Створення графічних примітивів

Delphi дозволяє програмісту розробляти програми, які можуть виводити графіку: схеми, креслення, ілюстрації.

Програма виводить графіку на поверхню об'єкту (форми або компоненту *Image*). Поверхні об'єкту відповідає властивість *Canvas*. Для того, щоб вивести на поверхню об'єкту графічний елемент (пряму лінію, коло, прямокутник і т. д.), необхідно застосувати до властивості *Canvas* цього об'єкту відповідний метод. Наприклад, інструкція *Form1.Canvas.Rectangle* (10,10,100,100) викреслює у вікні програми прямокутник.

Полотно

Як було сказано раніше, поверхні, на яку програма може виводити графіку, відповідає властивість *Canvas*. У свою чергу, властивість *Canvas* - це об'єкт типу *TCanvas*. Методи цього типу забезпечують виведення графічних примітивів (точок, ліній, кіл, прямокутників і т. д.), а властивості дозволяють задати характеристики графічних примітивів, що виводяться: колір, товщину і стиль ліній; колір і вид заповнення областей; характеристики шрифту при виведенні текстової інформації.

Методи виведення графічних примітивів розглядають властивість *Canvas* як деяке абстрактне полотно, на якому вони можуть малювати (*canvas* переводиться як "поверхня", "полотно для малювання"). Полотно складається з окремих точок - пікселів. Розташування пікселя характеризується його горизонтальною (X) і вертикальною (Y) координатами. Лівий верхній піксель має координати (0, 0). Координати зростають зверху вниз і зліва направо (рис. 24). Значення координат правої нижньої точки полотна залежать від розміру полотна.

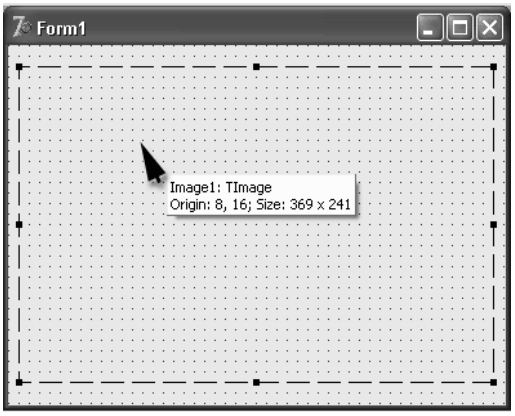

рис. 24. Використання компоненту ***Canvas***

Розмір полотна можна одержати, звернувшись до властивостей *Height* і *Width* області ілюстрації (*Image*) або до властивостей форми*: ClientHeight і Clientwidth*.

Олівець і кисть

Художник в своїй роботі використовує олівці і кисті. Методи, що забезпечують викреслювання на поверхні полотна графічних примітивів, теж використовують олівець і кисть. Олівець застосовується для викреслювання ліній і контурів, а кисть - для зафарбування областей, обмежених контурами.

Олівцю і кисті, використовуваним для виведення графіки на полотні, відповідають властивості *Pen* (олівець) і *Brush* (кисть). Значення властивостей цих об'єктів визначають вид графічних елементів, що виводяться.

Олівець

Олівець використовується для викреслювання крапок, ліній, контурів геометричних фігур: прямокутників, кіл, еліпсів, дуг і ін. Вид лінії, яку залишає олівець на поверхні полотна, визначають властивості об'єкту *Pen*, які перераховані табл.

Таблиця. Властивості об'єкту *Pen* (олівець)

Властивість	**Визначає**
Color	Колір лінії
Width	Товщину лінії
Style	Вид лінії
Mode	Режим відображення

Властивість *Color* задає колір лінії, що викреслюється олівцем. У табл. перераховані іменовані константи (тип *TColor*), які можна використовувати як значення властивості *Color*.

Таблиця. Значення властивості *Colo*r визначає колір лінії

Колір	**Константа**	**Колір**	**Константа**
Чорний	*clBlack*	Сірий	*clGray*
Сріблястий	*clSilver*	Білий	*clWhite*
Червоний	*clRed*	Каштановий	*clMaroon*
Салатний	*clLime*	Зелений	*clGreen*
Синій	*clBlue*	Оливковий	*clOlive*
Яскраво-рожевий	*clFuchsia*	Темно-синій	*clNavy*
Бірюзовий	*clAqua*	Зелено-блакитний	*clTeal*

Властивість *Width* задає товщину лінії (у пікселях). Наприклад, інструкція *Canvas.Pen.Width: =3* встановлює товщину лінії в 3 пікселі.

Властивість *Style* визначає вид (стиль) лінії, яка може бути безперервною або пунктирною різної довжини. У табл. перераховані іменовані константи, що дозволяють задати стиль лінії. Товщина пунктирної лінії не може бути більше 1. Якщо значення властивості *Pen.Width* більше одиниці, то пунктирна лінія буде виведена як суцільна.

Таблиця. Значення властивості *Pen. style* визначає вид лінії

Константа	**Вид лінії**
psSolid	Суцільна лінія
psDash	Пунктирна лінія, довгі штрихи
psDot	Пунктирна лінія, короткі штрихи
psDashDot	Пунктирна лінія, чергування довгого і короткого штрихів
psDashDotDot	Пунктирна лінія, чергування одного довгого і двох коротких штрихів
psClear	Лінія не відображається (використовується, якщо не треба зображати межу області, наприклад, прямокутника).

Властивість *Mode* визначає, як формуватиметься колір точок лінії залежно від кольору точок полотна, через які ця лінія прокреслюється. За замовчуванням вся лінія викреслюється кольором, визначуваним значенням властивості *Pen.Color*.

Проте програміст може задати інверсний колір лінії по відношенню до кольору фону. Це гарантує, що незалежно від кольору фону всі ділянки лінії будуть видно, навіть в тому випадку, якщо колір лінії і колір фону співпадають.

Таблиця. Значення властивості *Pen.Mode* впливає на колір лінії

Константа	**Колір лінії**

pmBlack	Чорний, не залежить від значення властивості *Pen.Color*
pmWhite	Білий, не залежить від значення властивості *Pen.Color*
pmCopy	Колір лінії визначається значенням властивості *Pen.Color*
pmNotCopy	Колір лінії є інверсним по відношенню до значення властивості *Pen.Color*
pmNot	Колір точки лінії визначається як інверсний по відношенню до кольору точки полотна, в яку виводиться точка лінії

Кисть

Кисть *(Canvas.Brush*) використовується методами, що забезпечують викреслювання замкнутих областей, наприклад геометричних фігур, для заливки цих областей. Кисть має дві властивості, які перераховані в табл.

Таблиця. Властивості об'єкту *TBrush* (кисть)

Властивість	**Визначає**
Color	Колір зафарбування замкнутої області
Style	Стиль (тип) заповнення області

Область усередині контура може бути зафарбована або заштрихована. У першому випадку область повністю перекриває фон, а в другому - крізь незаштриховані ділянки області буде видно фон.

Як значення властивості *Color* можна використовувати будь-яку з констант типу *TColor*.

Таблиця. Значення властивості *Brush. Style* визначають тип зафарбування

Константа	**Тип заповнення (заливки) області**
bsSolid	Суцільна заливка
bsClear	Область не зафарбовується
bsHorizontal	Горизонтальне штрихування
bsVertical	Вертикальне штрихування
bsFDiagonal	Діагональне штрихування з нахилом ліній вперед
bsBDiagonal	Діагональне штрихування з нахилом ліній назад
bsCross	Горизонтально-вертикальне штрихування, в клітинку
bsDiagCross	Діагональне штрихування, в клітинку

Виведення тексту

Для виведення тексту на поверхню графічного об'єкту використовується метод *TextOut*. Інструкція виклику методу *TextOut* в загальному вигляді виглядає таким чином:

Об'єкт.Canvas.TextOut(x, y, Текст),

де: об'єкт - ім'я об'єкту, на поверхню якого виводиться текст; x, y - координати точки графічної поверхні, від якої виконується виведення тексту (рис. 25); Текст - змінна або константа символьного типу, значення якої визначає текст, що виводиться методом.

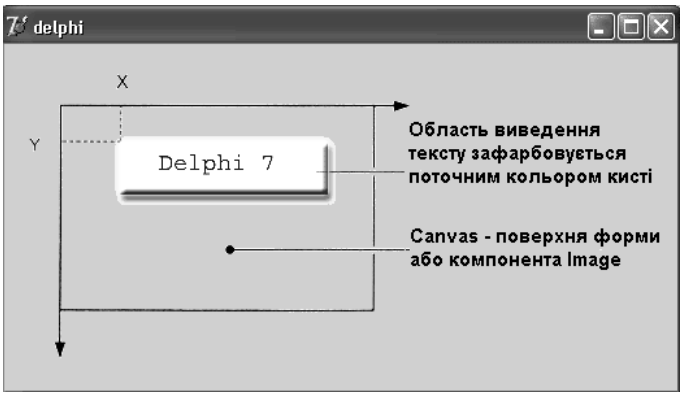

рис. 25. Використання компоненту ***Canvas***

Шрифт, який використовується для виведення тексту, визначається значенням властивості *Font* відповідного об'єкту *Canvas*. Властивість *Font* є об'єктом типу *TFont*. У таблиці перераховані властивості об'єкту *TFont*, що дозволяють задати характеристики шрифту, використовуваного методами *TextOut* і *TextRect* для виведення тексту.

Таблиця. Властивості об'єкту *TFont*

Властивість	Визначає
Name	Використовуваний шрифт. Як значення слід використовувати назву шрифту, наприклад Arial
Size	Розмір шрифту в пунктах (points). Пункт - це одиниця вимірювання розміру шрифту, використовувана в поліграфії. Один пункт рівний 1/72 дюйма
Style	Стиль зображення символів: нормальний, напівжирний, курсив, підкреслений, перекреслений. Стиль задається за допомогою наступних констант: *fsBold* (напівжирний), *fsItalic* (курсив), *fsUnderline* (підкреслений), *fsStrikeOut* (перекреслений).
Style	Властивість *style* є множиною, що дозволяє комбінувати необхідні стилі. Наприклад, інструкція програми, що встановлює стиль "напівжирний курсив", виглядає так: *Объект.Canvas.Font*: *= [fsBold, fs Italic]*
Color	Колір символів. Як значення можна використовувати константу типу *TColor*

Область виведення тексту зафарбовується поточним кольором кисті. Тому перед виведенням тексту властивості *Brush.Color* потрібно надати значення *bsClear* або задати колір кисті, який співпадає з кольором поверхні, на яку виводиться текст.

Наступний фрагмент програми демонструє використання функції *TextOut* для виведення тексту на поверхню форми:

```
with Form1.Canvas do begin // встановити характеристики шрифту
Font.Name := 'Tahoma';
Font.Size := 20;
Font.Style := [fsItalic, fsBold] ;
Brush.Style := bsClear; // область виведення тексту не зафарбовується
TextOut(0, 10, 'Borland Delphi 7');
end;
```

Після виведення тексту методом *TextOut* покажчик виведення (олівець) переміщається в правий верхній кут області виведення тексту.

Іноді потрібно вивести який-небудь текст після повідомлення, довжина якого під час розробки програми невідома. Наприклад, це може бути слово "грн." після значення числа, записаного прописом. В цьому випадку необхідно знати координати правої межі вже виведеного тексту. Координати правої межі тексту, виведеного методом *TextOut*, можна одержати, звернувшись до властивості *PenPos.*

Наступний фрагмент програми демонструє можливість виведення рядка тексту за допомогою двох інструкцій *TextOut*:

```
with Form1.Canvas do begin
TextOut(0, 10, 'Borland');
TextOut(PenPos.X, PenPos.Y, 'Delphi 7');
end;
```

Методи викреслювання графічних примітивів

Будь-яка картинка, креслення, схема можуть розглядатися як сукупність графічних примітивів: точок, ліній, кіл, дуг і ін. Таким чином, для того, щоб на екрані з'явилася потрібна картинка, програма повинна забезпечити викреслювання (виведення) графічних примітивів, з яких складається дана картинка.

Викреслювання графічних примітивів на поверхні компоненту (форми або області виведення ілюстрації) здійснюється застосуванням відповідних методів до властивості *Canvas* цього компоненту.

Лінія

Викреслювання прямої лінії здійснює метод *LineTo*, інструкція виклику якого в загальному вигляді виглядає таким чином:

Компонент.Canvas.LineTo(x,y)

Метод *LineTo* викреслює пряму лінію від поточної позиції олівця в точку з координатами, вказаними при виклику методу.

Початкову точку лінії можна задати, перемістивши олівець в потрібну точку графічної поверхні. Зробити це можна за допомогою методу *MoveTo*, вказавши як параметри координати нового положення олівця.

Вид лінії (колір, товщина і стиль) визначається значеннями властивостей об'єкту *Pen* графічної поверхні, на якій викреслюється лінія.

Ламана лінія

Метод *Polyline* викреслює ламану лінію. Як параметр метод одержує масив типу *TPoint*. Кожен елемент масиву є записом, поля x і y якої містять координати точки перегину ламаної. Метод *Polyline* викреслює ламану лінію, послідовно сполучаючи прямими точки, координати яких знаходяться в масиві: першу з другою, другу з третьою, третю з четвертою і т.д.

Як приклад використання методу *Polyline* в лістингу приведена процедура, яка виводить графік зміни деякої величини. Передбачається, що початкові дані знаходяться в доступному процедурі масиві Data (тип Integer).

Лістинг. Графік функції (використання методу *Polyline*)

```
procedure TForml.Button1Click(Sender: TObject);
var
gr: array[1..50] of TPoint; // графік — ламана лінія
x0,y0: integer; // координати точки початку координат
dx,dy: integer; // крок координатної сітки по осях X і Y
```

```
i: integer;
begin
x0 := 10; y0 := 200; dx :=5; dy := 5;
// заповнимо масив gr
for i:=1 to 50 do begin
gr[i].x := x0 + (i-1)*dx;
gr[i].y := y0 - Data[i]*dy;
end;
// будуємо графік
with form1.Canvas do begin
MoveTo(x0,y0); LineTo(x0,10); // вісь Y
MoveTo(x0,y0); LineTo(200,y0); // вісь X
Polyline(gr); // графік
end; end;
```

Коло і еліпс

Метод *Ellipse* викреслює еліпс або коло, залежно від значень параметрів. Інструкція виклику методу в загальному вигляді виглядає таким чином:

Об'єкт.Canvas.Ellipse(x1,y1, x2,y2),

де: об'єкт - ім'я об'єкту (компоненту), на поверхні якого виконується викреслювання; x1, y1, x2, y2 - координати прямокутника, усередині якого викреслюється еліпс або, якщо прямокутник є квадратом, коло (рис.26).

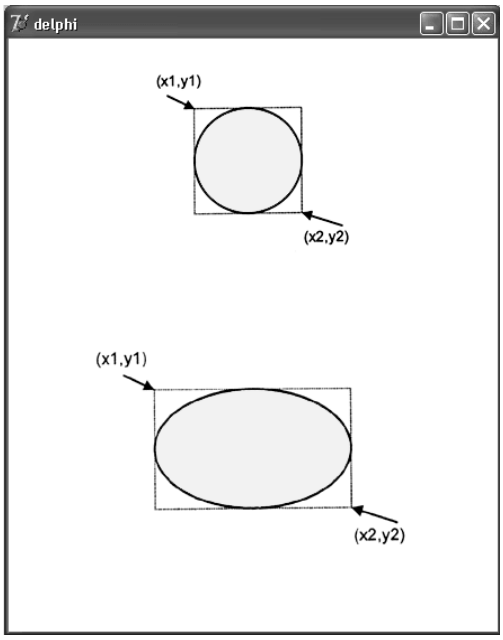

рис. 26. Значення параметрів методу *Ellipse* визначають вид геометричної фігури

Колір, товщина і стиль лінії еліпса визначаються значеннями властивості *Pen*, а колір і стиль заливки області усередині еліпса - значеннями властивості *Brush* поверхні (*Canvas*), на яку виконується виведення.

Дуга

Викреслювання дуги виконує метод *Arc*, інструкція виклику якого в загальному вигляді виглядає таким чином:

Об'єкт.Canvas.Arc(x1,y1,x2,y2,x3,y3,x4,y4),

де: x1, y1, x2, y2 - параметри, що визначають еліпс (коло), частиною якого є викреслювана дуга; x3, y3 - параметри, що визначають початкову точку дуги; x4, y4 - параметри, що визначають кінцеву точку дуги.

Початкова (кінцева) точка - це точка перетину межі еліпса і прямої, проведеної з центру еліпса в точку з координатами x3 і y3 (x4, y4). Дуга викреслюється проти годинникової стрілки від початкової точки до кінцевої (рис. 27).

Колір, товщина і стиль лінії, якою викреслюється дуга, визначаються значеннями властивості *Pen* поверхні (canvas), на яку виконується виведення.

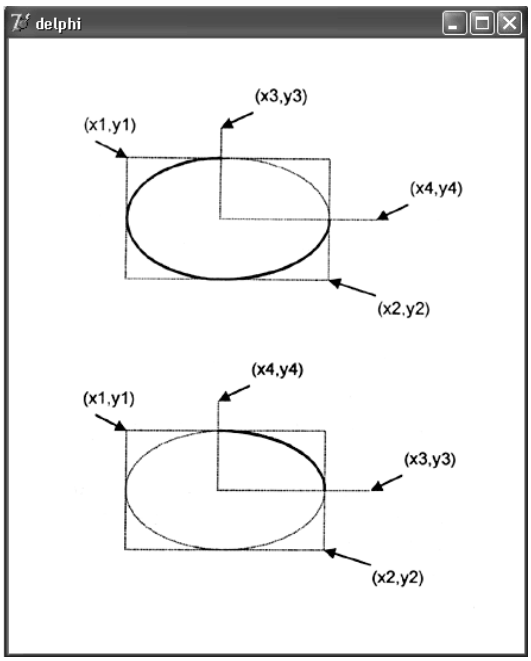

рис. 27. Значення параметрів методу *Arc* визначають дугу як частину еліпса (кола)

Прямокутник

Прямокутник викреслюється методом *Rectangle*, інструкція виклику якого в загальному вигляді виглядає таким чином:

Об'єкт.Canvas.Rectangle(x1, y1,x2, y2),

де: об'єкт - ім'я об'єкту (компоненту), на поверхні якого виконується викреслювання; x1, y1 і x2, y2 - координати лівого верхнього і правого нижнього кутів прямокутника.

Метод *RoundRec* теж викреслює прямокутник, але з заокругленими кутами. Інструкція виклику методу *RoundRec* виглядає так:

Об'єкт.Canvas.RoundRec(x1,y1,x2, y2, x3, y3),

де: x1, y1, x2, y2 - параметри, що визначають положення кутів прямокутника, в який вписується прямокутник з заокругленими кутами; x3 і y3 - розмір еліпса, одна чверть якого використовується для викреслювання заокругленого кута (рис. 28).

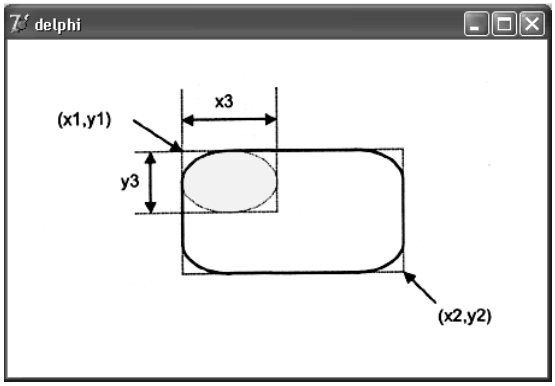

рис. 28. Метод *RoundRec* викреслює прямокутник з заокругленими кутами

Вид лінії контура (колір, ширина і стиль) визначається значеннями властивості *Pen*, колір і стиль заливки області усередині прямокутника - значеннями властивості *Brush* поверхні (*Canvas*), на якій прямокутник викреслюється.

Є ще два методи, які викреслюють прямокутник, використовуючи як інструмент тільки кисть (*Brush*). Метод *FillRect* викреслює зафарбований прямокутник, а метод *FrameRect* - тільки контур. У кожного з цих методів лише один параметр - структура типу *TRect*. Поля структури *TRect* містять координати прямокутної області, вони можуть бути заповнені за допомогою функції *Rect*.

Нижче як приклад використання методів *FillRect* і *FrameRect* приведена процедура, яка на поверхні форми викреслює прямокутник з червоною заливкою і прямокутник із зеленим контуром.

```
procedure TForm1.Button1Click(Sender: TObject);
var
r1, r2: TRect; // координати кутів прямокутників
begin
// задамо координати кутів прямокутників
r1 := Rect(20,20,60,40); r2 := Rect(10,10,40,50);
with form1.Canvas do begin
Brush.Color := clRed;
FillRect(r1); // зафарбований прямокутник
Brush.Color := clGreen;
FrameRect(r2}; // тільки межа прямокутника
end;
end;
```

Багатокутник

Метод *Polygon* викреслює багатокутник. Як параметр метод одержує масив типу *TPoint*. Кожен елемент масиву є записом, поля (x,y) якого містять координати однієї вершини багатокутника. Метод *Polygon* викреслює багатокутник, послідовно сполучаючи прямими лініями точки, координати яких знаходяться в масиві: першу з другою, другу з третьою, третю з четвертою і т.д. Потім з'єднуються остання і перша крапки.

Колір і стиль межі багатокутника визначаються значеннями властивості *Pen*, а колір і стиль заливки області, обмеженою лінією межі, - значеннями властивості *Brush*, причому область зафарбовується з використанням поточного кольору і стилю кисті.

Нижче приведена процедура, яка, використовуючи метод *Polygon*, викреслює трикутник:

```
procedure TForm1.Button2Click(Sender: TObject);
var
pol: array[1..3] of TPoint; // координати точок трикутника
begin
pol[1].x := 10;
polf1].y := 50;
pol[2].x := 40;
pol[2].y := 10;
pol[3].x := 70;
pol[3].y := 50;
Form1.Canvas.Polygon(pol);
end;
```

Сектор

Метод *Pie* викреслює сектор еліпса або круга. Інструкція виклику методу в загальному вигляді виглядає таким чином:

Об'єкт. Canvas.Pie(x1,y1,x2,y2,x3,y3,x4,y4),

де: x1, y1, x2, y2 - параметри, що визначають еліпс (коло), частиною якого є сектор; x3, y3, x4, y4 - параметри, що визначають координати кінцевих точок прямих, що є межами сектора.

Початкові точки прямих співпадають з центром еліпса (кола). Сектор вирізується проти годинникової стрілки від прямої, заданою точкою з координатами (x3, y3), до прямої, заданою точкою з координатами (x4, y4) (рис. 29).

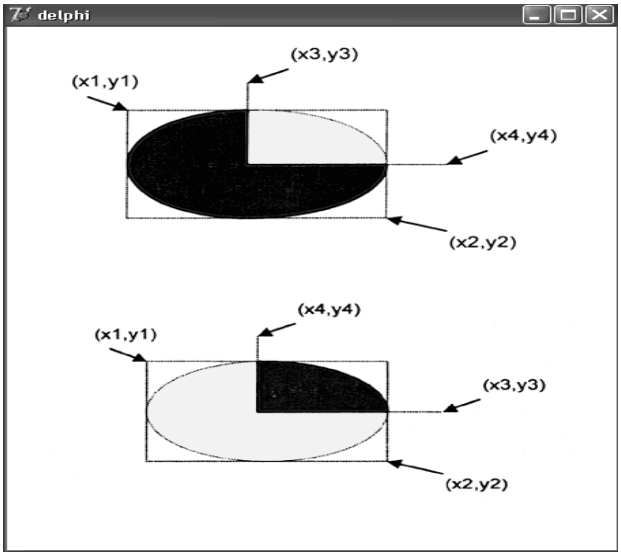

рис. 29. Значення параметрів методу *Pie* визначають сектор як частину еліпса (кола)

Точка

Поверхні, на яку програма може здійснювати виведення графіки, відповідає об'єкт *Canvas*. Властивість *Pixels*, що є двовимірним масивом типу *TColor*, містить інформацію про колір кожної точки графічної поверхні. Використовуючи властивість *Pixels*, можна задати необхідний колір для будь-якої точки графічної поверхні, тобто "намалювати" точку. Наприклад, інструкція *Form1.Canvas.Pixels[10,10]:=clRed* зафарбовує точку поверхні форми в червоний колір.

Розмірність масиву *Pixels* визначається розміром графічної поверхні. Розмір графічної поверхні форми (робочої області, яку також називають клієнтською) задається значеннями властивостей *ClientWidth* і *ClientHeight*, а розмір графічної поверхні компоненту *Image* - значеннями властивостей *Width* і *Height*. Лівій верхній точці робочої області форми відповідає елемент *Pixels [0,0]*, а правої нижньої – *Pixels [ClientWidth - 1],ClientHeight - 1]*.

Практична робота. Оброблення графічної інформації в Delphi

Мета: закріпити теоретичні знання та отримати практичні навички роботи з оброблення графічної інформації за допомогою системи Delphi 7.

Завдання 1. Створити програму в середовищі Delphi 7 для побудови на екрані дисплея комп'ютера геометричного об'єкта, зображеного на мал. (колір кіл: синій, чорний, червоний, жовтий, зелений відповідно).

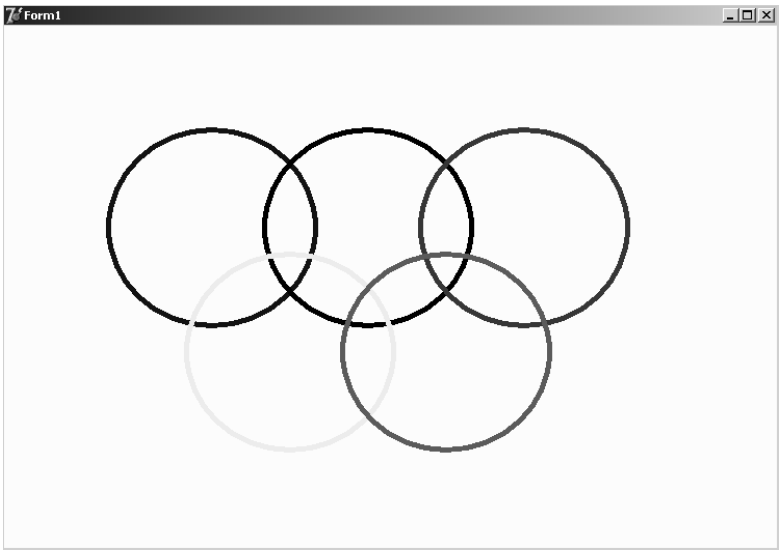

рис. 30. Зразок зображення

Завдання 2. Створити програму в середовищі Delphi 7 для побудови на екрані дисплея комп'ютера геометричного об'єкта, зображеного на мал. (колір фігур та заливка обирається довільно)

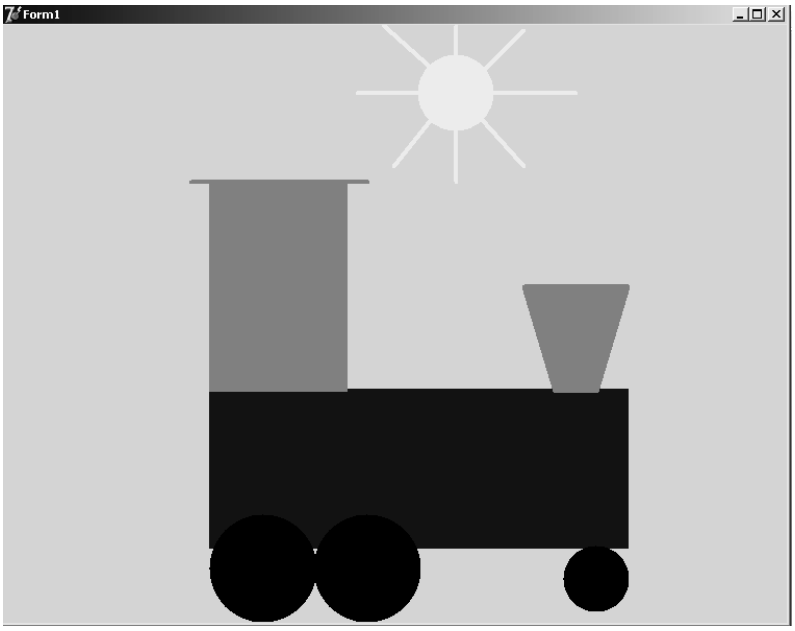

рис. 31. Зразок зображення