计算机C/C++语言系列丛书

附范例磁盘

洪锦魁 编著

如何用 Visual C++设计

图形、游戏与动画

学苑出版社

计算机 C/C++语言系列丛书

如何用 *Visual C++*设计图形、游戏与动画

——附完整范例及软盘

洪锦魁　　编著
肖　何　王海潜　改编
文　都　希　望　审校

学苑出版社
1994.

(京)新登字 151 号

内 容 提 要

当您玩赌博性的电子游戏输钱时，不要怪自己运气不好，因为绝大多数电子游戏都是可控制输赢比率的。当您看到市面上许多精彩的电子游戏时，不要望洋兴叹。只要您懂得本书精华，您也可以设计精彩的电子游戏、图形和动画。

本书主要由两大部分组成。第一章主要介绍设计电子游戏、图形和动画的基础知识。第二章主要以 OOP 实例说明游戏、图形和动画程序设置

本书中的所有程序均采用 Visual C/C++编写而成。本书含配套软盘。

本书叙述清晰，使用方便，是 Visual C/C++语言及 OOP 程序设计人员极为有用的工具书，是计算机应用人员和大中专院校师生必备的参考书。

欲购本书的用户，请直接与北京 8721 信箱资料部联系，电话 2562329，邮码 100080。

版 权 声 明

计算机 C/C++语言系列丛书

如何用 Visual C++设计
图形、游戏与动画

编　　著：洪锦魁
改　　编：肖　何　王海潜
审　　校：文　都　希　望
责任编辑：甄国宪
出版发行：学苑出版社　　**邮政编码**：100036
社　　址：北京市海淀区万寿路西街 11 号
印　　刷：双青印刷厂
开　　本：787×1092　1/16
印　　张：21.25　　**字　数**：50 千字
印　　数：1～5000 册
版　　次：1994 年 5 月北京第 1 版第 1 次
ISBN7—5077—0875—6/TP・24
本册定价：57.00 元(含盘)

学苑版图书印、装错误可随时退换

欢迎加入希望用户协会

当今的计算机技术发展迅猛，应用领域繁多，如何准确、有效、地为您提供服务，已成为我们迫切关心的问题。为此，我们决定创立希望用户协会，其宗旨在于加强与用户的联络，了解用户的各种信息与需求，为用户提供更完善更周到的服务。

本协会的成员将定期获得各种软件、资料与培训等讯息，优先得到有关技术服务。请您认真填写后面的会员登记表，其中的信息将用电脑进行管理。

作为美国Borland软件公司在中国的总代理，北京希望电脑公司经Borland公司授权，对其产品dBASE Ⅳ 2.0 进行系统级汉化。另外，对Borland C++ & AF 3.1、 Turbo C++ for DOS 3.0、Turbo C++ for Windows、Turbo C++ Visual Edition for Windows四个产品进行了本地化。预计94年3月推出以上产品。

Borland 软件系列清单：

1. Borland C++ & AF 3.1
2. Borland C++ 3.1
3. Turbo C++ for DOS 3.0
4. Turbo C++ for Windows
5. Paradox for DOS 4.0
6. Paradox for Windows
7. Pdox Engine & Database Framework 3.0
8. Turbo Pascal for DOS 7.0
9. Turbo Pascal for Windows 1.5
10. dBASE Ⅳ 2.0
11. dBASE Ⅳ Compiler
12. Object Vision 2.1

软件与培训

北京希望电脑公司即将与Borland公司密切合作，经销Borland多种软件产品的同时，提供配套的函授与培训服务，以满足用户的不同要求。欢迎用户索取软件与培训资料。

联系方法

有关资料事宜，请与朱红小姐联系，有关软件事宜，请与周东先生联系，有关函授与培训事宜，请与宋明华先生联系。 联系方法如下：

通信地址： 北京 8721 信箱 邮政编码： 100080
联系电话： (01)2562329 (01)2541992 (01)2579874
传真电话： (01)2561057

目　　录

第一章　设计游戏程序的基本技巧

本章介绍几个设计游戏程序的技巧，这些概念有助于读者了解第二章中游戏程序实例的基本设计概念。

1.1　时间延迟的概念

在 Borland C++中存在下列函数：

```
delay (延迟时间);
```

上述函数中的参数"延迟时间"是一个实数。这个函数非常有用，例如，若希望计算机暂停工作 2 秒，则可执行下列函数调用：

```
delay (2.0);
```

但是，Visual C++并没有提供该函数，因此，我们必须自己设计该函数。设计此函数时要使用下列函数：

```
clock_t clock ( );          // 在 time.h 中定义
```

上述 clock( )函数将返回从程序开始运行到当前为止所花的 tick(一种时间单位)数。将 tick 数除以 CLOCKS_PER_SEC，就可以得到秒数（数据类型为实数）。有了上述概念之后，读者就可以自己设计下列时间延迟函数 delay( )。

```
//------------------------------------------//
// 时间延迟，参数单位是秒                   //
//------------------------------------------//
void delay (secs)
float secs;
{
    clock_t end;
    end = (int) (secs * CLOCKS_PER_SEC) + clock( );
    while ( clock( ) < end)
        ;
}
```

程序实例 ch1_1.c

每隔 1 秒打印一次某个字符串，连续打印 5 次。

```
//------------------------------------------
// Program Title: ch1_1.c
// 每隔 1 秒输出字符串一次
//------------------------------------------
#include <time.h>
#include <stdio.h>
//------------------------------------------
// 时间延迟，参数单位是秒
//------------------------------------------
```

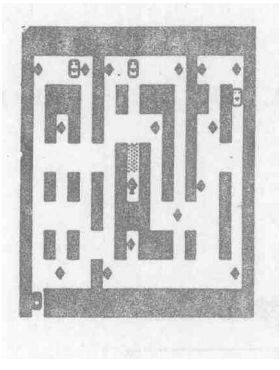
```
void delay(secs)
float secs;
{
    clock_t end;
    end = (int) (secs * CLOCKS_PER_SEC) + clock();
    while ( clock() < end )
        ;
}
void main()
{
    int i;
    for ( i = 0; i < 5; i++ )
    {
        printf("The KWEI Group\n");
        delay(1.0);
    }
}
```

运行结果

```
The KWEI Group
The KWEI Group
The KWEI Group
The KWEI Group
The KWEI Group
```

1.2 计算游戏所花的时间

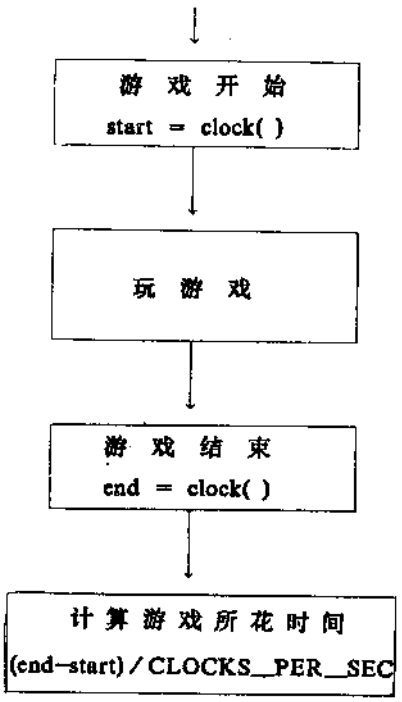
许多游戏都需要计算到底花了多少时间才达到目的。例如，下述迷宫寻宝游戏就是如此。



Press any key. The time will start running.

在上述游戏中，不仅要找到宝藏，同时要使所花的时间最少。若要处理这类问题，则

可使用 clock( )函数．计算整个游戏所花时间的流程如下：



程序实例 ch1_2.c

计算打印某个字符串所花的时间．值得注意的是，在不同的计算机上运行本程序可能会得到不同的结果．

```
//-----------------------------------------
// Program Title: ch1_2.c
// 输出循环所花的时间
//-----------------------------------------
#include <time.h>
#include <stdio.h>
void main()
{
    int i;
    clock_t start, end;
    double var;

    start = clock();
    for ( i = 0; i < 10; i++ )
       printf("Video Game Design using C, By Jiin Kwei Hung\n");
    end = clock();
    var = (double) ( end - start ) / CLOCKS_PER_SEC;
    printf("It take you %6.4f second",var);
}
```

运行结果

```
Video Game Design using C, By Jiin Kwei Hung
Video Game Design using C, By Jiin Kwei Hung
Video Game Design using C, By Jiin Kwei Hung
Video Game Design using C, By Jiin Kwei Hung
Video Game Design using C, By Jiin Kwei Hung
Video Game Design using C, By Jiin Kwei Hung
Video Game Design using C, By Jiin Kwei Hung
Video Game Design using C, By Jiin Kwei Hung
Video Game Design using C, By Jiin Kwei Hung
Video Game Design using C, By Jiin Kwei Hung
It take you 0.0600 second
```

1.3 声音的产生

如果游戏程序没有声音，那将"大剥风景"。本小节将介绍产生声音的原理与技巧。

在 Borland C++中存在下列函数:

```
sound (频率值);
```

执行上述函数时将产生指定频率的声音，但是，由于 Visual C++并不提供该函数，所以我们只能自己设计此函数。

在 IBM PC 内我们可以利用计时器 (timer) 发出声音，这种方法的最大特点是，当微处理器发出某种声音后，即使用户继续从事其它工作，声音仍将持续。而且，要改成不同频率的声音，只能重新设置计时器。

在 IBM PC 机中，计时器其实是一个芯片内的一小部分功能。在 8086，8088 和 80286 内，这个芯片名为 8253。在 80386 和 80486 内，这个芯片名为 82380。事实上，PC 中有三个计时器:

(1) 计时器 0: 在 DMA (Direct Memory Access)数据传送时使用。

(2) 计时器 1: 作为系统时钟使用。

(3) 计时器 2: 连接喇叭，这也是本节的重点。

利用计时器产生声音的步骤如下:

(1) 确定计时器2的初始状态，如下所示:

```
__outp (0x43, 0xb6);      // 将0xb6输出到0x43端口
```

至于为什么要将0xb6输出到0x43端口，这个问题很复杂，不在本书的讨论范围内，读者可参考 BIOS 方面的书籍。

(2) 输入一个16位数值到计时器2，以便建立所要产生的音调频率。方法如下:

```
data = 1193180 / 频率;
```

然后依次将高位字节和低位字节所求得的结果(data)放入0x42端口上。

(3) 打开输出端口0x61，使喇叭产生声音:

```
portdata = __inp (0x61);                  // 读取0x61端口的值
__outp (0x61, portdata | 0x3);            // 使末2位为1，输出到0x61口
```

有了以上概念之后，就可以设计下列函数，以产生声音：

```
//————————————————————————
// 声音函数，参数是频率
//————————————————————————
void sound (unsigned freq)
{
    int portdata;

    __outp (0x43, 0xb6);
    freq = (unsigned) (1193180L / freq);
    __outp (0x42, (char) freq);
    __outp (0x42, (char) (freq >> 8));
    portdata = __inp (0x61);
    __outp (0x61, portvalue | 0x3);
}
```

若想关闭声音，只要恢复原 0x61 端口的值即可．因此，可以设计关闭声音的函数 nosound( )：

```
//————————————————————————
// 关闭声音
//————————————————————————
void nosound ( )
{
    __outp (0x61, portvalue);
}
```

上述声音函数的最大缺点是，若连续调用两次 sound( )函数，以后将无法正常关闭声音：

```
sound (500);          // 产生频率为500的声音
nosound ( );          // 可正常关闭声音
```

另一个例子如下：

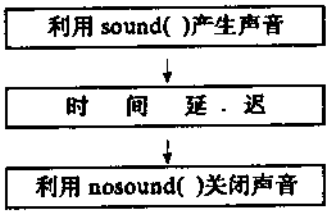
```
sound (500);          // 产生频率为500的声音
sound (300);          // 产生频率为300的声音
nosound ( );          // 无法正常关闭声音
```

上述无法正常关闭声音的最大原因是，在连续调用 sound( )函数后．将无法恢复原来的 0x61 端口的值．若要正常产生及关闭声音，可使用下述修改后的 sound( )函数：

```
static int portvalue = 0;      // 用于恢复原端口的值0x61
//————————————————————————
// 声音函数，参数是频率
//————————————————————————
void sound (unsigned freq)
{
    int portdata;

    __outp (0x43, 0xb6);
    freq = (unsigned) (1193180L / freq);
```

```
    __outp (0x42, (char) freq);
    __outp (0x42, (char) (freq >> 8));
    portdata = __inp (0x61);
    if (portvalue == 0)
        portvalue = portdata;
    __outp (0x61, portvalue | 0x3);
}

//-----------------------------------
// 关闭声音
//-----------------------------------
void nosound ( )
{
    __outp (0x61, portvalue);
}
```

有了上述概念后，就可以利用下列流程产生某时间段某频率的声音：



程序实例 ch1__3.c

产生频率为 500 的声音，持续 3 秒钟。

```
//-----------------------------------
//  Program Title: ch1__3.c
//  产生频率为 500 的声音 3 秒
//-----------------------------------
#include <time.h>
#include <conio.h>
//-----------------------------------
// 时间延迟，参数单位是秒
//-----------------------------------
void delay(secs)
float secs;
{
    clock__t end;
    end = (int) (secs * CLOCKS__PER__SEC) + clock();
    while ( clock() < end )
        ;
}
static int portvalue = 0;    // 用于恢复 0x61 PORT 值
//-----------------------------------
// 声音函数，参数是频率
//-----------------------------------
void sound(unsigned freq)
{
```

```
        int portdata;

        __outp(0x43,0xb6);
        freq = (unsigned) (1193180L / freq);
        __outp(0x42, (char)freq);
        __outp(0x42, (char)(freq >> 8));
        portdata = __inp(0x61);
        if ( portvalue == 0 )
           portvalue = portdata;
        __outp(0x61,portvalue | 0x3);
    }
    // ------------------------------------
    // 关闭声音
    // ------------------------------------
    void nosound()
    {
        __outp(0x61,portvalue);
    }
    void main()
    {
        sound(500);
        delay(3.0);
        nosound();
    }
```

程序实例 ch1__4.c

救护车声音程序设计。

```
    // ------------------------------------
    //  Program Title: ch1__4.c
    //  救护车声音
    // ------------------------------------
    #include <time.h>
    #include <conio.h>
    // ------------------------------------
    // 时间延迟，参数单位是秒
    // ------------------------------------
    void delay(secs)
    float secs;
    {
        clock__t end;
        end = (int) (secs * CLOCKS__PER__SEC) + clock();
        while ( clock() < end )
           ;
    }
    static int portvalue = 0;      // 用于恢复 0x61 PORT 值
    // ------------------------------------
    // 声音函数，参数是频率
    // ------------------------------------
    void sound(unsigned freq)
```

```
{
    int portdata;

    __outp(0x43,0xb6);
    freq = (unsigned) (1193180L / freq);
    __outp(0x42, (char)freq);
    __outp(0x42, (char)(freq >> 8));
    portdata = __inp(0x61);
    if ( portvalue == 0 )
       portvalue = portdata;
    __outp(0x61,portvalue | 0x3);
}
// ---------------------------------------
// 关闭声音
// ---------------------------------------
void nosound()
{
    __outp(0x61,portvalue);
}
void main()
{
    int i;
    for ( i = 0; i < 5; i++ )
    {
       sound(900);
       delay(0.3);
       sound(300);
       delay(0.3);
    }
    nosound();
}
```

1.4 美妙的音乐

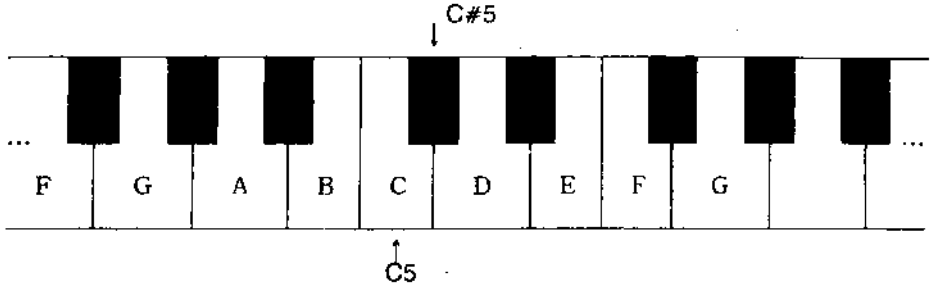
要想产生音乐，首先必须了解音符和频率的关系对照表，下面是其对照表。

音符	频率	音符	频率	音符	频率	音符	频率
C2	65	A3	220	F#5	740	C7	2489
C#2	69	A#3	233	G5	787	C#7	2637
D2	73	B3	247	G#5	831	D7	2794
D#2	78	C4	262	A5	880	D#7	2960
E2	82	C#4	277	A#5	932	E7	3136
F2	87	D4	294	B5	988	F7	3322
F#2	93	D#4	311	C6	1047	F#7	3520
G2	98	E4	330	C#6	1109	G7	3729
G#2	104	F4	349	D6	1175	G#7	3951
A2	110	F#4	370	D#6	1245	A7	4186
A#2	116	G4	392	E6	1319	A#7	4435

(续表)

音符	频率	音符	频率	音符	频率	音符	频率
B2	123	G#4	415	F6	1397	B7	4699
C3	131	A4	440	F#6	1480	C8	4978
C#3	139	A#4	466	G6	1568	C#8	5274
D3	147	B4	494	G#6	1661	D8	5587
D#3	156	C5	523	A6	1760	F#8	5919
E3	165	C#5	554	A#6	1865	G8	6271
F3	175	D5	587	B6	1976	G#8	6645
F#3	185	D#5	622	C6	2093	A8	7040
G3	196	E5	659	D6	2217	A#8	7459

通常，读者所看到的音乐键盘格式如下：



也就是说，在整个键盘中会有许多不同的音阶，而其中最适合人类发出的声音是第 5 音阶，这个音阶在钢琴键的中间位置。在音乐和频率的对照表中，C5 就代表第 5 音阶的 C 音，D7 就代表第 7 音阶的 D 音，其它的依此类推。

音符和音乐代号的关系如下：

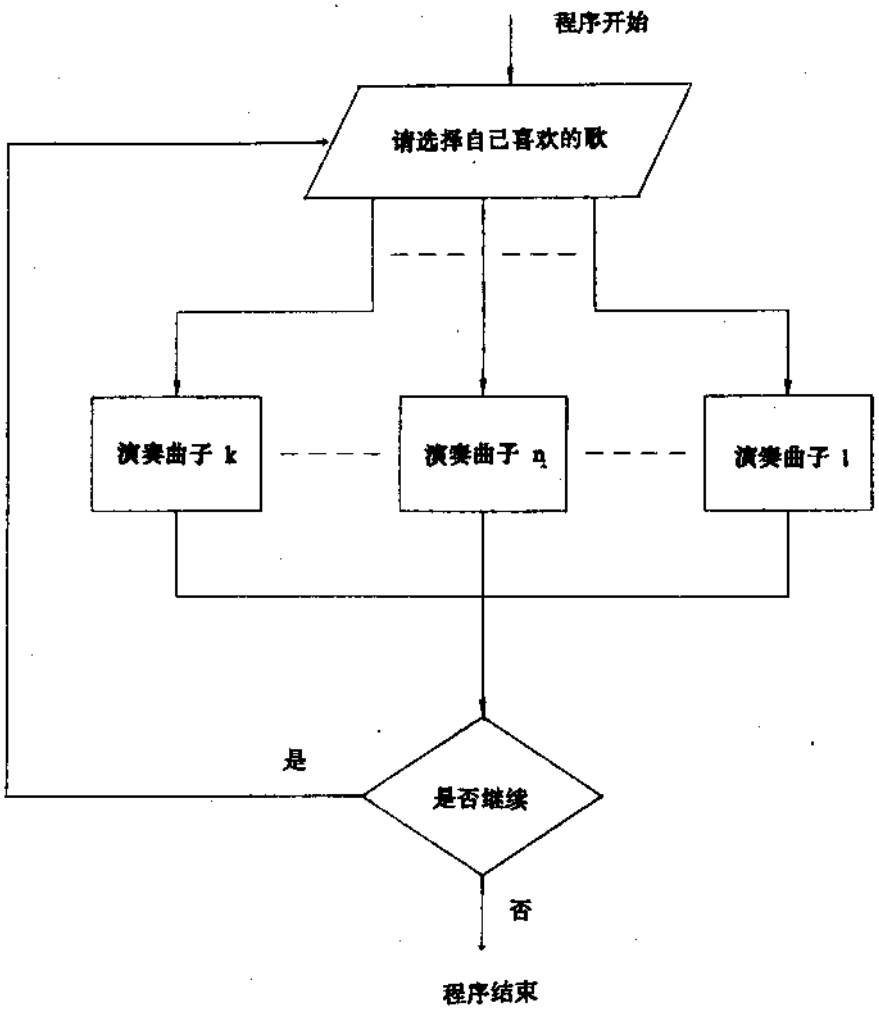
音符	音乐名称	音乐代号
C	Do	1
D	Re	2
E	Mi	3
F	Fa	4
G	So	5
A	La	6
B	Si	7

另外，要想产生音乐，我们还必须有节拍的概念。通常，音乐演奏速度与节拍的关系如下：

速度	节　拍
很慢	30–50
慢	60–80
中等	110–130
快	140–160
很快	160 以上

程序实例 ch1_5.c

都市夜生活中，进入卡啦 OK 厅，选择自己喜欢的歌曲，然后跟着音乐节奏唱首歌，这是一件很浪漫的事。下面就用流程说明，应如何使用计算机设计卡啦 OK 程序。



有关上述流程可参考 game9.c。

下面我们就来设计"两只老虎"音乐程序，此儿歌曲谱如下：

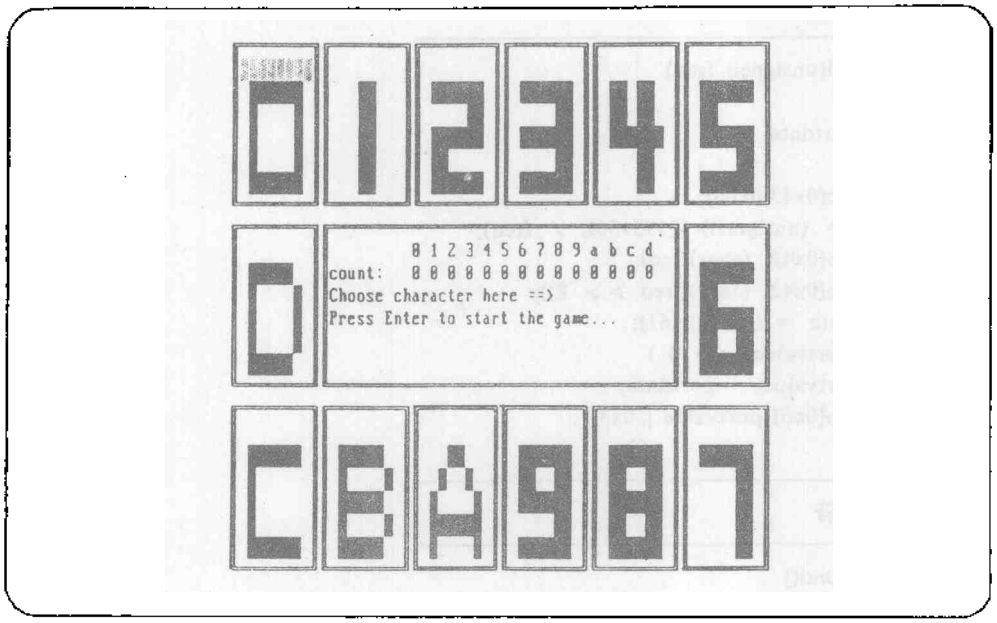
1 2 3 1	1 2 3 1	3 4 5 —	3 4 5 —
56 54 3 1	56 54 3 1	1 5 1 —	1 5 1 —

```
// ------------------------------------
//  Program Title: ch1_5.c
//  两只老虎音乐
// ------------------------------------
#include <conio.h>
#include <time.h>
// ------------------------------------
// 时间延迟, 参数单位是秒
// ------------------------------------
void delay(secs)
float secs;
{
    clock_t end;
    end = (int) (secs * CLOCKS_PER_SEC) + clock();
    while ( clock() < end )
        ;
}
static int portvalue = 0;    // 用于恢复 0x61 PORT 值
// ------------------------------------
// 声音函数, 参数是频率
// ------------------------------------
void sound(unsigned freq)
{
    int portdata;

    _outp(0x43,0xb6);
    freq = (unsigned) (1193180L / freq);
    _outp(0x42, (char)freq);
    _outp(0x42, (char)(freq >> 8));
    portdata = _inp(0x61);
    if ( portvalue == 0 )
        portvalue = portdata;
    _outp(0x61,portvalue | 0x3);
}
// ------------------------------------
// 关闭声音
// ------------------------------------
void nosound()
{
    _outp(0x61,portvalue);
}
void main()
{
    unsigned freq[] = { 523, 587, 659, 523, 523, 587, 659, 523,
                        659, 698, 784,   0, 659, 698, 784,   0,
                        784, 880, 784, 698, 659, 523,
                        784, 880, 784, 698, 659, 523,
                        523, 392, 523,   0, 523, 392, 523,   0 };
    double duration[] = { 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
                          0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
```

```
                        0.25,0.25,0.25,0.25,0.5, 0.5,
                        0.25,0.25,0.25,0.25,0.5, 0.5,
                        0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5 };
    int i;

    for ( i = 0; i < 36; i++ )
    {
        if ( freq[i] > 0 )
            sound(freq[i]);
        delay(duration[i]);
    }
    nosound();
}
```

1.5 随机数

假设读者想设计一个吃角子老虎程序，此吃角子老虎程序的屏幕如下所示：



如果读者不想设计一个会欺骗顾客的程序，那么最好的方法就是让吃角子老虎的长滚动条(当前停留在位置 0)能平均地落在各方块的小窗口内，也就是说，0，1，2，3，4，5，6，7，8，9，A，B，C，D 各小窗口的概率相同。

Visual C++ 提供了一个随机数函数，可让我们很轻易地实现上述目的：

```
int rand( );
```

上述函数产生一个随机数，它的值平均分布在 0 到 RAND_MAX (32767) 之间。由于吃角子老虎的小窗口只有 14 格，因此我们还必须利用一个求余数的指令，以便计算长滚动条应停留在哪一位置，如下所示：