Свойство Canvas компонентов TForm и TPaintBox.

Pen, Brush, Pixels

1. Инструменты рисования и их свойства
   * Кисти Brush
   * Карандаши
   * Шрифт
2. Функции рисования
3. Функции заливки
4. Массив для работы с отдельными пикселами
5. События

* OnResize и WM\_SIZE
* OnPaint и WM\_PAINT
* События мыши (рассмотренные в предыдущих работах).

Обработка событий **OnMouseDown**и **OnMouseUp**используется для операций, требуемых при нажатии и отпускании пользователем какой-либо кнопки мыши. Обработчики этих событий имеют параметры **Sender**, **Shift**, **Button**, **X** и**Y**. С первыми двумя параметрами мы уже знакомы из описания предыдущих событий. Значения параметра **Button**определяют, какая кнопка мыши нажата: **mbLeft**— левая, **mbRight**— правая, **mbMiddle**— средняя. Параметры **X** и **Y** определяют координаты указателя мыши в клиентской области компонента.

1. Объекты типа TPoint, TRect
2. Методы

* :: InvalidateRect
* BOOL InvalidateRect
* (
* HWND hWnd, // указатель на окно
* CONST RECT \*lpRect, // прямоугольник перерисовки
* BOOL bErase // режим перерисовки
* );
* Если первый параметр **NULL**, то будут перерисованы все окна. Если второй параметр **NULL**, то будет перерисована вся область окна. Последний параметр указывает на действия с фоном, если **TRUE**, то фон будет стерт, иначе оставлен. При удачном выполнении этой функции результат отличен от нуля.
* Указанные области перерисовки накапливаются, пока не будет получено сообщение **WM\_PAINT** или пока эта функция не будет отменена противоположной функцией **ValidateRect()**. Само сообщение **WM\_PAINT** посылается при необходимости изменить область, но только когда нет в очереди других сообщений. При установке нескольких областей производится их объединение и обработка одним **WM\_PAINT**.
* Refresh – обновление формы

## Задание

Программные приложения разрабатываются в среде Builder C++.

1. С помощью графических функций нарисовать на канве формы картинку с использованием всевозможных графических примитивов, карандашей, кистей и заливок.

void DrawPicture()

{

TCanvas \*canvas = Form1->Canvas;

TPen \*pen = canvas->Pen;

TBrush \*brush = canvas->Brush;

pen->Width=1;

//небо

brush->Color = clWebSkyBlue;

pen->Color = clWebSkyBlue;

canvas->Rectangle(10,10 + butOffset,600,170 + butOffset);

//трава

brush->Color = clWebLime;

pen->Color = clWebLime;

canvas->Rectangle(10,170 + butOffset,600,250 + butOffset);

// Дом

brush->Color = clWebPeru;

pen->Color = clBlack;

canvas->Rectangle(100,200 + butOffset,300,100 + butOffset);

TPoint p[3];

p[0] = Point(100,100 + butOffset);

p[1] = Point(300,100 + butOffset);

p[2] = Point(200,50 + butOffset);

canvas->Polygon(p,2);

//окна

brush->Color = clWebCornFlowerBlue;

canvas->Rectangle(130, 120 + butOffset, 170, 170 + butOffset);

canvas->Rectangle(210, 120 + butOffset, 250, 170 + butOffset);

brush->Color = clWebGoldenRod;

canvas->Rectangle(148, 120 + butOffset, 152, 170 + butOffset);

canvas->Rectangle(130, 136 + butOffset, 170, 140 + butOffset);

canvas->Rectangle(228, 120 + butOffset, 232, 170 + butOffset);

canvas->Rectangle(210, 136 + butOffset, 250, 140 + butOffset);

//солнце

brush->Color = clWebYellow;

pen->Color = clWebYellow;

canvas->Ellipse(500, 30 + butOffset, 550, 80 + butOffset);

pen->Width=3;

canvas->MoveTo(525, 30 - 4 + butOffset);

canvas->LineTo(525, 30 - 4 - 13 + butOffset);

canvas->MoveTo(525, 80 + 4 + butOffset);

canvas->LineTo(525, 80 + 4 + 13 + butOffset);

canvas->MoveTo(500 - 4, 55 + butOffset);

canvas->LineTo(500 - 4 - 13, 55 + butOffset);

canvas->MoveTo(550 + 4, 55 + butOffset);

canvas->LineTo(550 + 4 + 13, 55 + butOffset);

canvas->MoveTo(514, 44 + butOffset);

canvas->LineTo(514 - 13, 44 - 13 + butOffset);

canvas->MoveTo(539, 44 + butOffset);

canvas->LineTo(539 + 13, 43 - 13 + butOffset);

canvas->MoveTo(514, 69 + butOffset);

canvas->LineTo(514 - 13, 69 + 13 + butOffset);

canvas->MoveTo(539, 69 + butOffset);

canvas->LineTo(539 + 13, 69 + 13 + butOffset);

}

1. На компоненте PaintBox вывести график функции

F(x)=x/2\*exp (x/9+0,8) \* cos (x+1,2) на отрезке [-7, 7 ].

* Оси координат должны быть черными с подписями, стрелками и делениями;
* График должен быть цветным (красным или синим), толщина пера 3. Для рисования следует использовать функции MoveTo и LineTo.

void DrawGrafic()

{

int width = Form1->PaintBox1->Width;

int height = Form1->PaintBox1->Height;

int xdel = width / 14;

int ydel = height / 14;

TCanvas \*canvas = Form1->PaintBox1->Canvas;

TPen \*pen = canvas->Pen;

pen->Width=1;

// координатные линии Х и У

pen->Color = clBlack;

canvas->MoveTo(width / 2, 0);

canvas->LineTo(width / 2 ,height);

canvas->MoveTo(0, height / 2);

canvas->LineTo(width, height / 2);

// стрелка Х

canvas->MoveTo(width, height / 2);

canvas->LineTo(width - 5, height / 2 - 3);

canvas->MoveTo(width, height / 2);

canvas->LineTo(width - 5, height / 2 + 3);

// подпись Х

canvas->MoveTo(width - 2, height / 2 + 5);

canvas->LineTo(width - 6, height / 2 + 12);

canvas->MoveTo(width - 6, height / 2 + 5);

canvas->LineTo(width - 2, height / 2 + 12);

// стрелка Y

canvas->MoveTo(width / 2, 0);

canvas->LineTo(width / 2 - 3, 0 + 5);

canvas->MoveTo(width / 2, 0);

canvas->LineTo(width / 2 + 3, 0 + 5);

// подпись Y

canvas->MoveTo(width / 2 - 5, 0 + 2);

canvas->LineTo(width / 2 - 9, 0 + 12);

canvas->MoveTo(width / 2 - 9, 0 + 2);

canvas->LineTo(width / 2 - 7, 0 + 7);

// штрихи

for (int x=xdel; x<width; x+=xdel)

{

canvas->MoveTo(x, height / 2 + 3);

canvas->LineTo(x, height / 2 - 3);

}

for (int y=ydel; y<height; y+=ydel)

{

canvas->MoveTo(width / 2 + 3, y);

canvas->LineTo(width / 2 - 3, y);

}

double x0 = -7.0;

double y0 = 0.0;

double dx = 0.25;

double xMax = 7.0;

pen->Color = clBlue;

pen->Width=3;

// график

canvas->MoveTo(x0 \* xdel, height / 2 - func(x0) \* ydel);

for (double x = x0 + dx; x <= xMax; x = x + dx) {

canvas->LineTo(width / 2 + x \* xdel, height / 2 - func(x) \* ydel);

}

}

1. Работа с областью отсечения (используем методы :: InvalidateRect, Refresh и события мыши)
   1. По щелчку на кнопке рисуем круг и заставляем его исчезнуть, если щелкаем мышью внутри круга.
   2. Круг рисуется по событию OnPaint. С помощью мыши круг перемещаем по поверхности формы.

void DrawCircle(int X0, int Y0)

{

Form1->Canvas->Pen->Width = 1;

Form1->Canvas->Brush->Color = clBlue;

Form1->Canvas->Pen->Color = clBlue;

circle.Left = X0;

circle.Top = Y0;

circle.Right = X0 + 200;

circle.Bottom = Y0 + 200;

Form1->Canvas->Ellipse(circle.Left, circle.Top, circle.Right, circle.Bottom);

}

void \_\_fastcall TForm1::FormMouseDown(TObject \*Sender, TMouseButton Button, TShiftState Shift,

int X, int Y)

{

if (X > circle.Left && X < circle.Right && Y > circle.Top && Y < circle.Bottom ) {

if(Button == mbLeft)

f4 = true;

else if (f3 && Button == mbRight) {

f3 = false;

::InvalidateRect(Form1->Handle, &circle, true);

Form1->Canvas->Refresh();

}

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormMouseUp(TObject \*Sender, TMouseButton Button, TShiftState Shift,

int X, int Y)

{

if(Button == mbLeft)

f4 = false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormMouseMove(TObject \*Sender, TShiftState Shift, int X, int Y)

{

if (f4) {

::InvalidateRect(Form1->Handle, &circle, true); // удалить

Form1->Canvas->Refresh();

circleX = X - 100;

circleY = Y - 100;

Form1->FormPaint(Sender);

}

}

//---------------------------------------------------------------------------