第3章图形模式补充知识

课程目标

- 二维图形补充知识
- 普通直线段扫描转换算法
- 数值微分画线算法
- 中点画线算法
- Bresenham画线算法

3.1 补充知识

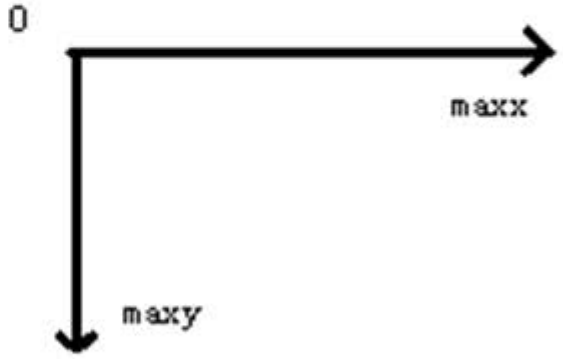
- 图形模式
- 点坐标
- 绘制图形的一般步骤
- 图形库函数
- 像素的扫描转换

1. 图形模式

- 图形模式：屏幕上显示图形的方式
- 屏幕是由像素点组成的，像素点多少决定了屏幕的分辨率
- 屏幕上每个像素的显示位置用点坐标来描述

2. 点坐标

- 屏幕左上角为坐标原点（0，0）
- 水平方向为x轴
- 垂直方向为y轴



3. 绘制图形的一般步骤

- 设置屏幕为图形模式
- 选择背景和实体颜色
- 计算坐标
- 调用绘制语句绘制实体

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

int main()
{
  initgraph(640, 480); // 创建绘图窗口，分辨率 640x480
  circle(320, 240, 100);// 画圆，圆心 (320, 240)，半径 100
  _getch();                // 按任意键继续
  closegraph();       // 关闭图形界面
  return 0;
}
```

4. 软件安装

- VC++6.0
- EasyX

在VisualC++使用#include <graphics.h>，编译时会报错显示

```
1 | Cannot open include file: 'graphics.h': No such file or directory
```

原因是graphics.h是TC里面专有的图形库，若要使用VC++来操作则需要安装相关的头文件和库文件：

```
1 | 头文件：graphics.h
2 | 库文件：graphics.lib
```

这里推荐一个及其简单快捷的安装方式，有人制作了一个软件包EasyX，里面有头/库文件和安装方法，

在安装向导中点击下一步，并在VisualC++那行点击安装即可完成

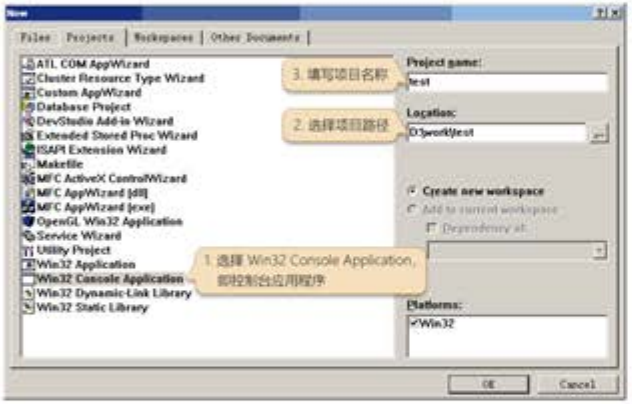
不过这里的文件是经过封装处理过的，但其原理都是一样，即将头文件放进VisualC++安装路径里的Microsoft Visual Studio\ **VC98\Include** 文件中，库文件放进VisualC++安装路径里的Microsoft Visual Studio\ **VC98\Lib** 文件中

注意：EasyX只能应用于C++程序，不能应用于C程序

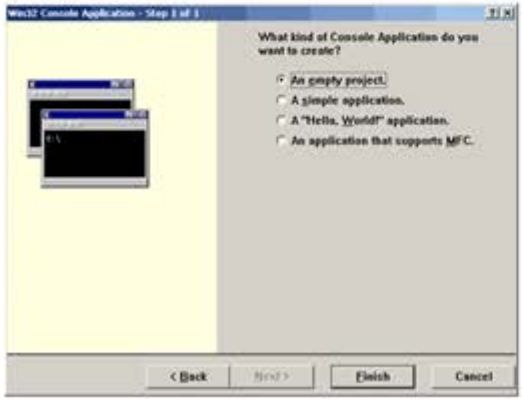
5. 创建新项目

1. 启动 VC6，点击菜单 File -> New...，打开 New 对话框：

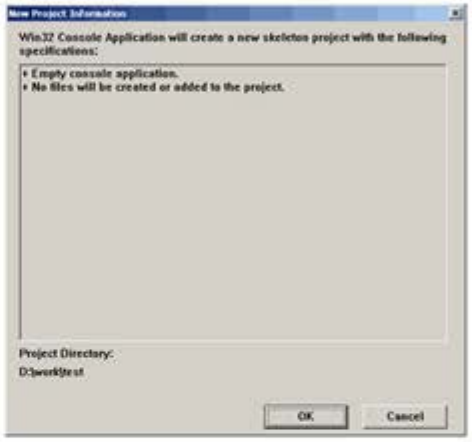
2. 在 Projects 选项卡里选择 "Win32 Console Application" 类型的项目，并选择项目所在路径，填写项目名称，点击"OK"继续：



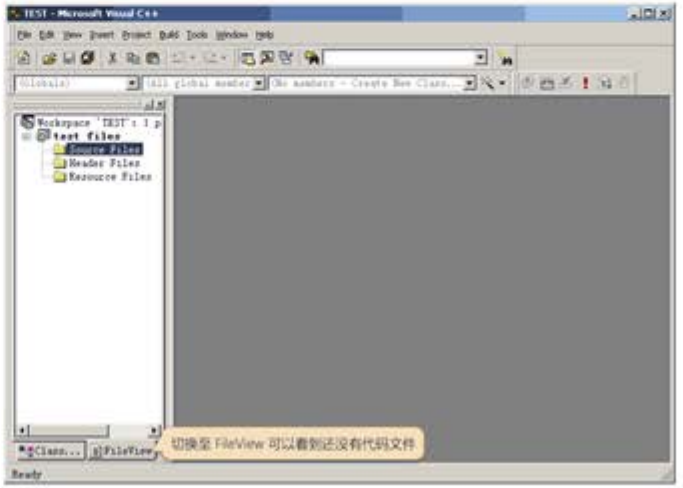
3. 选择 "An empty project" 创建一个空项目，稍后再添加代码文件，点击"Finish"完成：



4. 这一步列出了项目的相关信息，直接点击"OK"完成：



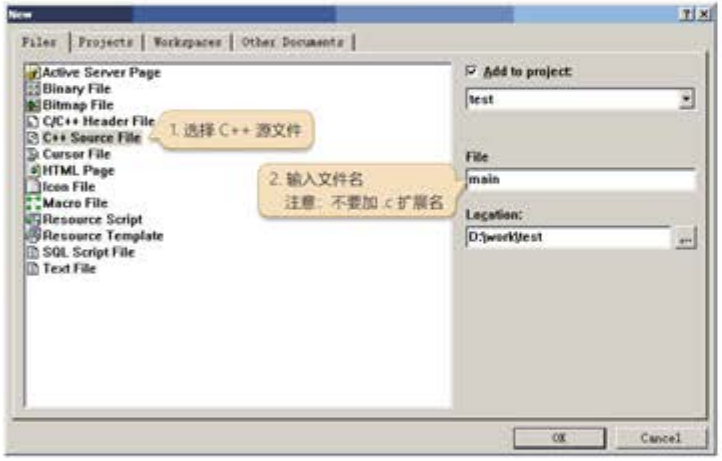
5. 创建项目完毕，在 FileView 选项卡里可以看到项目里面没有任何文件：



5.创建新项目

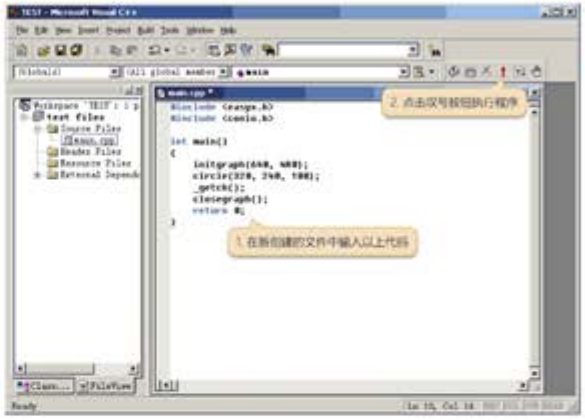
6. 点击菜单 File -> New...，打开 New 对话框：

7. 在 Files 选项卡里选择 "C++ Source File"，然后填写文件名（注意不要加 .c 扩展名），点击"OK"继续：



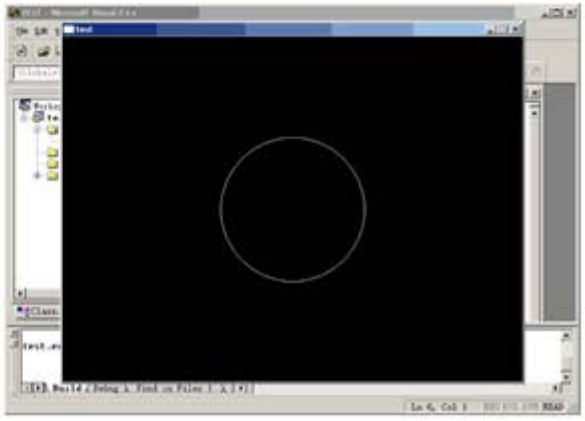

```
#include <graphics.h>
#include <conio.h>

int main()
{
    initgraph(640, 480);              // 创建绘图窗口，分辨率 640x480
    circle(320, 240, 100);            // 画圆，圆心 (320, 240)，半径 100
    _getch();                         // 按任意键继续
    closegraph();                     // 关闭图形界面
    return 0;
}
```

8. 新文件创建后，键入以下代码，点击双号按钮执行程序：



9. 提示是否要在执行前保存程序，点击"是"，确定保存程序：

10. 然后即可看到执行结果：



6. 学习单步执行

```c
#include <graphics.h>           // 绘图库头文件，绘图语句需要
#include <conio.h>              // 控制台输入输出头文件，_getch()语句需要

int main()
{
    initgraph(640, 480);        // 初始化640x480的绘图屏幕
    line(200, 240, 440, 240);   // 画线(200,240) - (440,240)
    line(320, 120, 320, 360);   // 画线(320,120) - (320,360)

    _getch();                   // 按任意键
    closegraph();               // 关闭绘图屏幕
    return 0;
}
```

解释：

1. 创建的绘图屏幕 640x480，表示横向有 640 个点，纵向有 480 个点。左上角是原点 (0, 0)，右下角坐标是 (639, 479)，x 轴 向右为正，y 轴向下为正（和数学的 y 轴相反）。
2. _getch 实现按任意键功能，按任意键后，程序继续执行。否则，程序会立刻执行 closegraph 以至于看不到绘制的内容。

调试执行与断点

1. 编译成功后，按 F5，直接调试执行，可以看到程序的执行结果。
2. 光标点到"line(320, 120, 320, 360); // 画线(320,120) - (320,360)"这行，按 F9，会看到这行左边多了一个红点，这是断点标记。再按 F9 可以取消断点。也可以用鼠标点一下左边的的红点位置来设置断点。
3. 再次按 F5，会看到代码执行到断点位置就不再执行了。程序断下来后，可以观察执行到此时的情况（如变量的值）。注意：由于绘图机制的限制，程序断下来时无法立刻看到绘图窗口的变化。
4. 再次按 F5，程序会从断下来的位置继续执行，直到结束。
5. 程序里可以设置多个断点。

单步执行

1. 编译成功后，按一下 F10（单步执行），会看到屏幕上出现一个黄色的小箭头，指示将要执行的代码。
2. 一下下按 F10，会看到黄色的小箭头逐步下移。注意：由于绘图机制的限制，程序断下来时无法立刻看到绘图窗口的变化。
3. 在 _getch() 这行按下 F10 后，无法继续按 F10，因为执行 _getch() 语句后，程序在等待用户输入。这时候切换到绘图窗口，并按任意键，_getch() 即可执行完毕，再切换回 VC，可以继续调试。
4. 直接按 F5 执行全部剩余程序，结束。

7. 熟悉更多的绘图语句

[常用的绘图语句]

- line(x1, y1, x2, y2); // 画直线 (x1,y1)-(x2,y2)，都是整形
- circle(x, y, r); // 画圆，圆心为 (x,y)，半径为 r
- putpixel(x, y, c); // 画点 (x,y)，颜色 c

还有很多，如画椭圆、圆弧、矩形、多边形，等等，请参考 EasyX 在线帮助 https://docs.easyx.cn

[设置颜色]

setlinecolor(c); // 设置画线颜色，如 setlinecolor(RED) 设置画线颜色为红色

常用的颜色常量可以用：

- BLACK 黑　　DARKGRAY 深灰
- BLUE 蓝　　LIGHTBLUE 亮蓝
- GREEN 绿　　LIGHTGREEN 亮绿
- CYAN 青　　LIGHTCYAN 亮青
- RED 红　　LIGHTRED 亮红
- MAGENTA 紫　　LIGHTMAGENTA 亮紫
- BROWN 棕　　YELLOW 黄
- LIGHTGRAY 浅灰　WHITE 白

7. 熟悉更多的绘图语句

[配出更多的颜色]

颜色除了前面写的 16 种以外，还可以自由配色。格式：RGB(r, g, b)

r / g / b 分别表示红色、绿色、蓝色，范围都是 0~255。例如，RGB(255,0,0) 表示纯红色。

红色和绿色配成黄色，因此 RGB(255, 255, 0) 表示黄色。

嫌调色麻烦可以用画笔里面的调色试试，调好了以后直接将数值抄过来就行。

例如，画两条红色浓度为 200 的直线，可以这么写：

```
setlinecolor(RGB(200, 0, 0));
line(100, 100, 200, 100);
line(100, 120, 200, 120);
```

[用数字表示颜色]

除了用 RGB(r,g,b) 方式外，还可以用16进制表示颜色，格式：0xbbggrr

例如，setlinecolor(0x0000ff) 和 setlinecolor(RGB(255, 0, 0)) 是等效的。

[延时语句]

这个很简单，Sleep(n) 就可以表示 n 毫秒的延时。例如延时 3 秒，可以用 Sleep(3000);

[作业]

1. 简单看一下绘图库的帮助文件，了解更多的绘图语句。
2. 绘制更丰富的图形内容，不低于20行。
3. 将延时语句适当的插入上个作业的代码中，看看执行效果。

8. 结合流程控制语句来绘图

例如，画10条直线的代码：

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

int main()
{
    initgraph(640, 480);

    for(int y=100; y<200; y+=10)
        line(100, y, 300, y);

    _getch();
    closegraph();
    return 0;
}
```

换一下循环的范围和间隔，看看效果。

还可以用来画渐变色，例如：

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

int main()
{
    initgraph(640, 480);

    for(int y=0; y<256; y++)
    {
        setcolor(RGB(0,0,y));
        line(100, y, 300, y);
    }

    _getch();
    closegraph();
    return 0;
}
```

8. 结合流程控制语句来绘图

配合 if 语句，实现红色、蓝色交替画线：

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

int main()
{
    initgraph(640, 480);

    for(int y=100; y<200; y+=10)
    {
        if ( y/10 % 2 == 1)    // 判断奇数行偶数行
            setcolor(RGB(255,0,0));
        else
            setcolor(RGB(0,0,255));

        line(100, y, 300, y);
    }

    _getch();
    closegraph();
    return 0;
}
```

[作业]

1. 画围棋棋盘。
2. 画中国象棋的棋盘。
3. 画国际象棋的棋盘，看手册找到颜色填充语句，实现国际象棋棋盘的区块填充。

9.数学知识在绘图中的运用

1. 最简单的，来个全屏的渐变色吧，是上一课的扩展。就是需要将 0~255 的颜色和 0~479 的 y 轴对应起来

c 表示颜色，范围 0~255

y 表示y轴，范围 0~479

于是：

c / 255 = y / 479

c = y / 479 * 255 = y * 255 / 479 （先算乘法再算除法可以提高精度）

看代码：

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

int main()
{
    initgraph(640, 480);

    int c;
    for(int y=0; y<480; y++)
    {
        c = y * 255 / 479;
        setlinecolor(RGB(0,0,c));
        line(0, y, 639, y);
    }

    _getch();
    closegraph();
    return 0;
}
```

9.数学知识在绘图中的运用

2. 画一个圆形的渐变色

首先，我们要用到圆形的基本公式：

x * x + y * y = r * r

让弧度从 0~2 * 3.14，然后需要根据弧度和半径算出 (x,y)，

用 pi 表示圆周率

用 r 表示半径

用 a 表示弧度（小数）

用 c 表示颜色

于是：

x = r * cos(a)

y = r * sin(a)

c = a * 255 / (2 * pi)

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

#define PI 3.14159265359

int main()
{
    initgraph(640, 480);

    int c;
    double a;
    int x, y, r = 200;
    for(a = 0; a < PI * 2; a += 0.0001)
    {
        x=(int)(r * cos(a) + 320 + 0.5);
        y=(int)(r * sin(a) + 240 + 0.5);
        c=(int)(a * 255 / (2 * PI) + 0.5);
        setlinecolor(RGB(c, 0, 0));
        line(320, 240, x, y);
    }

    _getch();
    closegraph();
    return 0;
}
```

10. 实现简单动画

所谓动画，其实是连续显示一系列图形而已。
结合到程序上，我们需要以下几个步骤：

1. 绘制图像
2. 延时
3. 擦掉图像

循环以上即可实现动画。

举一个例子，我们实现一条直线从上往下移动：

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

int main()
{
    initgraph(640, 480);

    for(int y = 0; y < 480; y++)
    {
        // 绘制绿色直线
        setlinecolor(GREEN);
        line(0, y, 639, y);

        // 延时
        Sleep(10);

        // 绘制黑色直线（即擦掉之前画的绿线）
        setlinecolor(BLACK);
        line(0, y, 639, y);
    }

    closegraph();
    return 0;
}
```

10. 实现简单动画

再看一个例子，实现一个圆从左往右跳动：

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

int main()
{
    initgraph(640, 480);

    for(int x = 100; x < 540; x += 20)
    {
        // 绘制黄线、绿色填充的圆
        setlinecolor(YELLOW);
        setfillcolor(GREEN);
        fillcircle(x, 100, 20);

        // 延时
        Sleep(500);

        // 绘制黑线、黑色填充的圆
        setlinecolor(BLACK);
        setfillcolor(BLACK);
        fillcircle(x, 100, 20);
    }

    closegraph();
    return 0;
}
```

11. 实现简单动画

再看一个例子，实现一个圆从左往右跳动：

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

int main()
{
    initgraph(640, 480);

    for(int x = 100; x < 540; x += 20)
    {
        // 绘制黄线、绿色填充的圆
        setlinecolor(YELLOW);
        setfillcolor(GREEN);
        fillcircle(x, 100, 20);

        // 延时
        Sleep(500);

        // 绘制黑线、黑色填充的圆
        setlinecolor(BLACK);
        setfillcolor(BLACK);
        fillcircle(x, 100, 20);
    }

    closegraph();
    return 0;
}
```

也就是说，移动的间距小、延时短，动画就会越细腻。但当画面较复杂时，会带来画面的闪烁

[作业]
绘制一个沿 45 度移动的球，碰到窗口边界后反弹。

12.捕获按键，实现动画的简单控制

最常用的一个捕获按键的函数：_getch()
前几课，都把这个函数当做"按任意键继续"来用，现在我们用变量保存这个按键：

char c = _getch();

然后再做判断即可。

不过程序执行到 _getch() 是会阻塞的，直到用户有按键才能继续执行。可游戏中总不能因为等待按键而停止游戏执行吧？所以，要有一个函数，判断是否有用户按键：_kbhit()
这个函数返回当前是否有用户按键，如果有，再用 _getch() 获取即可，这样是不会阻塞的。

即：

```
char c;
if (_kbhit())
    c = _getch();
```

12.捕获按键，实现动画的简单控制

举一个简单的例子，如果有按键，就输出相关按键。否则，输出“.”。每隔 100 毫秒输出一次。按 ESC 退出。
注：ESC 的 ASCII 码是 27。

完整代码如下：

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

int main()
{
    char c = 0;
    while(c != 27)
    {
        if (_kbhit())
            c = _getch();
        else
            c = '.';

        printf("%c", c);
        Sleep(100);
    }

    return 0;
}
```

12.捕获按键，实现动画的简单控制

结合上一课的简单动画，就可以做出来靠按键移动的图形了吧，看以下代码，实现 a、d 控制圆的左右移动：

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

int main()
{
    initgraph(640, 480);

    int x = 320;

    // 画初始图形
    setlinecolor(YELLOW);
    setfillcolor(GREEN);
    fillcircle(x, 240, 20);

    char c = 0;
    while(c != 27)
    {
        // 获取按键
        c = _getch();

        // 先擦掉上次显示的旧图形
        setlinecolor(BLACK);
        setfillcolor(BLACK);
        fillcircle(x, 240, 20);

        // 根据输入，计算新的坐标
        switch(c)
        {
            case 'a': x-=2; break;
            case 'd': x+=2; break;
            case 27: break;
        }

        // 绘制新的图形
        setlinecolor(YELLOW);
        setfillcolor(GREEN);
        fillcircle(x, 240, 20);

        // 延时
        Sleep(10);
    }

    closegraph();
    return 0;
}
```

请继续完成这个程序，实现以下功能：

1. 上下的控制；
2. 边界检测；
3. 结合 kbhit 实现惯性移动（即按一下方向键，圆就会一直向这个方向移动）

注：上下左右等按键的控制，会返回 2 个字符。由于该系列教程面向初学者，因此有兴趣的请查看 MSDN。

animation.ad.cpp

13. 结合流程控制语句来绘图

配合 if 语句，实现红色、蓝色交替画线：

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

int main()
{
    initgraph(640, 480);

    for(int y=100; y<200; y+=10)
    {
        if ( y/10 % 2 == 1)    // 判断奇数行偶数行
            setcolor(RGB(255,0,0));
        else
            setcolor(RGB(0,0,255));

        line(100, y, 300, y);
    }

    _getch();
    closegraph();
    return 0;
}
```

[作业]

1. 画围棋棋盘。
2. 画中国象棋的棋盘。
3. 画国际象棋的棋盘，看手册找到颜色填充语句，实现国际象棋棋盘的区块填充。

14. 随机函数

绘图中的应用

来一个简单的程序，在屏幕上任意位置画任意颜色的点(按任意键退出)：

```
#include <graphics.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>

int main()
{
    srand( (unsigned)time( NULL ) );

    initgraph(640, 480);

    int x, y, c;
    while(!_kbhit())
    {
        x = rand() % 640;
        y = rand() % 480;
        c = RGB(rand() % 256, rand() % 256, rand() % 256);
        putpixel(x, y, c);
    }

    closegraph();
    return 0;
}
```

14. 随机函数

随机种子

做了多次试验，我们会发现一个问题：虽然产生的数字是随机的，但每次产生的数字序列都一样。为了解决这个问题，我们需要用"随机种子"。随机函数的产生原理简单来说，就是：前一个随机函数的值，决定下一个随机函数的值。

根据这个原理我们可以知道：只要第一个随机函数的值确定了，那么后面数字序列就是确定的。如果想得到不同的数字序列，就需要确定第一个随机函数的值，对于设置第一个随机函数的值，叫做设置"随机种子"。容易理解，随机种子设置一次即可。

设置随机种子的函数如下：
srand(种子);

通常，我们用当前时间来做随机种子：
srand((unsigned)time(NULL));

因为使用 time 函数，所以记得引用 <time.h>。

14. 随机函数

函数简介

游戏中，许多情况都是随即发生的。还有一些图案程序，例如屏保，也是随即运动的。这就需要用随机函数。

随机函数很简单，只有一个：

rand()

该函数返回 0～32767 之间的一个整数。(不需要记住 32767 这个数字，大概知道这个范围就行了)

该函数在头文件 <stdlib.h> 中声明，使用前记得引用。

简单测试

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int r;
    for(int i=0; i<10; i++)
    {
        r = rand();
        printf("%d\n", r);
    }

    return 0;
}
```

15.鼠标控制及高级按键控制

首先，获取鼠标消息：

```
ExMessage m;
m = getmessage();
```

ExMessage 是 EasyX 定义的一个表示消息的结构体类型，以上代码表示用该类型声明了一个变量 m，然后通过 getmessage 函数获取消息，并返回给变量 m。根据 m 的内容，进一步分析获取到的是什么消息。鼠标消息可以通过以下成员获取鼠标消息中的信息：

```
    USHORT message;     // 当前消息
    bool ctrl;          // Ctrl 键是否按下
    bool shift;         // Shift 键是否按下
    bool lbutton;       // 鼠标左键是否按下
    bool mbutton;       // 鼠标中键是否按下
    bool rbutton;       // 鼠标右键是否按下
    int x;              // 当前鼠标 x 坐标
    int y;              // 当前鼠标 y 坐标
    int wheel;          // 鼠标滚轮滚动值
```

其中，“当前消息”可以是以下值：

- WM_MOUSEMOVE　鼠标移动消息
- WM_MOUSEWHEEL　鼠标滚轮拨动消息
- WM_LBUTTONDOWN　左键按下消息
- WM_LBUTTONUP　左键弹起消息
- WM_LBUTTONDBLCLK 左键双击消息
- WM_MBUTTONDOWN　中键按下消息
- WM_MBUTTONUP　中键弹起消息
- WM_MBUTTONDBLCLK 中键双击消息
- WM_RBUTTONDOWN　右键按下消息
- WM_RBUTTONUP　右键弹起消息
- WM_RBUTTONDBLCLK 右键双击消息

例如，判断获取的消息是否是鼠标左键按下，可以用：

```
if (m.message == WM_LBUTTONDOWN)
```

安卓机器人

- 画出来一个安卓机器人的图片。没什么复杂的代码，逻辑很简单，甚至连 if for 都没用到，就是简单的绘图语句堆叠。

安卓机器人

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

const double PI = 3.1415926536;

int main()
{
    // 创建大小为 800 × 600 的绘图窗口
    initgraph(800, 600);

    // 设置原点 (0, 0) 为屏幕中央（Y轴默认向下为正）
    setorigin(400, 300);

    // 使用藏青色填充背景
    setbkcolor(0x7c5731);
    cleardevice();

    // 设置绘图样式
    setlinecolor(WHITE);                          // 设置线条颜色为白色
    setlinestyle(PS_SOLID | PS_ENDCAP_FLAT, 10);  // 设置线条样式为宽度 10 的实线，端点是平的
    setfillcolor(0x24c097);                       // 设置填充颜色为绿色

    // 画身体
    fillroundrect(-75, -111, 75, 39, 36, 36);

    // 画头
    fillpie(-75, -166, 75, -30, 0, PI);           // 脸
    circle(-31, -131, 3);                         // 右眼
    circle( 31, -131, 3);                         // 左眼

    // 画天线
    fillellipse(-52, -181, -38, -167);            // 右天线
    line(-50, -169, -41, -156);
    line(-38, -177, -28, -162);
    fillellipse( 52, -181,  38, -167);            // 左天线
    line( 50, -169,  41, -156);
    line( 38, -177,  28, -162);
    // 用绿色擦掉天线部分多余的线
    setlinecolor(0x24c097);                       // 设置线条颜色为绿色
    setlinestyle(PS_SOLID | PS_ENDCAP_ROUND, 5);  // 设置线条样式为宽度 5 的实线，端点为圆形
    line(-44, -174, -23, -142);                   // 画右天线内部的绿线
    line( 44, -174,  23, -142);                   // 画左天线内部的绿线
    setlinecolor(WHITE);                          // 恢复线条颜色为白色
    setlinestyle(PS_SOLID | PS_ENDCAP_FLAT, 10);  // 恢复线条样式为宽度 10 的实线，端点是平的

    // 画胳膊
    fillroundrect(-117, -99, -75, 7, 42, 42);     // 右胳膊
    fillroundrect( 117, -99,  75, 7, 42, 42);     // 左胳膊

    // 画腿
    fillpie(-50, 49, -8, 91, PI, PI * 2);         // 右腿
    line(-50, 40, -50, 70);
    line(-8, 40, -8, 70);
    solidroundrect(-45, 0, -13, 86, 32, 32);
    fillpie( 50, 49,  8, 91, PI, PI * 2);         // 左腿
    line( 50, 40,  50, 70);
    line( 8, 40,  8, 70);
    solidroundrect( 45, 0,  13, 86, 32, 32);

    // 画字母 A
    arc(-185, 132, -144, 173, PI / 2, PI * 3 / 2);
    line(-165, 132, -135, 132);
    line(-165, 173, -154, 173);
    line(-140, 127, -140, 178);

    // 画字母 N
    arc(-118, 131, -78, 171, 0, PI);
    line(-118, 151, -118, 178);
    line(-78, 151, -78, 178);

    // 画字母 D
    arc(-57, 132, -16, 173, PI * 3 / 2, PI / 2);
    line(-60, 132, -37, 132);
    line(-60, 173, -37, 173);

    // 画字母 R
    arc(14, 132, 40, 158, PI * 3 / 2, PI / 2);
    arc(-2, 158, 38, 198, 0, PI / 2);
    line(1, 132, 27, 132);
    line(1, 158, 27, 158);

    // 画字母 O
    circle(81, 152, 21);

    // 画字母 I
    line(124, 127, 124, 178);

    // 画字母 D
    arc(144, 132, 185, 173, PI * 3 / 2, PI / 2);
    line(141, 132, 164, 132);
    line(141, 173, 164, 173);

    // 按任意键退出
    _getch();
    closegraph();
    return 0;
}
```

Android.cpp

哆啦A梦

- 这个家伙叫机器猫还是小叮当还是多啦A梦就不管啦
- 学编程最需要注意的就是多动手，多练习。

哆啦A梦

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

const double PI = 3.1415926536;

// 主函数
int main()
{
    // 创建大小为 800 × 600 的绘图窗口
    initgraph(800, 600);

    // 设置原点 (0, 0) 为屏幕中央（Y轴默认向下为正）
    setorigin(400, 300);

    // 使用白色填充背景
    setbkcolor(WHITE);
    cleardevice();


    // 画脸
    setfillcolor(RGB(7, 190, 234));                     // 头
    setlinecolor(BLACK);
    fillroundrect(-135, -206, 135, 54, 248, 248);

    setfillcolor(WHITE);                                // 脸
    fillellipse(-115, -144, 115, 46);

    fillroundrect(-63, -169, 0, -95, 56, 56);           // 右眼
    fillroundrect(0, -169, 63, -95, 56, 56);            // 左眼

    setfillcolor(BLACK);
    solidcircle(-16, -116, 6);                          // 右眼球
    solidcircle(16, -116, 6);                           // 左眼球

    setfillcolor(RGB(201, 62, 0));                      // 鼻子
    fillcircle(0, -92, 15);

    line(0, -77, 0, -4);                                // 人中
    arc(-108, -220, 108, -4, PI * 5 / 4, PI * 7 / 4);   // 嘴

    line(-42, -73, -90, -91);                           // 胡子
    line(42, -73, 90, -91);
    line(-41, -65, -92, -65);
    line(41, -65, 92, -65);
    line(-42, -57, -90, -39);
    line(42, -57, 90, -39);

    // 画身体
    line(-81, 32, -138, 72);                            // 手臂(上)
    line(81, 32, 138, 72);
    line(-96, 96, -116, 110);                           // 手臂(下)
    line(96, 96, 116, 110);

    line(-96, 85, -96, 178);                            // 腿外侧
    line(96, 85, 96, 178);
    arc(-10, 168, 10, 188, 0, PI);                      // 腿内侧

    setfillcolor(WHITE);                                // 手
    fillcircle(-140, 99, 27);
    fillcircle(140, 99, 27);
    fillroundrect(-2, 178, -112, 205, 24, 24);          // 脚
    fillroundrect(2, 178, 112, 205, 24, 24);

    setfillcolor(RGB(7, 190, 234));                     // 身体填充蓝色
    floodfill(0, 100, BLACK);

    setfillcolor(WHITE);                                // 肚皮
    fillcircle(0, 81, 75);
    solidrectangle(-60, 4, 60, 24);                     // 用白色矩形擦掉多余的肚皮

    pie(-58, 23, 58, 139, PI, 0);                       // 口袋


    // 画铃铛
    setfillcolor(RGB(169, 38, 0));                      // 绳子
    fillroundrect(-100, 23, 100, 42, 12, 12);

    setfillcolor(RGB(245, 237, 38));                    // 铃铛外形
    fillcircle(0, 49, 19);

    setfillcolor(BLACK);                                // 铃铛上的洞
    solidellipse(-4, 50, 4, 57);
    setlinestyle(PS_SOLID, 3);
    line(0, 57, 0, 68);

    setlinestyle(PS_SOLID, 1);                          // 铃铛上的纹路
    line(-16, 40, 16, 40);
    line(-18, 44, 18, 44);


    // 按任意键退出
    _getch();
    closegraph();
    return 0;
}
```

Doraemon.cpp

奥特曼

奥特曼的组成

奥特曼是由斜的椭圆，圆角矩形，圆形，以及曲线的组成的。此处绘制中，主要应用了曲线的的绘制，将奥特曼画的比较饱满。

值得学习的地方

本次绘制过过程中，自己编写了两个函数。一个是绘制有倾斜角的椭圆，用来表示奥特曼的眼睛，这样可以使得奥特曼更加有灵魂。另一个是心形。在平时绘制别的东西时，如果需要，可以直接借鉴。

代码分析

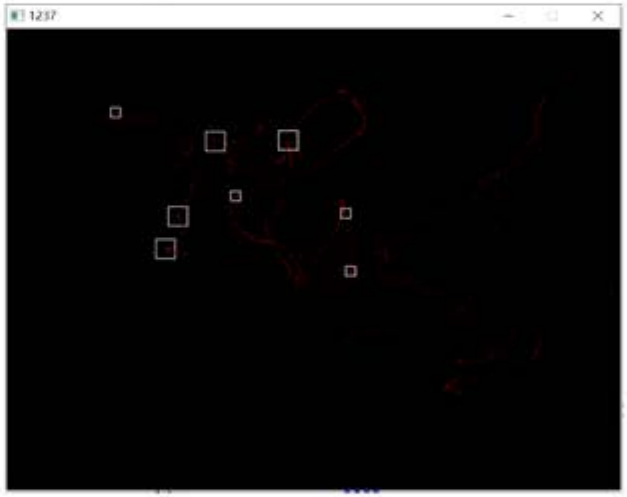
```
#include<conio.h>
#include<graphics.h>
#include<math.h>
#define PI acos(-1.0)
double th = PI / 180;

// 绘制斜的椭圆
void DrawEllipse(int x0, int y0, int a, int b, int k, int color);
// 绘制心形
void heart(int x0, int y0, int size, COLORREF C);

int main()
{
    initgraph(640, 640);
    setbkcolor(WHITE);
    cleardevice();
    // 设置线的宽度
    setlinestyle(PS_SOLID, 5);
    setlinecolor(BLACK);
    setfillcolor(RGB(238, 238, 238));
    // 左耳朵
    fillrectangle(175, 266, 190, 325);
    fillrectangle(159, 281, 175, 315);
    // 右耳朵
    fillrectangle(393, 268, 410, 324);
    fillrectangle(410, 286, 423, 311);
    fillellipse(187, 196, 397, 402);
    setfillcolor(WHITE);
    fillroundrect(288, 146, 302, 242, 10, 20);
    // 绘制左右眼睛
    DrawEllipse(243, 297, 38, 30, -30, BLACK);
    DrawEllipse(350, 297, 38, 30, 30, BLACK);
    setfillcolor(RGB(248, 245, 143));
    floodfill(243, 297, BLACK);
    floodfill(350, 297, BLACK);
    line(296, 422, 249, 394);
    line(296, 422, 336, 394);
    setfillcolor(RGB(235, 110, 69));
    floodfill(295, 410, BLACK);
    setfillcolor(RGB(137, 211, 211));
    fillcircle(294, 432, 10);
    // 绘制身体
    arc(222, 399, 286, 591, 145.0 / 180 * PI, PI + 145.0 / 180 * PI);
    arc(305, 413, 364, 591, PI + 35.0 / 180 * PI, 55.0 / 180 * PI);
    line(224, 485, 359, 485);
    line(224, 511, 278, 549);
    line(278, 549, 312, 549);
    line(312, 549, 360, 515);
    setfillcolor(RGB(235, 110, 69));
    floodfill(294, 517, BLACK);
    setfillcolor(RGB(238, 238, 238));
    floodfill(252, 554, BLACK);
    floodfill(334, 559, BLACK);
    // 绘制左边胳膊
    arc(189, 387, 353, 647, 109.0 / 180 * PI, PI);
    arc(189, 480, 223, 537, 10.0 / 180.0 * PI + PI, 0);
    line(196, 471, 222, 491);
    setfillcolor(RGB(235, 110, 69));
    floodfill(207, 501, BLACK);
    // 绘制右胳膊
    arc(230, 319, 424, 455, 110.0 / 180 * PI + PI, 5.0 / 180 * PI);
    arc(392, 360, 424, 395, -5.0 / 180 * PI, PI + PI / 2);
    arc(310, 286, 402, 394, 70.0 / 180 * PI + PI, 150.0 / 180 * PI + PI);
    line(372, 390, 394, 431);
    setfillcolor(RGB(235, 110, 69));
    floodfill(399, 402, BLACK);
    // 给身体颜色
    setfillcolor(RGB(238, 238, 238));
    floodfill(296, 458, BLACK);
    // 连接气球
    line(463, 187, 422, 365);
    heart(464, 67, 30, BLACK);
    setfillcolor(RGB(235, 110, 69));
    floodfill(464, 70, BLACK);
    setfillcolor(RGB(255, 232, 201));
    solidcircle(508, 70, 6);
    _getch();
    return 0;
}

void heart(int x0, int y0, int size, COLORREF C)
{
    double  m, n, x, y;
    double i;
    for (i = 0; i <= 2 * size; i = i + 0.01)
    {
        // 产生极坐标点
        m = i;
        n = -size * (((sin(i) * sqrt(fabs(cos(i)))) / (sin(i) + 1.4142)) - 2 * sin(i) + 2);
        // 转换为笛卡尔坐标
        x = n * cos(m) + x0;
        y = n * sin(m) + y0;
        setfillcolor(C);
        solidcircle((int)x, (int)y, 2);
    }
}

void DrawEllipse(int x0, int y0, int a, int b, int k, int color)
{
    double i;
    double x, y, tx, ty;
    for (i = -180; i <= 180; i = i + 0.5)
    {
        x = a * cos(i * th);
        y = b * sin(i * th);
        tx = x;
        ty = y;
        x = tx * cos(k * th) - ty * sin(k * th) + x0;
        y = y0 - (ty * cos(k * th) + tx * sin(k * th));
        setfillcolor(color);
        solidcircle((int)x, (int)y, 2);
    }
}
```

ultraman.cpp

15.鼠标控制及高级按键控制

程序会用红色的点标出鼠标移动的轨迹，按左键画一个小方块，按Ctrl+左键画一个大方块，按右键退出。

```
#include <graphics.h>

int main()
{
    // 初始化图形窗口
    initgraph(640, 480);

    ExMessage m;        // 定义消息变量

    while(true)
    {
        // 获取一条鼠标或按键消息
        m = getmessage(EM_MOUSE | EM_KEY);

        switch(m.message)
        {
            case WM_MOUSEMOVE:
                // 鼠标移动的时候画红色的小点
                putpixel(m.x, m.y, RED);
                break;

            case WM_LBUTTONDOWN:
                // 如果点左键的同时按下了 Ctrl 键
                if (m.ctrl)
                    // 画一个大方块
                    rectangle(m.x - 10, m.y - 10, m.x + 10, m.y + 10);
                else
                    // 画一个小方块
                    rectangle(m.x - 5, m.y - 5, m.x + 5, m.y + 5);
                break;

            case WM_KEYDOWN:
                if (m.vkcode == VK_ESCAPE)
                    return 0;    // 按 ESC 键退出程序
        }
    }

    // 关闭图形窗口
    closegraph();
    return 0;
}
```

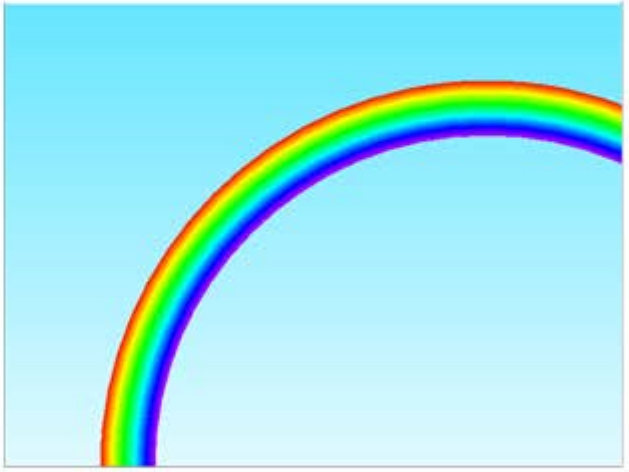


mouse.cpp

彩虹

通过调节 HSL 模型的亮度绘制渐变的天空，调节色相绘制七色彩虹。

```
int main()
{
    // 创建绘图窗口
    initgraph(640, 480);

    // 画渐变的天空(通过亮度逐渐增加)
    float H = 190;       // 色相
    float S = 1;         // 饱和度
    float L = 0.7f;      // 亮度
    for(int y = 0; y < 480; y++)
    {
        L += 0.0005f;
        setlinecolor( HSLtoRGB(H, S, L) );
        line(0, y, 639, y);
    }

    // 画彩虹(通过色相逐渐增加)
    H = 0;
    S = 1;
    L = 0.5f;
    setlinestyle(PS_SOLID, 2);       // 设置线宽为 2
    for(int r = 400; r > 344; r--)
    {
        H += 5;
        setlinecolor( HSLtoRGB(H, S, L) );
        circle(500, 480, r);
    }

    // 按任意键退出
    _getch();
    closegraph();
    return 0;
}
```



rainbow.cpp

作业

- 绘制画作，并用不同的颜色和填充模式填充各个部分。