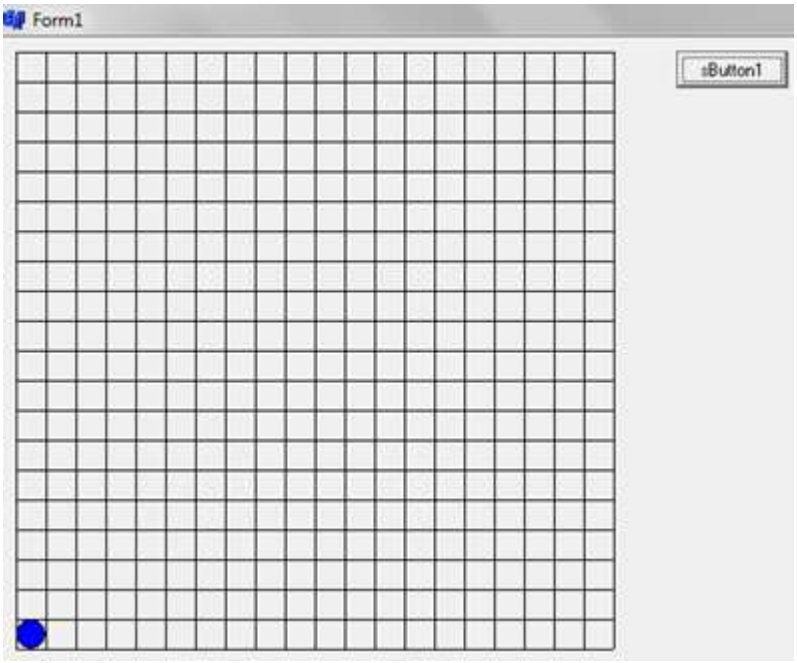
Лабораторная работа 1

Цель работы:

Выбрать среду программирования. Это может быть Microsoft .NET или C++Builder.
Освоить графические примитивы для рисования в выбранной системе программирования.

Результаты работы:

1. В результате работы нужно построить сетку. В программе изменять число квадратов сетки и их размер.
2. Рисование сетки в подпрограмме. Вывод точки – отдельная подпрограмма.
3. Вывести точку. Это может быть круг или весь заполненный квадрат. Координаты (0,0) нижний левый угол.



1. Теоретическая часть:

Microsoft .NET

Программирование графики в *Microsoft .NET* возможно в консольном режиме. Именно в нем необходимо выполнять лабораторные работы этого курса.

Пример работы с графикой в консольном режиме *Microsoft .NET:*

```
// line.cpp: определяет точку входа для консольного приложения.
//

#include "stdafx.h"
//#include <iostream>

struct Point
{
	int x;
	int y;
};

void DrawGrid(HDC *hdc);

void main()
{
	POINT p;
	HWND hWnd = GetConsoleWindow();                 // получаем идентификатор окна
	HDC hDC = GetDC(hWnd);                          // получаем контекст отображения

	HPEN whitePen = GetStockPen(WHITE_PEN);         //создаем перо для рисование сетки

	HBRUSH blackBrush = GetStockBrush(BLACK_BRUSH);       //создаем кисти
	HBRUSH greenBrush = CreateSolidBrush(RGB(0, 255, 0));
	HBRUSH blueBrush = CreateSolidBrush(RGB(0, 191, 255));



	SelectBrush(hdc, blackBrush);                   //обновляем окно (черный цвет)
	FloodFill(hdc, 0, 0, RGB(0, 0, 1));

	MoveToEx(hdc, 0, 0, NULL);                     // рисуем линию
	LineTo(hdc, 100, 100);


	HBRUSH brush = GetStockBrush(WHITE_BRUSH);  //создание кисти для зарисовки квадрата
	SelectBrush(hdc, brush);
	Rectangle(hdc, 110, 110, 120, 120);         // Рисуем прямоугольник
	Ellipse(hdc, 200, 200, 100, 50);            // Рисуем эллипс


	SelectPen(hdc, whitePen);                   //рисуем сетку
	DrawGrid(&hdc);

	//DeleteObject(whitePen);
	ReleaseDC(hwnd, hdc);

	system("pause");

}
```

```
void DrawGrid(HDC *hdc)
{
	int  x, y, x1, y1;
	//устанавливаем начальное положение пера
	x = y = x1 = y1 = X0;
	//рисование по y
	for (int i = 0; i <= (data.height); i++)
	{
	  MoveToEx(*hdc, x, y, NULL);
	  LineTo(*hdc, x + (data.width*data.size), y);
	  y += (data.size);
	}
	//рисование по x
	for (int j = 0; j <= data.width; j++)
	{
	  MoveToEx(*hdc, x1, y1, NULL);
	  LineTo(*hdc, x1, y1 + (data.height*data.size));
	  x1 += (data.size);
	}
}
```

C++Builder

В *C++Builder* возможны работа в консольном режиме только в ранних версиях.

Нам доступна работа с графикой в оконных приложениях. Для этого необходимо создать какой-либо графический примитив. (Например, кнопку). И привязать обработчик событий к нашей программе. Для этого после создания кнопки на ней щелкнуть мышью два раза.

Основные понятия

В среде C++Builder существует три рода объектов, которые имеют отношение к графике:

- Канва - предоставляет битовую карту поверхности окна приложения, компоненты, принтера и т.п., которая может быть использована для вывода графики. Канва не самостоятельный объект, она всегда является свойством какого-то другого графического объекта.
  - Графика - представляет растровое изображение некоторого файла или ресурса (битового образа, пиктограммы или метафайла).

    C++Builder определяет производные от базового класса TGraphic объектные классы:

    - TBitmap,
    - Ticon,
    - TMetafile.
- Рисунок (TPicture) представляет собой контейнер для графики, который может содержать любые классы графических объектов. Таким образом, контейнерный класс TPicture может содержать битовый образ, пиктограмму, метафайл или некоторый другой графический тип, определенный пользователем, а приложение может стандартно обращаться ко всем объектам контейнера посредством объекта TPicture.

Объектный класс канвы

Инкапсулированные и перегруженные функции GDI и WinApi объектного класса канвы многие авторы относят к трем различным уровням. В этой условной классификации функции высокого уровня обеспечивают возможность рисования линий, фигур и текста. Определение свойств и методов манипулирования графическими примитивами канвы отнесены к среднему уровню. Нижний уровень обеспечивается доступ к самим функциям Windows GDI. Классификация не бесспорна, но она позволяет ориентироваться в достаточно большом колличестве свойств и методов канвы и, поэтому, приведем эту классификацию.

Уровень	Метод (Функция)	Свойства	Действие
Высокий	**MoveTo**	**PenPos**	**Определяет текущую позицию пера**
	LineTo	**PenPos**	**Рисует прямую до заданной точки**
	Rectangle		**Рисует прямоугольник**
	Ellipse		**Рисует эллипс**
	Arc		**Рисует дугу**
	Polyline		**Рисует ломаную линию**

	PolyBezier		**Рисует кривую Блейзера**
	Chord		**Рисует сектор**
	DrawFocusRect		**Рисует прямоугольник**
	FrameRect		**Выводит рамку вокруг прямоугольника**
	Pie		**Выводит сектор круга**
	TextOut		**Выводит текстовую строку**
	TextHeight		**Задает высоту текстовой строки**
	TextWidth		**Задает ширину для вывода текстовой строки**
	TextRect		**Вывод текста внутри прямоугольника**
	FillRect		**Заливка указанного прямоугольника цветом и текстурой текущей кисти**
	FloodFill		**Заливка области канвы (произвольной формы) заданным цветом**
Средний		**Pen**	**Используется для установки цвета, стиля, ширины и режима пера**
		Brush	**Используется для установки цвета и текстуры при заливке графических фигур и фона канвы.**
		Font	**Используется для установки шрифта заданного цвета, размера и стиля**
		Pixels	**Используется для чтение и записи цвета заданного пикселя канвы**
	CopyRect	**CopyMode**	**Копирует прямоугольную область канвы в режиме CopyMode**
	BrushCopy		**Копирует прямоугольную область канвы с заменой цвета**
	Draw		**Рисует битовый образ, пиктограмму, метафайл в заданном месте канвы**
	StretchDraw		**Рисует битовый образ, пиктограмму или метафайл так, чтобы целиком заполнить заданный прямоугольник**
Низкий		**Handle**	**Используется как параметр при вызове функций Windows GDI**

Пример программы:

```
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "sButton"
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
const int x0=10; const int y0=10; int a=20; int b=20;
void lines()
{
    int i, j, x=x0, y=y0, p=x0, q=y0;
    for (i=0; i<=b; i++)
     {
         Form1->Canvas->MoveTo(x,y);
         Form1->Canvas->LineTo(x,y+a*b);
         x=x+a;
      }
  for (j=0; j<=b; j++)
     {
       Form1->Canvas->MoveTo(p,q);
       Form1->Canvas->LineTo(p+a*b,q);
       q=q+a;
     }

}
int r; int l;
void krug(int x,int y)
   {
       Form1->Canvas->Brush->Color=clBlue;
       if (l<0||r<0||l>=b||r>=b)
       {return;}
        Form1->Canvas->Ellipse(x0+l*a, x0+(b-1)*a-r*a, x0+a+l*a, x0+b*a-r*a);
   }
void __fastcall TForm1::sButton1Click(TObject *Sender)
   {
       lines();
       krug(0,0);
   }
```