C 语言图形编程

内容提要：本文对 C 语言的图形功能做了详细的介绍，特别着重阐述了其丰富的库函数，并引申到了动画的基本设计方法。通过本文的学习，读者应该能够编制绘制基本图案的 C 语言程序和基本的动画程序。

关键字：图形模式的初始化、图形的坐标、坐标轴的变换、关闭图形系统、背景色和作图色的设置、调色板的设置、基本图形函数

引言：C 语言具有强大的图形编程功能。因为它不仅有高级语言那种完成复杂处理和运算的能力，还具有汇编语言的特点。它可以直接控制显示屏幕等系统硬件。C 语言具有丰富的图形函数，对图形程序和系统的开发和研制有很大的帮助。

正文：

1　本文导学

计算机图形学是一门研究怎样用计算机生成、处理和显示图形的学科。过去计算机主要用于科学计算和数据处理。

随着计算机的速度和性能的大大提高，应用范围的不断扩大，当前的计算机已经在许多领域上帮助人们完成各种各样的工作。近年来，随着计算机图形学的兴起和发展，直观的图形界面正在成为软件设计的新潮流，将逐步取代字符界面。

计算机图形学的应用可以追溯到 50 年代初。当时的美国麻省理工学院（MIT）研制出了计算机旋风Ⅰ号（Whirlwind Ⅰ），它带有一台可以显示图形的显示器。

1962 年，美国麻省理工学院（MIT）的 Ivan・E・Sutherland 发表一篇题为“Sketchpad：一人一机通讯的图形系统”的博士论文中首次使用了“计算机图形学”这个术语，并且在论文证明交互式计算机图形学是一个可行的、有用的研究领域。从此，计算机图形学蓬勃地发展起来，新的成果不断地涌现。

到了 70 年代，计算机图形技术的应用进入了使用化阶段，交互式图形系统在许多国家得到应用，许多关于计算机图形学的硬件被研制出来，计算机图形学得到了进一步的发展。

目前，计算机图形学已经进入社会的各个领域，主要有：计算机辅助设计与制造（CAD/CAM），计算机辅助教学，科学技术及事务管理，过程控制与系统环境模拟，艺术模拟等等。

C 语言具有强大的图形编程功能。因为它不仅有高级语言那种完成复杂处理和运算的能力，还具有汇编语言的特点。它可以直接控制显示屏幕等系统硬件。C 语言具有丰富的图形函数，对图形程序和系统的开发和研制有很大的帮助。

计算机图形学的内容十分丰富，技术也比较复杂。但是无论多么复杂的图形都是由点、线、距形和圆形等组成，掌握了这些简单的图形的处理方法，再去处理复杂的图形，也就不成问题了。

2　图形模式的初始化

在作图之前，一定要先设置显示器为图形方式才能作图。但是不同的显示器适配器有不同的图形分辨率。即是同一显示器适配器，在不同的模式下也有不同分辨率。

因此，在作图之前，必须根据显示器适配器种类将显示器设置成为某种图形模式，在未设置图形模式之前，微机系统默认屏幕为文本模式（80 列，25 行字符模式），此时所有图形函数均不能工作。

图形驱动程序由 Turbo C 出版商提供，文件扩展名为.BGI。根据不同的图形适配器有不同的图形驱动程序。

例如：对于 EGA、VGA 图形适配器就调用驱动程序 EGAVGA.BGI。如表 1 所示是 CGA、EGA、VGA 图形适配器的一些参数。

表 1　图形适配器的一些参数

gdriver		gmode		表示的色调	图象分辨率
驱动器常量	驱动器数值	模式常量	模式数值		
CGA	1	CGAC0	0	C0	320*200
		CGAC1	1	C1	320*200
		CGAC2	2	C2	320*200
		CGAC3	3	C3	320*200
		CGAHI	4	C4	640*200
EGA	3	EGALO	0	16 色	640*200
		EGAHI	1	16 色	640*350
VGA	9	VGALO	0	16 色	640*200
		VGAMED	1	16 色	640*350
		VGAHI	2	16 色	640*480
DETECT	0	自动检测显示器硬件			

设置屏幕为图形模式，可用图形初始化函数：

void far initgraph(int far *gdriver, int far *gmode, char *path);

图形初始化函数中的 gdriver 表示图形的驱动器，gmode 表示图形的模式，path 是指图形驱动程序所在的目录路径。

例 1：使用图形初始化函数设置 EGA 的 EGAHI 图形模式。

```
#include <graphics.h>
#include <stdio.h>
main()
{
   int gdriver = EGA, gmode=EGAHI;              /*设置 EGA 的 EGAHI 图形模式*/
     int errorcode;
     initgraph(&gdriver, &gmode, "");           /*初始化图形系统*/
     errorcode = graphresult();
     if (errorcode != grOk)
     {
        printf("Graphics error: %s\n", grapherrormsg(errorcode));
        printf("Press any key to halt:");
        getch();
        exit(1);
     }
     circle(100,100,50);                        /*画圆心为（100，100），半径为 50 的圆*/
     getch();
     closegraph();
}
```

程序输出结果如图 1 所示。

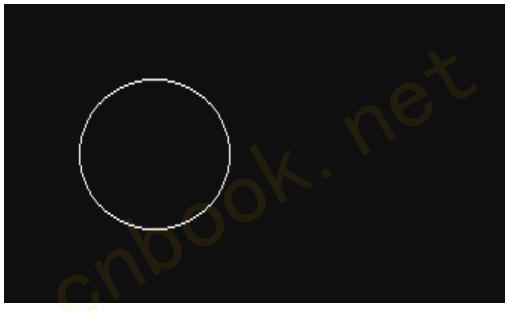

图 1

有时不知道所用的图形显示器适配器种类，这时可以用自动探测的方法，即使用自动探测函数：

void far detectgraph(int *gdriver, *gmode);

例 2：自动探测图形显示器适配器种类的方法。

```
#include <graphics.h>
#include <stdio.h>
main()
{
   int gdriver = EGA, gmode=EGAHI;          /*设置 EGA 的 EGAHI 图形模式*/
     int errorcode;
     detectgraph(&gdriver, &gmode);         /*自动探测图形显示器适配器种类*/
   initgraph(&gdriver, &gmode, "");         /*初始化图形系统*/
   errorcode = graphresult();
   if (errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
    }
   circle(100,100,50);                      /*画圆心为（100，100），半径为 50 的圆*/
   getch();
   closegraph();
}
```

程序输出结果如图 2 所示。

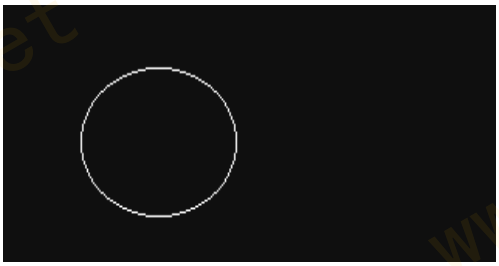

图 2

除了使用自动探测函数来实现自动探测外，还可以应用 DETECT 方法实现自动探测，即把 DETECT 直接赋予 gdriver（gdriver=DETECT）。

例 3：应用 DETECT 方法自动探测图形显示器适配器种类。

```
#include <graphics.h>
#include <stdio.h>
main()
{
   int gdriver = DETECT, gmode;             /*自动探测图形显示器适配器种类*/
     int errorcode;
     initgraph(&gdriver, &gmode, "");       /* 初始化图形系统 */
   errorcode = graphresult();
   if (errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
   }
   circle(100,100,50);                      /* 画圆心为（100，100），半径为 50 的圆*/
   getch();
```

```
    closegraph();
}
```

程序输出结果如图 3 所示。

图 3

3　图形的坐标

在文本方式下，屏幕被分成 25、40 或 50 行，80 或 40 列宽（系统默认的坐标是：原点在屏幕左上角（1,1），坐标是整数，X 坐标从左向右增加，Y 坐标从上到下增加）。

在图形方式下，屏幕被划分为像素，每个像素在屏幕上显示一个点。像素的个数取决于机器中安装的视频适配器的类型和适配器的工作和方式（系统默认的坐标是：原点在屏幕左上角（0,0），坐标为整数，X 坐标从左向右增加，Y 坐标从上到下增加。屏幕上的 X、Y 值都有一个最大值。如图 4 所示屏幕的坐标范围）。

图 4

可以通过下面一个程序来获取屏幕最大的 x 值和 y 值：

例 4：获取屏幕最大的 x 值和 y 值。

```
#include <graphics.h>
#include <stdio.h>
main()
{
   int gdriver = DETECT, gmode, errorcode;
   int Max_x,Max_y;
   initgraph(&gdriver, &gmode, "");              /* 初始化图形系统 */
   errorcode = graphresult();
   if (errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
```

```
    }
    Max_x=getmaxx();                /*getmaxx()函数是获取当前图形状态下最大的 x 坐标值*/
    Max_y=getmaxy();                /*getmaxy()函数是获取当前图形状态下最大的 y 坐标值*/
    printf("The maximum value of x is %d\n",Max_x);
    printf("The maximum value of y is %d\n",Max_y);
    getch();
      closegraph();
}
```

程序输出结果如图 5 所示。

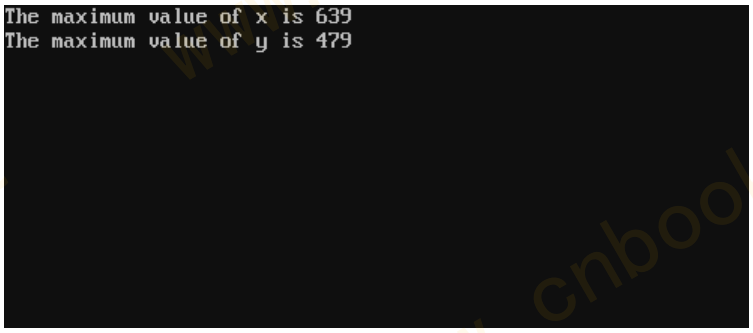

图 5

4　坐标轴的变换

C 语言中，在图形模式下（系统默认的坐标是：原点在屏幕左上角（0,0），坐标为整数，X 坐标从左向右增加，Y 坐标从上到下增加）。有时画图需要不同坐标原点，这就需要用到坐标变换。

现在假设需要以屏幕上的点（origin_x，origin_y）作为新坐标系的原点，那么原来坐标系的点的坐标（before_x，before_y）与对应的新坐标系的点（later_x，later_y）的关系有以下的计算公式：

before_x= origin_x＋later_x

before_y= origin_y－later_y

新旧坐标系的关系如图 6 所示。

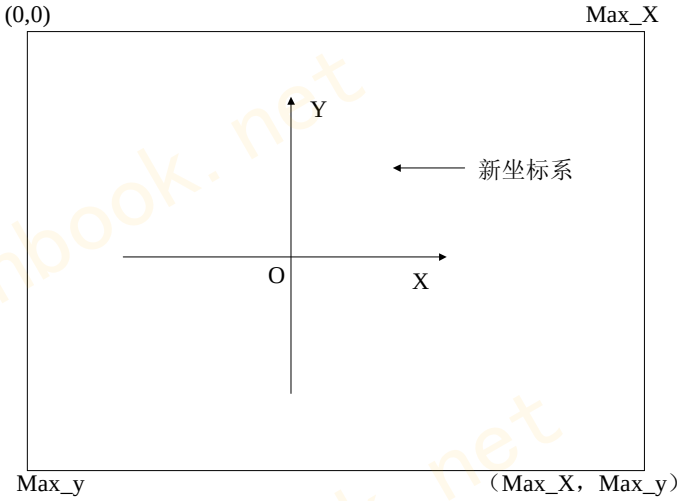

图 6

例 5：在以点（320，220）为原点的新坐标系中，画一个以该新坐标系中的点（50,50）为圆心，半径为 50 的圆。

```
#include <graphics.h>
#include <stdio.h>
main()
{
```

```
    int before_x,before_y,later_x,later_y;          /*定义新旧坐标的点*/
    int origin_x,origin_y;                          /*新坐标的原点在旧坐标上对应的点的坐标*/
    int gdriver = DETECT, gmode, errorcode;
    later_x=50;later_y=50;
    origin_x=320;origin_y=220;
    initgraph(&gdriver, &gmode, "");                /*初始化图形系统*/
    errorcode = graphresult();
    if (errorcode != grOk)
    {
        printf("Graphics error: %s\n", grapherrormsg(errorcode));
        printf("Press any key to halt:");
        getch();
        exit(1);
    }
    before_x= origin_x+ later_x;          /*坐标变换*/
    before_y= origin_y-later_y;
    circle(before_x,before_y,50);
      /*画圆心为（before_x，before_y），半径为 50 的圆*/
    getch();
    closegraph();
}
```

程序输出结果如图 7 所示。

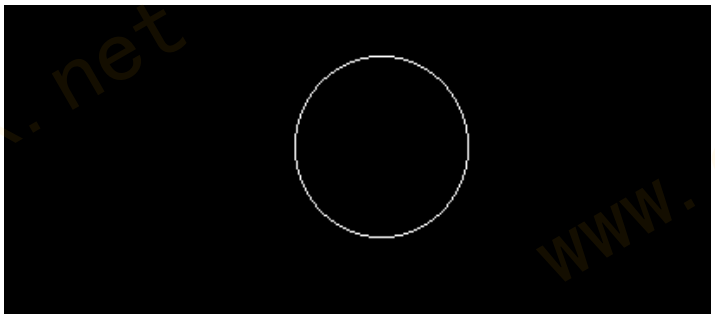

图 7

5 屏幕文本模式和图形模式之间的切换

在一些情况下，需要屏幕文本模式和图形模式之间的互相切换。C 语言提供了三个函数，可以实现这个功能。

（1）int far getgraphmode(void);

这个函数的功能是返回当前图形模式，前提是先前已经成功调用了 initgraph()函数。

（2）void far restorecrtmode(void);

这个函数的功能是将屏幕模式恢复为图形初始化前的模式。

（3）void far setgraphmode(int mode);

这个函数的功能是将系统设置成指定的图形模式且清屏。

例 6：屏幕文本模式和图形模式之间的切换的例子。

```
#include <graphics.h>
#include <stdio.h>
main()
{
    int gdriver = DETECT, gmode;              /*自动探测图形显示器适配器种类*/
    int errorcode;
    initgraph(&gdriver, &gmode, "");          /* 初始化图形系统 */
```

```
    errorcode = graphresult();
    if(errorcode != grOk)
    {
       printf("Graphics error: %s\n", grapherrormsg(errorcode));
       printf("Press any key to halt:");
       getch();
       exit(1);
    }
    circle(100,100,50);                    /* 画圆心为（100，100），半径为 50 的圆*/
    getch();
    gmode=getgraphmode();                  /*获取当前图形模式*/
    restorecrtmode();                      /*将屏幕模式恢复为图形初始化前的模式(文本模式)*/
    printf("We're now in text mode.\n");
    printf("Press any key to return to graphics mode:");
    getch();
    setgraphmode(gmode);
    circle(250,300,50);                    /* 另画圆心为（250,300），半径为 50 的圆*/
    getch();
    closegraph();
}
```

程序输出结果为：

（1）程序刚开始运行时是图形模式，如图 8 所示。

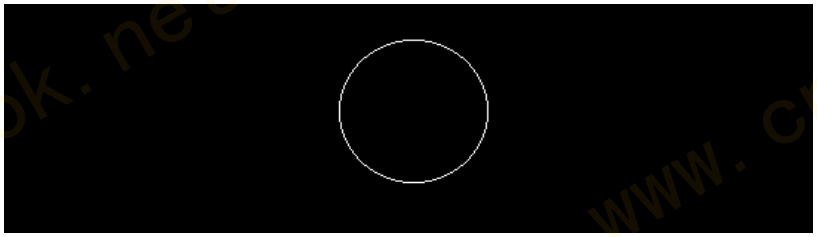

图 8

（2）切换为文本模式如图 9 所示。

图 9

（3）最后返回图形模式如图 10 所示。

图 10

6 关闭图形系统

当程序需要由图形模式退回到文本模式或者程序结束的时候需要用到关闭图形系统函数 void far closegraph(void)。因此有关图形的程序段应该在调用初始化函数 initgraph()后和调用关闭图形系统函数 closegraph()前。

7 背景色和作图色的设置

在 C 语言中，图形模式的屏幕颜色设置分为背景色的设置和作图色的设置。背景色的设置通过对函数 setbkcolor(int color), 作图色的设置通过对函 setcolor(int color)的调用来实现的。两个函数的形参 color 为整形数据，其取值范围为：0~15。color 的每一个值代表不同的颜色，如表 2 所示。

表 2　函数的形参 color 数值、含义及符号常数说明

颜色数值	含义	符号表示	颜色数值	含义	符号表示
0	黑色	BLACK	8	深灰	DARKGRAY
1	蓝色	BLUE	9	淡蓝	LIGHTBLUE
2	绿色	GREEN	10	淡绿	LIGHTGREEN
3	青色	CYAN	11	淡青	LIGHTCYAN
4	红色	RED	12	淡红	LIGHTRED
5	洋红	MAGENTA	13	淡洋红	LIGHTMAGENTA
6	棕色	BROWN	14	黄色	YELLOW
7	淡灰	LIGHTGRAY	15	白色	WHITE

例 7：用白色的背景，红色的画笔，以点（100，100）为圆心，50 为半径作一个圆。

```
#include <graphics.h>
#include <stdio.h>
main()
{
   int gdriver = DETECT, gmode, errorcode;
   initgraph(&gdriver, &gmode, "");                  /* 初始化图形系统 */
   errorcode = graphresult();
   if (errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
   }
   setbkcolor(WHITE);                                /* 设置背景颜色 */
   setcolor(RED);                                    /*设置画笔颜色*/
   circle(100,100,50);                               /* 画圆心为（100，100），半径为 50 的圆*/
   getch();
   closegraph();
}
```

程序输出结果如图 11 所示。

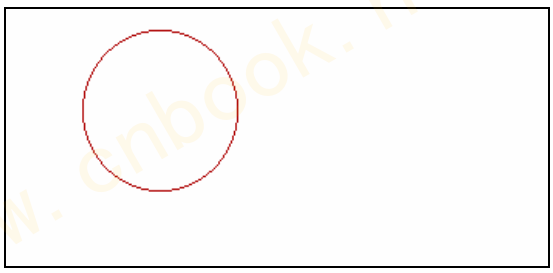

图 11

8 调色板的设置

对于 CGA 适配器，有四块调色板，每块调色板有四种颜色选择，颜色值为 0~3，调色板的值也是 0~3，如表 3 所示。

表 3 CGA 适配器调色板说明

调色板		颜色数值			
符号表示	符号的对应值	0	1	2	3
C0	0	背景色	绿色	红色	黄色
C1	1	背景色	青色	洋红色	白色
C2	2	背景色	淡绿色	淡红色	黄色
C3	3	背景色	淡青色	淡洋红色	白色

注：背景色可以为表 3 中 16 种颜色的一种。

例 8：将图形系统设置为 CGA 调色板四种中的 C1。

```
#include <graphics.h>
#include <stdio.h>
main()
{
   int gdriver = CGA, gmode=CGAC1;
   int errorcode;
   initgraph(&gdriver, &gmode, "");                    /* 初始化图形系统 */
   errorcode = graphresult();
   if (errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
   }
   printf("Now the palette is C1.\n");
    getch();
    closegraph();
 }
```

程序输出结果如图 12 所示。

图 12

9 基本图形函数

9.1 画点函数 putpixel()

函数 putpixel()的作用是在屏幕上指定的位置打一个点。

该函数的定义原形为：void far putpixel (int x, int y, int pixelcolor);
参数 x 和 y 是该指定点的坐标，pixelcolor 是打点时所用的颜色。

例 9：画一个矩形，每条边用不同的颜色。

```
#include <graphics.h>
#include <stdio.h>
main()
{
   int i;
   int gdriver = DETECT, gmode;          /*自动探测图形显示器适配器种类*/
   int errorcode;
   initgraph(&gdriver, &gmode, "");       /* 初始化图形系统 */
   errorcode = graphresult();
   if(errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
   }

   for(i=200;i<501;i++)
   /*从点(200,100)到点(500,100)用红色画线*/
   {
      putpixel(i,100,RED);
   }
   getch();                              /*按任何一键继续*/
   for(i=100;i<400;i++)
   /*从点(500,100)到点(500,400)用绿色画线*/
   {
      putpixel(500,i,GREEN);
   }
   getch();                              /*按任何一键继续*/
   for(i=200;i<500;i++)
   /*从点(200,400)到点(500,400)用蓝色画线*/
   {
      putpixel(i,400,BLUE);
   }
   getch();                              /*按任何一键继续*/
   for(i=100;i<400;i++)
   /*从点(200,100)到点(200,400)用黄色画线*/
   {
      putpixel(200,i,YELLOW);
   }
   getch();                              /*按任何一键继续*/
   closegraph();                         /*关闭图形系统*/
   getch();
   closegraph();
}
```

程序输出结果如图 13 所示。

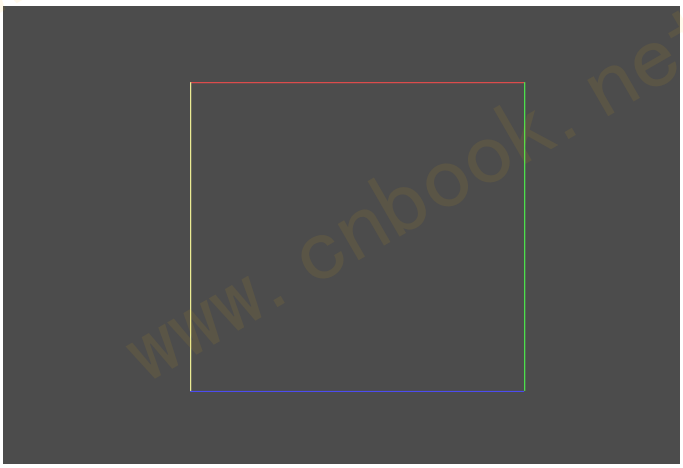

图 13

程序运行的结果是：一个长为 300，宽为 300 的矩形，其中顶的颜色是红色，右侧边的颜色是绿色，底边的颜色是蓝色，左侧边的颜色黄色。

9.2 坐标位置的函数 getmaxx()，getmaxy()，getx()，gety()，moveto()，moverel()

关于坐标位置的函数 getmaxx()，getmaxy()，getx()，gety()，moveto()，moverel()的函数原型及其功能，下面分别说明：

（1）函数 getmaxx()，getmaxy()

原形：int far getmaxx(void)；int far getmaxy(void)；

功能：获取屏幕最大的 x 值和 y 值。

（2）函数 getx()，gety()

原形：int far getx(void)；int far gety(void)；

功能：返回当前图形位置的 x 坐标和 y 坐标。

（3）函数 moveto()，moverel()

原形：void far moveto(int x, int y)；void far moverel(int dx, int dy)；

功能：函数 moveto(int x, int y)的功能是把画笔从当前位置移到点(x,y)。

函数 moverel(int dx, int dy) 的功能是把画笔从当前位置(x0,y0)移到点(x0+dx,y0+dy)。

例 10：在屏幕中央按笔画顺序写一个“巨”字。

```
#include <graphics.h>
#include <stdio.h>
main()
{
   int i;
   char msg[50];
   int x_start,y_start;                          /*每条线的开始的坐标(x_start,y_start)*/
   int gdriver = DETECT, gmode;                  /*自动探测图形显示器适配器种类*/
   int errorcode;
   initgraph(&gdriver, &gmode, "");              /*初始化图形系统*/
   errorcode = graphresult();
   if(errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
```

```
    }
    x_start=getmaxx()/2-30;                    /*第一笔：横*/
    y_start=getmaxy()/2-30;
    for(i=x_start;i<=x_start+60;i++)
    putpixel(i,y_start,RED);
    getch();
    for(i=x_start;i<=x_start+60;i++)           /*第二笔：横折*/
    putpixel(i,y_start+20,RED);
    x_start=i-1;
    moveto(x_start,y_start+20);
    x_start=getx();
    y_start=gety();
    for(i=y_start;i<=y_start+15;i++)
    putpixel(x_start,i,RED);
    getch();
    moverel(-60,15);                           /*第三笔：横*/
    x_start=getx();
    y_start=gety();
    for(i=x_start;i<=x_start+60;i++)
    putpixel(i,y_start,RED);
    getch();
    moverel(0,-35);                            /*第四笔：竖折*/
    x_start=getx();
    y_start=gety();
    for(i=y_start;i<=y_start+55;i++)
    putpixel(x_start,i,RED);
    moverel(0,55);
    x_start=getx();
    y_start=gety();
    for(i=x_start;i<=x_start+60;i++)
    putpixel(i,y_start,RED);
    getch();
    getch();
    closegraph();
  }
```

程序输出结果如图 14 所示。

图 14

9.3 图形模式下的文本输出函数 outtext()，outtextxy()

有时想给屏幕上的图形加点标识或解释，这就需要用到图形模式下的文本输出函数 outtext()，

outtextxy()。关于函数 outtext()，outtextxy()的原型及其功能，下面分别说明：

（1）函数 outtext()

原型：void far outtext(char far *textstring)；

功能：在图形模式下输出文本 textstring 输出文本时，可以先用 moveto()函数把画笔移到指定的点。

（2）函数 outtextxy()

原型：void far outtextxy(int x, int y, char *textstring)；

功能：图形模式下指定的点（x,y）开始输出文本 textstring。

例 11：分别用函数 outtext()和 outtextxy()在指定的点输出文本。

```
#include <graphics.h>
#include <stdio.h>
main()
{
   int gdriver = DETECT, gmode;                    /*自动探测图形显示器适配器种类*/
   int errorcode;
   initgraph(&gdriver, &gmode, "");                /* 初始化图形系统 */
   errorcode = graphresult();
   if(errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
   }
   moveto(310,150);                                /*把画笔定位于点(310,150)*/
   outtext("The \"outtext()\" is up.");            /*然后用函数 outtext()输出文本*/
   getch();                                        /*停顿，按任意键继续*/
     outtextxy(310,350,"The \"outtextxy()\" is down.");   /*用函数 outtextxy()输出文本*/
     getch();
   closegraph();
}
```

程序输出结果如图 15 所示。

图 15

9.4 画线函数 line()，lineto()，linerel()

前几节，用函数 putpixel()来画线段。现在，可以利用 line()，lineto()，linerel()三个函数来实现画线的功能。

关于 line()，lineto()，linerel()三个函数的原型及其功能，下面分别说明：

（1）函数 line()

原型：void far line(int x0, int y0, int x1, int y1);

功能：在图形模式下画一条以点(x0,y0)，(x1,y1)为端点的线段。

（2）函数 lineto()

原型：void far lineto(int x, int y);

功能：在图形模式下画一条以当前点为一个端点，(x,y)为另一个端点的线段，并改变当前画笔的位置。

（3）函数 linerel()

原型：void far linerel(int dx, int dy);

功能：在图形模式下画一条以当前点为一个端点，(x+dx,y+dx)为另一个端点的线段。

例 12：利用画线函数在屏幕中央画一个“巨”字。

```
#include <graphics.h>
#include <stdio.h>
main()
{
   int x,y;
   int gdriver = DETECT, gmode;                    /*自动探测图形显示器适配器种类*/
   int errorcode;
   initgraph(&gdriver, &gmode, "");
   /*初始化图形系统*/
   errorcode = graphresult();
   if(errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
   }
   x=getmaxx()/2-30;
   y=getmaxy()/2-27;
   line(x,y,x+60,y);                                /*第一笔：横*/
   getch();
   moveto(x,y+20);                                  /*第二笔：横折*/
   lineto(x+60,y+20);
   getch();
   linerel(0,15);
   getch();
   moverel(-60,0);                                  /*第三笔：横*/
   lineto(x+60,y+35);
   getch();
   line(x,y,x,y+55);                                /*第四笔：竖折*/
   moveto(x,y+55);
   linerel(60,0);
```

```
    getch();
      closegraph();
}
```

9.5 画圆及圆弧函数 circle()，arc()

在画图的时候，需要用到圆形或部分的圆（圆弧）。C 语言中提供了两个函数 circle()，arc()实现这些功能。

关于画圆及圆弧函数 circle()，arc()的原型及其功能，下面分别说明：

（1）画圆函数 circle()

原型：void far circle(int x, int y, int radius);

功能：在图形模式下画一个以点(x,y)为圆心，radius 为半径的圆。

（2）圆弧函数 arc()

原型：void far arc(int x, int y, int stangle, int endangle, int radius);

功能：在图形模式下画一个以点(x,y)为圆心，radius 为半径，起始角为 stangle，终止角为 endangle 的圆弧。

例 13：利用画圆及圆弧函数 circle()，arc()画一个“小眼睛”。

```
#include <graphics.h>
#include <stdio.h>
main()
{
   int x,y;
   int stangle,endangle,radius1,radius2;
   int gdriver = DETECT, gmode;                    /*自动探测图形显示器适配器种类*/
   int errorcode;
   initgraph(&gdriver, &gmode, "");
   /*初始化图形系统*/
   errorcode = graphresult();
   if(errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
   }
   x=getmaxx()/2;
   y=getmaxy()/2;
   stangle=25;
   endangle=155;
   radius1=30;
   radius2=45;
   circle(x,y,radius1);                            /*画“眼珠”*/
   setcolor(RED);
   arc(x,y,stangle,endangle,radius2);              /* 画“上眼眶”*/
   arc(x,y,stangle+180,endangle+180,radius2);      /* 画“下眼眶”*/
   getch();
   closegraph();
}
```

程序输出结果如图 16 所示。

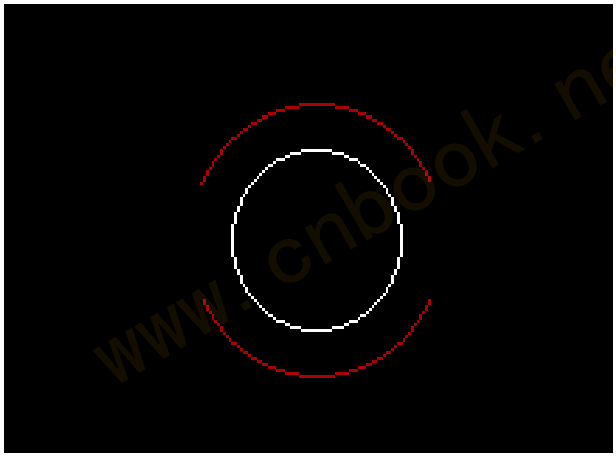

图 16

9.6 画椭圆函数 ellipse()

该函数的原型及其功能如下：

原型：ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius);

功能：画一个以点(x,y)为中心，起始角为 stangle，终止角为 endangle，长半轴为 xradius，短半轴为 yradius 的椭圆。

例 14：利用椭圆函数 ellipse()修改例 12 的画“小眼睛”的程序。

```
#include <graphics.h>
#include <stdio.h>
main()
{
   int x,y;
   int stangle,endangle,radius,xradius,yradius;
   int gdriver = DETECT, gmode;                    /*自动探测图形显示器适配器种类*/
   int errorcode;
   initgraph(&gdriver, &gmode, "");                 /*初始化图形系统*/
   errorcode = graphresult();
   if(errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
   }
   x=getmaxx()/2;
   y=getmaxy()/2;
   stangle=15;
   endangle=165;
   radius=30;
   xradius=60;
   yradius=40;
   circle(x,y,radius);                                /*画眼珠*/
   setcolor(RED);
   ellipse(x,y,stangle,endangle,xradius,yradius);     /*画上眼眶*/
   ellipse(x,y,stangle+180,endangle+180,xradius,yradius);   /*画下眼眶*/
     getch();
     closegraph();
}
```

9.7 画矩形框函数 rectangle()

该函数的原型及其功能如下：

原型：rectangle(int left, int top, int right, int bottom)；

功能：画一个矩形，其左上角坐标是(left,top)，右下角坐标是（right,bottom)。

例 15：画一个五彩的矩形。

```
#include <graphics.h>
#include <stdio.h>
#include <dos.h>
main()
{
    int i;
    int left, top,right,bottom;
    int gdriver = DETECT, gmode;              /*自动探测图形显示器适配器种类*/
    int errorcode;
    initgraph(&gdriver, &gmode, "");          /*初始化图形系统*/
    errorcode = graphresult();
    if(errorcode != grOk)
    {
       printf("Graphics error: %s\n", grapherrormsg(errorcode));
       printf("Press any key to halt:");
       getch();
       exit(1);
    }
    left=getmaxx()/2-120;
    top=getmaxy()/2-90;
    right=getmaxx()/2+120;
    bottom=getmaxy()/2+90;
    for(i=0;i<16;i++)
    {
       setcolor(i);                           /*改变画笔的颜色*/
       rectangle(left,top,right,bottom);      /*画矩形*/
       delay(30000);                          /*延时函数 void delay(unsigned milliseconds)*/
    }
    getch();
    closegraph();
}
```

9.8 设定线型函数 setlinestyle()

setlinestyle()函数用于设置当前绘图所用的线型和线宽。这些设置仅对直线类图形有效。setlinestyle()函数的原型为：

void far setlinestyle(int linestype, unsigned upattern,int thick)；

其参数的含义如下：

（1）linestype 整型变量。用于设定所画直线的类型，如表 4 所示。

表 4 linestype 整型变量符号常数、数值及含义说明

符号表示	对应数值	含义
SOLID_LINE	0	实线（系统缺省线型）
DOTTED_LINE	1	点线
CENTER_LINE	2	中心线
DASHED_LINE	3	虚线
USERBIT_LINE	4	用户自定义线型

（2）upattern，无符号整型变量。当用户自定义线型的时候使用该参数，该参数常用十六进制

数来表示。

例如：当 upattern 取 0xFFFF 时表示线型是实线；当 upattern 取 0xAAAA 时表示线型是点线。如果线型不是用户自定义时，该参数可取 0 值。

（3）thick 整型变量。用于设定所画直线的粗细，一般以像素表示。如表 5 所示。

表 5　thick 整型变量符号常数、数值及含义说明

符号表示	对应数值	含义
NORM_WIDTH	1	一个像素宽度(系统缺省宽度)
THICK_WIDTH	3	三个像素宽度

例 16：各种线型的展示。

```
#include <graphics.h>
#include <stdio.h>
main()
{
   int i;
   int left,top,right,bottom;
   int gdriver = DETECT, gmode;
   /*自动探测图形显示器适配器种类*/
   int errorcode;
   initgraph(&gdriver, &gmode, "");
   /*初始化图形系统*/
   errorcode = graphresult();
   if(errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
   }
   left=getmaxx()/2-70;
   top=getmaxy()/2-50;
   right=getmaxx()/2+70;
   bottom=getmaxy()/2+50;
   for(i=0;i<4;i++)
   {
      setlinestyle(i,0,NORM_WIDTH);                    /*设置线宽*/
      outtextxy(getmaxx()/2-105,top-60,"The line is NORM_WIDTH");
      switch(i)
      {
         case 0:outtextxy(getmaxx()/2-105,top-10,"The linestype is SOLID_LINE");
         case 1:outtextxy(getmaxx()/2-105,top-10,"The linestype is DOTTED_LINE");
         case 2:outtextxy(getmaxx()/2-105,top-10,"The linestype is CENTER_LINE");
         case 3:outtextxy(getmaxx()/2-105,top-10,"The linestype is DASHED_LINE");
      }
      rectangle(left,top,right,bottom);
      getch();
      setcolor(BLACK);                                 /*把画笔设置为背景色*/
        /*下面的八行程序是以背景色来重画刚画过的图形（文字），*/
        /*目的是擦去刚画过的图形和文字*/
      outtextxy(getmaxx()/2-105,top-60,"The line is NORM_WIDTH");
      switch(i)
```

```
    {

      case 0:outtextxy(getmaxx()/2-105,top-10,"The linestype is SOLID_LINE");
      case 1:outtextxy(getmaxx()/2-105,top-10,"The linestype is DOTTED_LINE");
      case 2:outtextxy(getmaxx()/2-105,top-10,"The linestype is CENTER_LINE");
      case 3:outtextxy(getmaxx()/2-105,top-10,"The linestype is DASHED_LINE");
    }

    rectangle(left,top,right,bottom);
    setcolor(WHITE);
  /*画笔的颜色恢复为以前的颜色*/
  }
}
```

程序输出结果如图 17a～图 17h 所示。

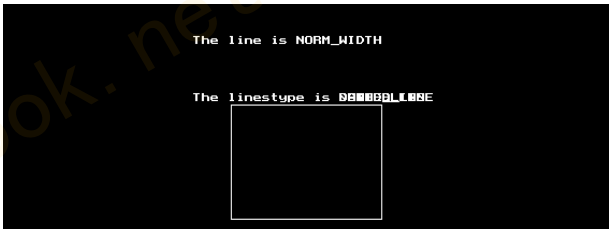

图 17a

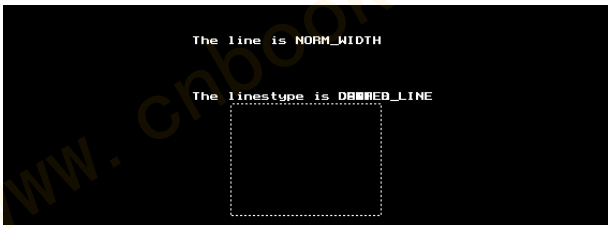

图 17b

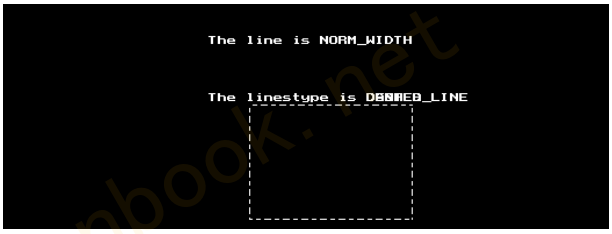

图 17c

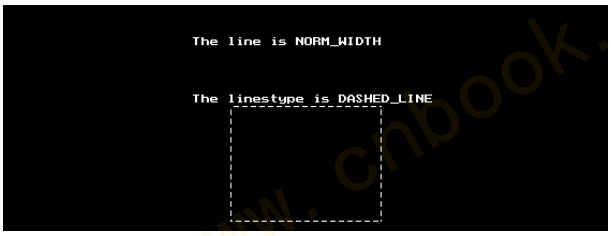

图 17d

图 17e

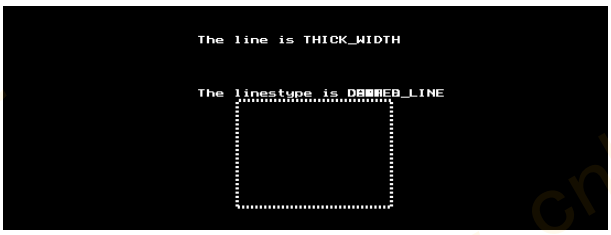

图 17f

图 17g

图 17h

（1）演示线宽为一个像素宽度，线型分别为实线、点线、中心线和虚线的距形。
（2）演示线宽为三个像素宽度，线型分别为实线、点线、中心线和虚线的距形。

9.9 填充函数 setfillstyle()，floodfill()，fillellipse()，sector()，fillpoly()

画一个图形，有时不仅需要对边着色，还需要对图形内部填充颜色，这就需要用到 C 语言的图

形填充函数 setfillstyle()，floodfill()，fillellipse()，sector()，fillpoly()。关于 C 语言的图形填充函数 setfillstyle()，floodfill()，fillellipse()，sector()，fillpoly()的原型及其功能，下面分别说明：

（1）setfillstyle()。

原型：void far setfillstyle(int pattern, int color);

功能：填充函数 setfillstyle()用于设置填充模式和颜色，以便用于填充指定的封闭区域。

参数 pattern 是填充所用的模式。模式一共有 11 种，如表 6 所示。

表 6　参数 pattern 的模式

符号表示	对应数值	含义
EMPTY_FILL	0	用背景色来填充
SOLID_FILL	1	用指定的颜色来填充
LINE_FILL	2	用线来填充
LTSLASH_FILL	3	用斜线来填充
SLASH_FILL	4	用粗斜线来填充
BKLASH_FILL	5	用粗反斜线来填充
LTKLASH_FILL	6	用反斜线来填充
HATCH_FILL	7	用网格线来填充
X HATCH_FILL	8	用斜交线来填充
INTERLEAVE_FILL	9	用间隔线来填充
WIDE_DOT_FILL	10	用稀疏点来填充
CLOSE_DOT_FILL	11	用密集点来填充
USER_FILL	12	用用户定义模式来填充

参数 color 是填充时所用的颜色。

（2）floodfill()。

原型：void far floodfill(int x, int y, int bcolor);

功能：用于填充一个有界区域，其填充的模式和颜色已由 setfillstyle()函数指定。

参数 x，y 指位于填充区域的任意一点（x,y）。参数 bcolor 是指被填充区域边界的颜色。

（3）fillellipse()。

原型：void far fillellipse(int x, int y, int xradius, int yradius);

功能：用当前颜色画出并填充一椭圆。点（x,y）是椭圆中心的坐标，xradius 是 x 轴方向的半轴长，yradius 是 y 轴方向的半轴长。

（4）sector()。

原型：void far sector(int x, int y, int stangle, int endangle, int xradius, int yradius);

功能：填充椭圆扇区并用当前颜色画出边线。点（x,y）是椭圆中心的坐标，stangle，endangle 分别是该椭圆扇区起始角和终止角，xradius 是 x 轴方向的半轴长，yradius 是 y 轴方向的半轴长。

（5）fillpoly()。

原型：void far fillpoly(int numpoints, int far *polypoints);

功能：用当前颜色画并填充一个多边形。

参数 numpoints 是一个整型数据，表示所画的多边形的顶点数目。参数 polypoints 是一个整型数组名，该数组中存放了 numpoints 个顶点坐标序列。

例 17：填充函数的用法举例。

```
#include <graphics.h>
#include <stdio.h>
main()
{
   int x,y,bcolor;
   int xradius,yradius;
   int stangle, endangle;
   int gdriver = DETECT, gmode;                    /*自动探测图形显示器适配器种类*/
   int errorcode;
```

```
    int poly[8];
    initgraph(&gdriver, &gmode, "");                    /*初始化图形系统*/
    errorcode = graphresult();
    if(errorcode != grOk)
    {
       printf("Graphics error: %s\n", grapherrormsg(errorcode));
       printf("Press any key to halt:");
       getch();
       exit(1);
    }
    x=150;
    y=240;
    stangle=0;
    endangle=360;
    xradius=100;
    yradius=70;
    poly[0]=400;     /*定义一个多边形,它有四个顶点:(400,50),(450,120),(500,140),(530,100) */
    poly[1]=50;
    poly[2]=450;
    poly[3]=120;
    poly[4]=500;
    poly[5]=140;
    poly[6]=530;
    poly[7]=100;
    circle(x,y,100);                                    /*画一个圆*/
    setfillstyle(SOLID_FILL, GREEN);                    /*设置填充模式和颜色*/
    floodfill(x,y,getmaxcolor());                       /*用绿颜色填充上面画的圆*/
    getch();
    ellipse(x+250,y,stangle,endangle,xradius,yradius);  /*画一个椭圆*/
    sector(x+250, y, 0, 90,xradius,yradius);            /*用绿颜色填充上面画的椭圆的 0~90 度的
扇区*/
    fillpoly(4, poly);                                  /*用绿颜色填充多边形*/
    getch();
    closegraph();
}
```

程序输出结果如图 18 所示。

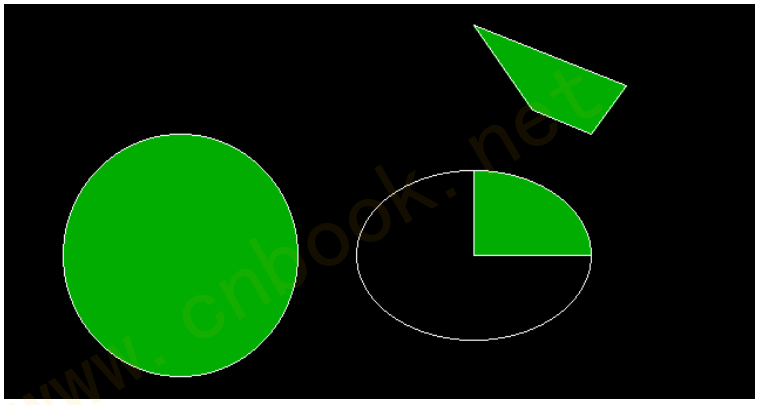

图 18

10　学以致用：综合举例

10.1　图形设计

1.　画一条具有立体效果的飘带

例 18：画一条具有立体效果的飘带。

```
#include <graphics.h>
#include <stdio.h>
#include <math.h>
main()
{
   int color=YELLOW;
   float a;
   float dotx_1,dotx_2,doty_1,doty_2,adjust_x=0.8,adjust_y=0.8;
   int gdriver = DETECT, gmode;                    /*自动探测图形显示器适配器种类*/
   int errorcode;
   initgraph(&gdriver, &gmode, "");                /*初始化图形系统*/
   errorcode = graphresult();
   if(errorcode != grOk)
   {

      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);

   }
   setbkcolor(0);
   setcolor(color);
   a=0;
   while(1)                                        /*开始画彩带*/
   {

      dotx_1=310*sin(2*a)*adjust_x+getmaxx()/2;     /*adjust_x 用来调整图形的宽度*/
      doty_1=100*sin(7*a)*cos(a/2.5)*adjust_y+getmaxy()/2;   /*adjust_y 用来调整图形的长度*/
      doty_2=doty_1;
      dotx_2=300*cos(1.4*a)*adjust_x+getmaxx()/2;
      moveto(dotx_1,doty_1);
      lineto(dotx_2,doty_2);
      a+=3.1416/600;
      if(color==YELLOW)      color=WHITE;             /*交替用黄色和白色画线*/
      else                    color=YELLOW;
      setcolor(color);
      if(a>3.1416) break;
  }
   getch();
   closegraph();
}
```

程序输出结果如图 19 所示。

图 19

2. 画金刚石图案

例 19：画金刚石图案（即将半径为 radius 的圆的圆周等分为 n 份，然后用直线将各等分点两两相连）。

```
#include <graphics.h>
#include <stdio.h>
#include <math.h>
#define PI    3.1415926
main()
{
   int n,i,j;
   int max_x,max_y;
   float x1,x2,y1,y2,radius,per_angle;
   int gdriver = DETECT, gmode;                    /*自动探测图形显示器适配器种类*/
   int errorcode;
   printf("Please input the radius and the \"n\"\n");
   scanf("%f%d",&radius,&n);
   initgraph(&gdriver, &gmode, "");                /*初始化图形系统*/
   errorcode = graphresult();
   if(errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
   }
   cleardevice();
   max_x=getmaxx();
   max_y=getmaxy();
   per_angle=2*PI/n;                               /* per_angle 为相邻两等分点对圆心所成的角度*/
   for(i=1;i<=n;i++)
   {
      x1=max_x/2+radius*cos(i*per_angle);
      y1=max_y/2-radius*sin(i*per_angle);
      for(j=i+1;j<=n;j++)
      {
         x2=max_x/2+radius*cos(j*per_angle);
```

```
            y2=max_y/2-radius*sin(j*per_angle);
            line(x1,y1,x2,y2);
          }
         }
      getch();
      closegraph();
    }
```

上面的程序中，有两个嵌套的 for 循环，外面的 for 循环的含义是选定直线的起始点，即依次从 1 到 n 选定直线的起始点（x1,y1）。里面的 for 循环的含义是选定直线的终止点，即依次从起始点的下一点到 n 选定直线的终止点（x2,y2）。然后用 line()函数把点(x1,y1)和点(x2,y2)连起来，所得的图形即为金刚石图案。请读者注意上述计算 x1，y1，x2，y2 的时候已经考虑到了坐标变换，在本程序中，新坐标原点是屏幕的中央，即点(max_x/2, max_y/2)。程序输出结果如图 20 所示。

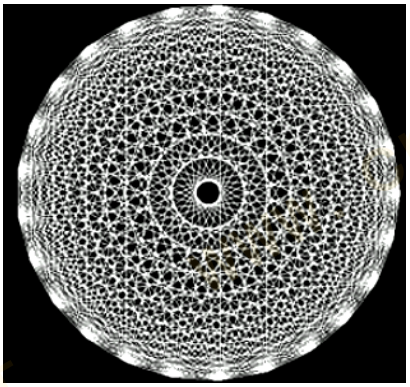

图 20

3. 七彩时空隧道的描绘

例 20：七彩时空隧道的描绘。

```
#include <graphics.h>
#include <stdio.h>
#define LENGTH   300
void draw_rectangle(n,color)          /*画矩形的函数,参数 n 是一个指示矩形左上角位置的参数,
*/
int n,color;                          /*参数 color 是指明画矩形时所用的颜色*/
{
   int x,y,length;
   x=200+5*n;
   y=100+5*n;
   length=300-10*n;
   setcolor(color);
   moveto(x,y);
   setcolor(color);
   lineto(x+length,y);
   lineto(x+length,y+length);
   lineto(x,y+length);
   lineto(x,y);
}
main()
{
   int i,n,color=1;
   int gdriver = DETECT, gmode;       /*自动探测图形显示器适配器种类*/
   int errorcode;
```

```
    printf("Please input the radius and the \"n\"\n");
    initgraph(&gdriver, &gmode, "");                    /*初始化图形系统*/
    errorcode = graphresult();
    if(errorcode != grOk)
    {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
    }
    for(i=300;i>0;i-=6)                                 /*画矩形组，造成隧道效果*/
    {
      n=50-i/6;
      color=color%15;                                   /*不断变换画距形的颜色,当 color 的值增到
15 的*/
      if(color==0) color++;                             /*时候, color 的值自动置为 1*/
      draw_rectangle(n,color);
      color++;
      delay(10000);
    }
    setcolor(GREEN);                                    /*以下是画矩形组的对角线*/
    moveto(200,100);                                    /*把画笔移到最外层矩形的左上顶点*/
    lineto(350-5,250-5);
    moveto(200+LENGTH,100+LENGTH);                      /*把画笔移到最外层矩形的右下顶点*/
    lineto(350+5,250+5);
    moveto(200+LENGTH,100);                             /*把画笔移到最外层矩形的右上顶点*/
    lineto(350+5,250-5);
    moveto(200,100+LENGTH);                             /*把画笔移到最外层矩形的左下顶点*/
    lineto(350-5,250+5);
  getch();
    closegraph();                                      /*关闭图形系统*/
  }
```

上面的三个例子的结构都不是很复杂，但是对读者的编程思维有很大的帮助。读者应从这些例子中体会到一些基本的编程思想：好的程序不一定复杂，简单的程序也可以完成复杂的任务，当然这需要程序员预先对编程任务的分析和算法的构造。

10.2 趣味小程序：用 C 语言实现动画的技巧

计算机动画的实现是通过一系列静止图像在不同的位置的重现。计算机图形动画技术可以分为画擦法和覆盖刷新法。

画擦法指的是先在屏幕上画上一个图形，跟着在很短的时间内擦掉它，并且迅速在这个图形的近旁画上这个图形，这样在人的眼睛看来，该图形好像是向某个方向移动了（利用了物理上的“视觉暂留效应”）。这种一擦一画的方法对实现简单图形的动态显示是比较有效的，然而当要显示较为复杂的图形的时候，由于画擦耗费的时间相对较长，致使画面在移动的时候出现闪烁的现象，影响动画的效果。

覆盖刷新法是一次性覆盖整个画面，并在瞬间完成。因此使用这种方法的动态特性比较平滑，动态效果比较好，可以克服画擦法的缺点。

下面就这两种方法各举一个例子。

例 21：使用画擦法来实现一个滚动的车轮。

```
#include <graphics.h>
```

```
#include <stdio.h>
main()
{
   int i,x,y;
   int stangle=10, endangle=80, radius=70;
   int gdriver = DETECT, gmode;                    /*自动探测图形显示器适配器种类*/
   int errorcode;
   printf("Please input the radius and the \"n\"\n");
   initgraph(&gdriver, &gmode, "");                /*初始化图形系统*/
   errorcode = graphresult();
   if(errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
   }
   x=getmaxx()/2-150;
   y=getmaxy()/2-35;
   for(i=0;i<=50;i++)
   {
      x=x+6;
      stangle=(stangle-10)%360;
      endangle=(endangle-10)%360;
      circle(x,y,radius);                          /*开始画图形*/
   arc(x,y,stangle,endangle,radius-35);
   arc(x,y,stangle+180,endangle+180,radius-35);
   getch();
   delay(5000);                                    /*暂停执行 5000 毫秒*/
   if(i!=50) setcolor(BLACK);                      /*开始擦去图形*/
   circle(x,y,radius);
   arc(x,y,stangle,endangle,radius-35);
   arc(x,y,stangle+180,endangle+180,radius-35);
   setcolor(WHITE);                                /*把画笔的颜色恢复*/
   }
   getch();
   closegraph();
}
```

程序分析：程序首先调用 initgraph()函数来初始化图形系统，然后分别将作为起始点的圆心坐标(x，y)赋值，跟着进入一个 for 循环语句。for 循环语句具体实现了“一个滚动的车轮”的功能。首先的语句 x=x+6;是用来不断改变程序运行的时候的圆心坐标，使圆会“动”。

语句 stangle=(stangle-10)%360;和 endangle=(endangle-10)%360；是用来不断改变圆里面的圆弧的起始角和终止角，起着一种参照的效果，使人觉得圆在“动”。语句 endangle=(endangle-10)%360；下面的三条语句用于画圆和圆弧。

语句 delay(5000)；使刚画出的图形在屏幕上停留 5000 毫秒。

接下来的语句 if(i!=50) setcolor(BLACK)；是把画笔的颜色设置为背景色，在原来的位置用背景色重画圆和圆弧，即相当于擦去刚刚画的图形。因为 i=50 的时候所画的图形是中点的图形，所以不应该擦去，即当 i=50 的时候语句 setcolor(BLACK)；不执行。

语句 setcolor(WHITE);的作用是把画笔的颜色恢复。

执行程序时，屏幕上会出现一个从左滚到右的小球，不过效果不是很好，屏幕有点闪烁。读者

从而可以体会到画擦法实现动画的缺点。

在举覆盖刷新法的例子之前，先说几个有关的函数：

（1）putimage()函数。

原型：void far putimage(int x, int y, void far *bitmap, int op)；

功能：该函数的功能是在屏幕上输出一个位图。参数 x，y 是用来指定显示的位置；bitmap 是用来指定图像存储的地址；op 是用来指定图形的显示方式，如表 7 所示。

表 7　函数说明

参数名称	符号表示	对应数值	功能说明
Op	COPY_PUT	0	拷贝
	XOR_PUT	1	异或
	OR_PUT	2	或
	AND_PUT	3	与
	NOT_PUT	4	非

（2）getimage()函数。

原型：void far getimage(int left, int top, int right, int bottom, void far *bitmap);

功能：在屏幕指定的位置将一个位图保存到内存。点（left,top）和点（right,bottom）分别是矩形区域的左上角和右下角的坐标；bitmap 是存储区域的地址。

（3）imagesize()函数。

原型：unsigned far imagesize(int left, int top, int right, int bottom)；

功能：返回保存位图像所需的字节数。点（left,top）和点（right,bottom）分别是矩形区域的左上角和右下角的坐标。

例 22：用覆盖刷新法来实现车轮在隧道中滚动。

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
main()
{
   int i;
   int x,y,radius=70;
   int left,top,right,bottom,size;
   void *bitmap;
   int gdriver = DETECT, gmode;                    /*自动探测图形显示器适配器种类*/
   int errorcode;
   printf("Please input the radius and the \"n\"\n");
   initgraph(&gdriver, &gmode, "");                 /*初始化图形系统*/
   errorcode = graphresult();
   if(errorcode != grOk)
   {
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
   }
   x=getmaxx()/2-70;
   y=getmaxy()/2-70;
   left=x-70;
   top=y-70;
   right=left+140;
```

```
    bottom=top+140;
    line(left,top-1,left+300,top-1);                    /*画隧道的上壁*/
    line(left,top+140+1,left+300,top+140+1);            /*画隧道的下壁*/
    size = imagesize(left,top,right,bottom);
    bitmap= malloc(size);
    circle(x,y,radius);                                 /*开始画图形*/
    getch();
    getimage(left, top, right,bottom,bitmap);
    for(i=0;i<50;i++)
    {
       putimage(left,top, bitmap, XOR_PUT);             /*擦去图形*/
       delay(100);                                      /*暂停执行 100 毫秒*/
       left=left+6;
       putimage(left,top, bitmap, COPY_PUT);            /*画图形*/
    }
    getch();
    closegraph();
}
```

程序分析：程序首先调用 initgraph()函数来初始化图形系统，然后给参数 x，y（起始的圆心坐标），left，top，right，bottom（要保存到内存的屏幕区域的范围）赋值。跟着用两个 line()函数画隧道。下一步是通过用语句 size = imagesize(left,top,right,bottom);计算保存到内存的屏幕区域的范围所费的内存的大小。接着通过语句 bitmap= malloc(size);获取系统分给程序的内存区域的地址。

画了初始圆之后利用语句 getimage(left, top, right, bottom, bitmap);把该区域的图像保存到内存。最后，擦去刚才所画的图形，把输出图形的坐标向右移动，利用语句 putimage(left,top, bitmap, COPY_PUT);输出图形。接着又擦去刚才所画的图形，把输出图形的坐标向右移动，利用语句 putimage(left,top, bitmap, COPY_PUT);输出图形。如此循环，造成一种车轮在隧道中滚动的动画的效果。