

**程序设计实习报告**

**专业班级： 计算机科学与技术1403班**

**姓 名: 张斌**

**学 号： 1407010319**

**小组成员：张斌 赵智博 张涛**

**2015年7月13日**

目录

[1、需求分析 2](#_Toc424576585)

[2、总体设计 2](#_Toc424576586)

[2.1 需求规定 2](#_Toc424576587)

[2.3基本设计概念和处理流程 2](#_Toc424576588)

[2.3.1基本模块 2](#_Toc424576589)

[2.3.2处理流程 2](#_Toc424576590)

[2.4数据结构设计 2](#_Toc424576591)

[3、详细设计 2](#_Toc424576592)

[数据存储模块 2](#_Toc424576593)

[界面初始化模块 2](#_Toc424576594)

[用户输入模块 2](#_Toc424576595)

[用户操作响应模块 2](#_Toc424576596)

[音效控制模块 2](#_Toc424576597)

[雨点绘制模块 2](#_Toc424576598)

[雨点控制模块 2](#_Toc424576599)

[计时器模块 2](#_Toc424576600)

[4、编码 2](#_Toc424576601)

[数据存储模块：数据存储与读写 2](#_Toc424576602)

[界面初始化模块：全局变量的初始化，窗口绘制 2](#_Toc424576603)

[用户输入模块：读取用户输入，传递输入数据 2](#_Toc424576604)

[用户操作响应模块：对输入数据进行处理，触发响应模块 2](#_Toc424576605)

[音效控制模块：控制音效的播放与停止 2](#_Toc424576606)

[雨点初始化、绘制模块：初始化雨点并绘制 2](#_Toc424576607)

[雨点控制模块：雨点的移动 2](#_Toc424576608)

[计时器模块：显示倒计时 2](#_Toc424576609)

[5 综合测试 2](#_Toc424576610)

[5.1下雨效果模拟1.0 版本测试 2](#_Toc424576611)

[5.1.1 经测试水波未擦出干净 2](#_Toc424576612)

[5.1.3手动输入雨量大小（2~5） 2](#_Toc424576613)

[5.1.4下雨效果显示 2](#_Toc424576614)

[5.2下雨效果模拟1.1 版本测试 2](#_Toc424576615)

[5.3 下雨效果模拟1.2版本测试 2](#_Toc424576616)

[5.3.1根据题目要求，加入多线程。 2](#_Toc424576617)

[5.3.2 运行程序 2](#_Toc424576618)

[5.3.3 错误原因 2](#_Toc424576619)

[5.4 下雨模拟效果2.0版本 2](#_Toc424576620)

[5.4.1 加入一滴雨，两滴雨效果 2](#_Toc424576621)

[5.4.2 效果显示 2](#_Toc424576622)

[5.5 下雨效果模拟2.1版本 2](#_Toc424576623)

[5.6 下雨效果模拟2.2版本 2](#_Toc424576624)

[5.7 下雨效果模拟2.2.1版本 2](#_Toc424576625)

[5.8 下雨效果模拟2.3版本 2](#_Toc424576626)

[5.9 下雨效果模拟2.3.1版本 2](#_Toc424576627)

[5.10 下雨效果模拟2.4版本 2](#_Toc424576628)

[5.11 下雨效果模拟2.4.1 2](#_Toc424576629)

[附录： 2](#_Toc424576630)

[模拟下雨效果 v3.0Final 2](#_Toc424576631)

# 1、需求分析

近年来，伴随着计算机图形学的迅速发展及计算机仿真在工业设计、过程控制、虚拟制造等方面的大量运用，掌握通过建立某一过程或某一系统，来描述一个现象的能力对当代计算机专业的学生显得尤为重要，而用计算机模拟下雨作为计算机图形学上的一个经典问题，既融合了图形学的一些基础知识，也涵盖了大量编程技巧，对编程能力有较高要求。据我们查阅资料了解，以往用计算机模拟下雨的案例或是融合大量计算机图形学专业知识，大一学生不具备如此高的专业素养，从中学习难度颇大，或是仅利用C/C++程序设计语言刻画出雨滴下落过程，其余自然特征表现力度不够，达不到计算机模拟下雨场景使人产生身临其境感觉的目的。针对这一现象，我们小组在本次程序设计实习中选择了计算机模拟下雨效果这一课题，力求达到以下效果：

1. 有合适的图形显示界面，包括大小合适的程序运行窗口，简洁但丰富的画面背景；

2. 雨的轨迹采用一小段线段表示，当雨点（线段）到达地面时，地面有水圈扩散至一定半径后消失；

3. 雨要一直下，雨点的产生、雨点位置、下落时间随机变化，不由程序员控制；

4. 雨量大小可调节，程序运行前应有窗口提示程序员手动设置雨量大小 ；

5. 可单独展示一滴雨下落的整个过程，亦可展示大量雨滴下落的过程，两个过程转换期间应有提示信息给予观赏者明确提示；

6. 整个模拟过程不仅包含雨滴下落过程，还应包含闪电、雷声、雨声、雨过天晴的画面转换、天晴后的阳光、鸟鸣等丰富的自然物理现象，使模拟的情景生动、逼真，富有感染力；

7. 下雨过程在运行期间可由观赏者自由决定是否结束下雨；

8. 程序运行至天晴画面后可自动关闭运行窗口，并提前给予观赏者提示信息；

# 2、总体设计

## 2.1 需求规定

本程序用于模拟真实下雨场，其需求如下:

1. 雨点产生：位置随机，在屏幕上方，速度在一定范围内随机。
2. 雨点下落：下落时有加速度存在，雨点下落速度逐渐加快。
3. 雨点落地：落地位置随机在北京中马路的范围中。
4. 水圈产生：水圈产生在雨点落地的位置，其半径逐渐由小变大，最后消失。
5. 音效：音乐的开始与停止根据场景的变化自动控制。
6. 雨量：雨量的大小包括雨点的数量、雨点下落的速度，可由用户自己输入控制。

2.2 运行环境

1. 硬件环境： PC机 CPU：Intel 赛扬 1.1GHz及以上 内存：512MB及以上
2. 支持环境： Windows XP/Windows vista/Windows 7/Windows 8/Windows 8.1/Windows 10

Linux6.0及以上版本/Mac OS X v10.0及以上版本

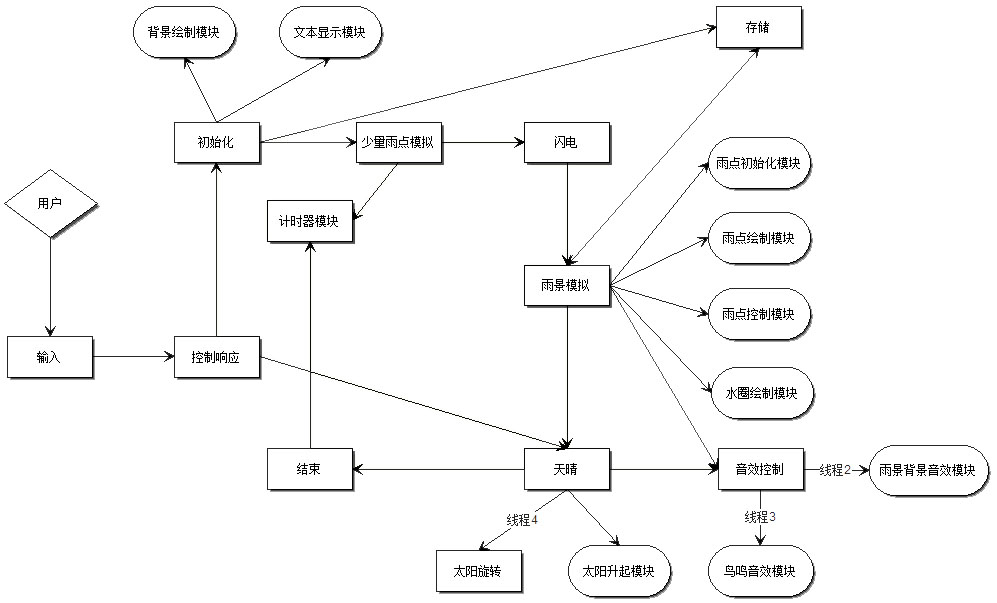
需包含C/C++ graphics图形库

1. 编写环境： Microsoft Visual Studio 2013

## 2.3基本设计概念和处理流程

### 2.3.1基本模块

1. 数据存储模块：数据存储与读写
2. 界面初始化模块：全局变量的初始化，窗口绘制
3. 用户输入模块：读取用户输入，传递输入数据
4. 用户操作响应模块：对输入数据进行处理，触发响应模块
5. 音效控制模块：控制音效的播放与停止
6. 雨点控制模块：雨点的移动
7. 太阳控制模块：太阳的升起与发光的运动
8. 少量雨点模拟模块：模拟少量雨点下落过程
9. 计时器模块：显示倒计时
10. 背景绘制模块：绘制背景
11. 文本绘制模块：绘制文本
12. 水圈绘制模块：水圈产生与消失过程的绘制
13. 闪电绘制模块：绘制闪电效果
14. 雨点绘制模块：雨点初始化与绘制
15. 太阳绘制模块：绘制太阳图案

2.3.2处理流程

## 2.4数据结构设计

使用结构体存储雨点的位置坐标与速度，利用数组存储全部雨点结构体。以此方便访问指定雨点的位置速

# 3、详细设计

### 数据存储模块

为同时记录雨点的x、y坐标和速度v，创建名为RAIN结构体，包含两个整型值x、y，浮点值v；将每个雨点的结构体存储于数组中，访问时直接通过数组下标访问每个结构体变量，快速的得到雨点的x、y坐标和速度。

设置全局变量：雨点总数，雨点初始随即位置相关参量，雨点初始下落随机速度相关参量。

***雨点总数 = (read\_in[0] - '0') \* 80;***

***雨点初始下落随机速度相关参量= ((用户输入 - '0') + 3)\*0.8;***

### 界面初始化模块

根据用户输入数据对全局变量进行初始化；设置随机种子（为更加真切的模拟雨点随机下落，程序中多次对随机种子进行设置，设置的参数与系统时间和程序中的变量均有关系）:

***void srand (unsigned int seed );***

***#include <time.h>***

***Time(NULL) 返回系统时间***

***利用graphics中函数在屏幕绘制800\*600大小的窗口:***

***HWND initgraph( int width, int height, int flag = NULL);***

***设置绘图线条颜色：***

***void setfillcolor(COLORREF color);***

***设置绘图填充颜色:***

***void setlinecolor(COLORREF color);***

### 用户输入模块

在程序开始时，控制台窗口中让用户输入雨量大小。输入数据应为一个整数，且大于等于2小于等于5。对于不合法的输入，重新提示用户输入。

不合法输入情况：

1. 用户输入一个字符，字符为数字，但小于2或大于5 。
2. 用户输入一个字符，字符不为数字。
3. 用户输入多个字符。

针对以上三种情况，采用字符数组来存储用户输入，并判断字符数组是否符合下述条件：

1. 数组第一个字符大于等于‘2’且小于等于‘5’;
2. 数组第二个字符为‘\0’。

***（read\_in[0] >= '2' && read\_in[0] <= '5'&&read\_in[1] == '\0'）***

若符合此条件则为合法输入。

### 用户操作响应模块

通过检测在某阶段中有无数据读入与键盘输入来执行相关函数。

1. 在用户输入模块中，若得到了合法的输入，运行界面初始化模块，对相关设置、变量进行初始化。
2. 在雨景模拟过程中，若有键盘输入，则会停止雨景模拟，清除屏幕上的雨点及水圈，进入天晴的场景，运行相关模块。

***#include <conio.h>***

***int \_kbhit( void );***

如果键盘按下了，\_kbhit返回一个非零值。否则返回0。

1. 若用户操作不位于用户输入和雨景模拟过程中，则不会对当前场景产生影响。

### 音效控制模块

利用c++的多线程机制，创建新线程并利用Media Control Interface提供的接口函数播放音效，由于播放音效的进程与主函数进程使用不同资源，所以不会产生干扰，更能提高程序效率。音效播放时间及内容：

1. 闪电绘制过程中，播放下雨及雷声音效（Rainymood.wav）。
2. 太阳升起过程中，播放鸟鸣音效（鸟鸣声.wav）。
3. ***static HANDLE CreateThread( LPSECURITY\_ATTRIBUTES lpsa,***

***DWORD dwStackSize, LPTHREAD\_START\_ROUTINE pfnThreadProc,***

***void\* pvParam, DWORD dwCreationFlags, DWORD\* pdwThreadId )throw( ); //创建线程***

*lpsa 新线程的安全特性。*

*dwStackSize 新线程的堆栈大小。*

*pfnThreadProc 新线程的线程过程。*

*pvParam 将传递的参数传递给线程过程。*

*dwCreationFlags 创建标志(0个或CREATE\_SUSPENDED）。*

*pdwThreadId [out] 中，若成功，接收新创建的线程的线程ID DWORD变量的地址。*

1. ***DWORD WINAPI MyThread(PVOID pvParam){***

***//加入创建的线程代码***

***return 0;***

***} //定义线程***

1. ***#include <mmsystem.h> //导入声音头文件***
2. ***#pragma comment(lib, "WINMM.LIB") //导入声音头文件库***
3. ***BOOL PlaySound(LPCSTR pszSound, HMODULE hwnd,DWORD fdwSound); //播放指定路径中wav文件***

*pszSound是指定了要播放声音的字符串。如果该参数为NULL则停止 正在播放的声音。*

*hwnd是应用程序的实例句柄。*

*fdwSound是标志的组合，如下表所示。若成功则函数返回TRUE，否 则返回FALSE。*

### 雨点绘制模块

利用graphics中函数将每个雨点均绘制为矩形，宽度为1像素，高度为4像素。

***void rectangle( int left, int top, int right, int bottom); //绘制矩形***

*left矩形左部x坐标。*

*top矩形上部y坐标。*

*right矩形右部x坐标。*

*bottom矩形下部y坐标。*

### 雨点控制模块

雨点下落过程为对每个雨点位置的刷新，即雨点的消除与重绘。

1. 第一次初始每个雨点时，其随机y坐标在0-370（雨点初始随即位置相关参量）像素之间（即窗口上边缘到马路上边缘之间），这样使雨点第一次出现时在屏幕上分布均匀。并将雨点初始随即位置相关参量设置为30，使下次雨点初始化时的起始位置在0-30像素范围中。

***雨点y坐标值=rand() % 370 //产生随机雨点y坐标值在0-369之间***

1. 循环遍历雨点结构体数组，得到雨点的坐标值，消除此坐标上的雨点。

***void clearrectangle( int left, int top, int right,int bottom);***

***//消除矩形区域***

*left矩形左部x坐标。*

*top矩形上部y坐标。*

*right矩形右部x坐标。*

*bottom矩形下部y坐标。*

1. 根据雨点速度值计算出雨点下一次出现的新坐标值，在新坐标值上重绘雨点。

雨点y坐标值=雨点y坐标值+雨点速度

1. 根据设定的加速度，对雨点的速度值进行更新。

雨点速度=雨点速度+加速度

1. 若雨点的y坐标大于370-600像素中的某一随机值（即位与背景的马路上下边缘之间），消除此雨点并运行水圈绘制函数，传递参数为此雨点结构体在数组中的下标。

***(雨点y坐标值 > rand() % 230 + 380) //判断条件***

1. 对于运行完水圈绘制函数的雨点，重新将其初始化，随机y坐标在0-30像素之间（即屏幕上缘天空范围）。

***雨点y坐标值=rand() % 370 //产生雨点y坐标值在0-369之间***

### 计时器模块

倒计时的动画出现在闪电绘制之前和窗口绘制之前。具体过程：

1. 初始计数变量，赋初值为5（倒计时6个数0-5）。
2. 利用循环语句，每次循环对显示的数字进行刷新，共循环6次。
3. 刷新数字时，先在此位置绘制数字（计数变量）的文本，然后挂起线程1200ms，保持文本显示，再消除此数字所在矩形区域，更新计数变量，下次循环时在此位置绘制新的数字文本。
4. ***char s[] = ""; //指定输出文本***
5. ***int sprintf( char \*buffer, const char \*format [ argument] ... ); // 向字符串写入设置数据格式***

*buffer 输出的存储位置*

*count 存储的最大字符数此函数的 Unicode 版本。*

*format 窗体控件字符串*

*argument 可选参数*

*locale 使用的区域设置。*

1. ***void outtextxy( int x, int y, LPCTSTR str);***

***//在指定位置输出指定字符串文本***

*字符串输出时头字母的 x 轴的坐标值。*

*字符串输出时头字母的 y 轴的坐标值。*

*待输出的字符串的指针。*

*待输出的字符。*

1. 当数字文本“0”所在矩形区域被消除后跳出循环，执行下一行程序。

# 4、编码

### 数据存储模块：数据存储与读写

/\*\*\*\*\*\*\*\*\*雨点结构体\*\*\*\*\*\*\*\*\*/

struct RAIN

{

int x; //雨点x坐标

int y; //雨点y坐标

double step; //雨点速度

};

RAIN rain[600]; //初始化雨点结构体数组

/\*\*\*\*\*\*\*\*\*全局变量\*\*\*\*\*\*\*\*\*/

int maxrain; // 雨点总数

int first = 370; //雨点初始随机位置相关参量

double v; //雨点初始下落随机速度相关参量

### 界面初始化模块：全局变量的初始化，窗口绘制

/\*\*\*\*\*\*\*\*\*初始化\*\*\*\*\*\*\*\*\*/

maxrain = (read\_in[0] - '0') \* 80; //对雨点数量初始化

v = ((read\_in[0] - '0') + 3)\*0.8; //对雨点随机速度参量初始化

srand((unsigned)time(NULL)); // 随机种子

initgraph(800, 600); // 创建窗口

setbkcolor(WHITE); //设置窗口背景为白色

cleardevice();

setfillcolor(BLACK); //设置雨点填充颜色为黑色

setlinecolor(BLACK); //设置线条颜色为黑色

background(); //绘制背景

### 用户输入模块：读取用户输入，传递输入数据

char read\_in[100];

while (1) //雨量大小选择

{

cout << "请输入雨量大小(2-5): ";

cin >> read\_in;

if (read\_in[0] >= '2' && read\_in[0] <= '5'&&read\_in[1] == '\0')

{

break;

}

}

### 用户操作响应模块：对输入数据进行处理，触发响应模块

while (!kbhit())

{

for (int i = 0; i < maxrain; i++)

MoveRain(i); //遍历雨点，更新其位置

background(); //对背景进行重绘

settextstyle(30, 19, \_T("楷体"));

TCHAR s[] = \_T("按下键盘任意键结束");

outtextxy(285, 180, s);

Sleep(10);

}

### 音效控制模块：控制音效的播放与停止

/\*\*\*\*\*\*\*\*\*线程-背景音乐\*\*\*\*\*\*\*\*\*/

DWORD WINAPI MyThread3(PVOID pvParam)

{

PlaySound(TEXT("F:\\my document\\SmallTerm\\28\\test2\\Rainymood.wav"), NULL, SND\_LOOP | SND\_ASYNC);

return 0;

}

DWORD WINAPI MyThread2(PVOID pvParam)

{

PlaySound(TEXT("F:\\my document\\SmallTerm\\28\\test2\\鸟鸣 声.wav"), NULL, SND\_LOOP | SND\_ASYNC);

return 0;

}

/\*\*\*\*\*\*\*\*\*停止音乐\*\*\*\*\*\*\*\*\*/

PlaySound(NULL, NULL, SND\_PURGE);

### 雨点初始化、绘制模块：初始化雨点并绘制

/\*\*\*\*\*\*\*\*\*雨点初始化\*\*\*\*\*\*\*\*\*/

void InitRain(int i)

{

rain[i].y = rand() % first; //随机雨点出现的y坐标--控制在0~30

rain[i].x = rand() % 800; //随即雨点出现的x坐标

rain[i].step = (rand() % 3000) / 3000.0 + v\*1.2;//随机雨点初始速度

}

for (int i = 0; i < maxrain; i++) // 初始化所有雨滴

{

InitRain(i);

}

first = 30; //更改雨点初始位置的y坐标

### 雨点控制模块：雨点的移动

/\*\*\*\*\*\*\*\*\*移动雨点（更新雨点位置）\*\*\*\*\*\*\*\*\*/

void MoveRain(int i)

{

clearrectangle((int)rain[i].x, rain[i].y, (int)rain[i].x + 1, rain[i].y + 4); // 擦掉原来的雨点

line(0, 370, 800, 370);

rain[i].y += (int)rain[i].step; //计算雨点新位置

rain[i].step += 0.1; //雨点加速度 0.1

srand(i\*i\*(unsigned)time(NULL)); //重设随机种子

if (rain[i].y > rand() % 230 + 380)//在落地点绘制水波并重新初始化雨点

{

print\_wave(i);

InitRain(i);

}

// 画新雨点

else {

rectangle((int)rain[i].x, rain[i].y, (int)rain[i].x + 1, rain[i].y + 4);

}

}

### 计时器模块：显示倒计时

/\*\*\*\*\*\*\*\*\*开始-倒计时\*\*\*\*\*\*\*\*\*/

void countdown()

{

settextstyle(35, 24, \_T("楷体"));

TCHAR s[] = \_T("大雨即将来临");

outtextxy(320, 151, s);

settextstyle(60, 37, \_T("楷体"));

for (int f = 5; f >= 0; f--)

{

\_stprintf(s, \_T("%d"), f);

outtextxy(455, 200, s);

Sleep(1200);

clearrectangle(450, 190, 520, 250);

}

clearrectangle(310, 150, 700, 200);

HANDLE hThread3 = CreateThread(NULL, 0, MyThread3, NULL, 0, NULL);

Sleep(20);

}

/\*\*\*\*\*\*\*\*\*结束-倒计时\*\*\*\*\*\*\*\*\*/

void end\_print()

{

cleardevice(); //清屏

PlaySound(NULL, NULL, SND\_PURGE); //停止音乐播放

background(); //重新绘制背景

TCHAR ss[] = \_T("天晴了");

outtextxy(295, 180, ss);

sunup(); //日出

HANDLE hThread1 = CreateThread(NULL, 0, MyThread1, NULL, 0, NULL); //创建太阳发光线程

Sleep(3000);

TCHAR s[] = \_T("窗口将在5秒后关闭");

outtextxy(285, 180, s);

Sleep(1200);

for (int f = 4; f >= 0; f--)

{

\_stprintf(s, \_T("%d"), f); //向字符串写入设置数据格式

outtextxy(475, 180, s);

Sleep(990);

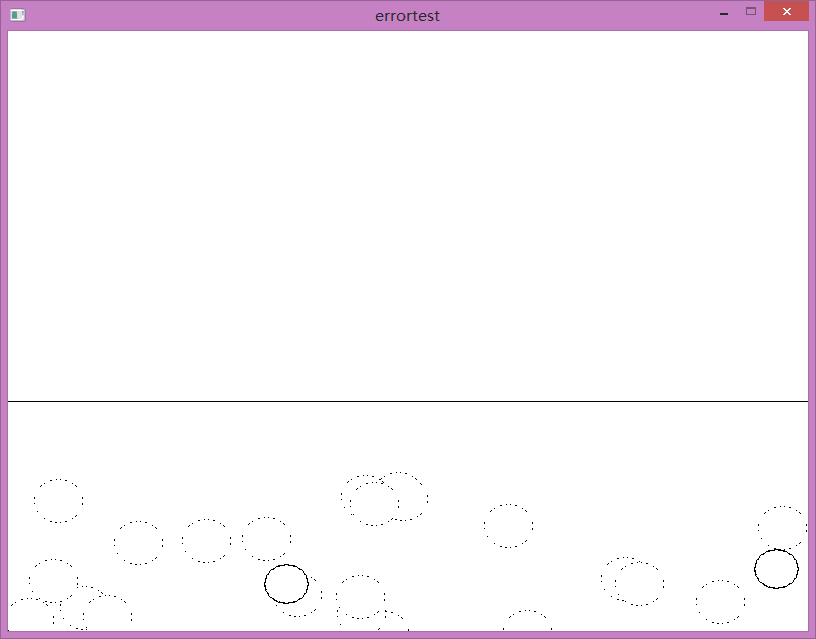
}

}

# 5 综合测试

## 5.1下雨效果模拟1.0 版本测试

### 5.1.1 经测试水波未擦出干净



错误代码如下：

do//永真循环，使其不断产生水波  
{  
for (i = 0; i < 20; i++)//对水波的出现、消失进行循环  
{  
arc(x1, y1, x2, y2, 60, 60);//绘制初始水波1  
arc(x11, y11, x22, y22, 60, 60);//绘制初始水波2  
Sleep(100);//产生水波暂留效果  
clearellipse(x1 , y1 , x2 , y2 );//擦除水波1  
clearellipse(x11 , y11 , x22 , y22 );//擦除水波2  
x1 = x1 - 1;//扩大水波1半径  
y1 = y1 - 1;  
x2 = x2 + 1;  
y2 = y2 + 1;  
x11 = x11 - 1;//扩大水波2半径  
y11 = y11 - 1;  
x22 = x22 + 1;  
y22 = y22 + 1;  
}

经过检查发现水波擦除函数有误，原因是擦除半径小了一个像素，经改正，将擦除半径扩大一个像素。

正确代码如下：

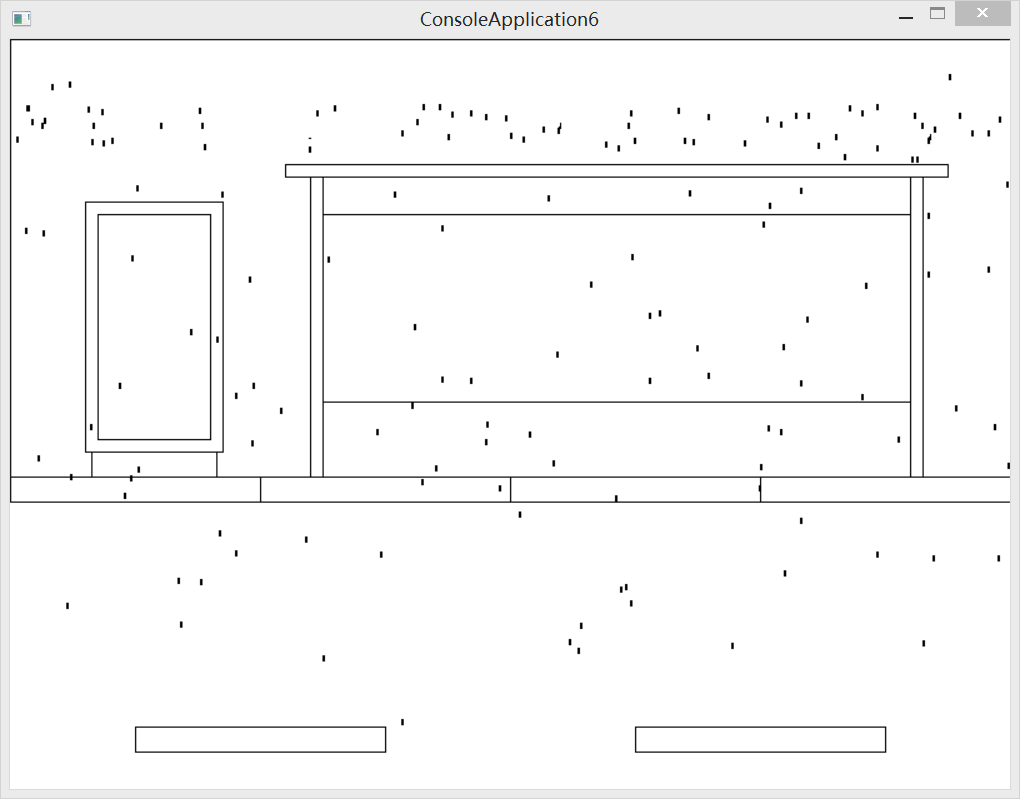
clearellipse(x1-1 , y1 -1, x2 +1, y2 +1);//擦除水波1

clearellipse(x11-1 , y11-1 , x22 +1, y22 +1);//擦除水波2

### 5.1.3手动输入雨量大小（2~5）



### 5.1.4下雨效果显示



## 5.2下雨效果模拟1.1 版本测试

将代码中引起误会的变量名称做了更改，如：

1.0版本中雨滴结构体定义为STAR，1.1版本变为了RAIN

改变前

struct STAR

{

double x;

int y;

double step;

int color;

};

改变后

struct RAIN

{

double x;

int y;

double step;

int color;

};

## 5.3 下雨效果模拟1.2版本测试

### 5.3.1根据题目要求，加入多线程。

把水圈产生的相关参数放在队列中，利用并行线程延时从队列中取出一个水圈位置参数，并将其水圈图像擦除，以达到延长水圈停留时间的目的。

新增头文件

#include <queue>

新增两个线程，一个是原来的程序，并对其中水圈产生函数做了修改；第二线程执行上一版本中擦出水波的功能。

DWORD WINAPI MyThread1(PVOID pvParam)

{

srand((unsigned)time(NULL)); // 随机种子

initgraph(800, 600, SHOWCONSOLE); // 创建窗口

setbkcolor(WHITE);

line(0, 370, 800, 370);

cleardevice();

setfillcolor(WHITE);

setlinecolor(BLACK);

for (int i = 0; i < maxstar; i++)

{

InitStar(i);

star[i].y = rand() % 800;

}

first = 30;

// 雨滴循环下落，随机位置

while (!kbhit())

{

for (int i = 0; i < maxstar; i++)

MoveStar(i);

background();

Sleep(20);

}

yes = false;

closegraph();

return true;

}

DWORD WINAPI MyThread2(PVOID pvParam)

{

while (1)

{

if (!q.empty())

{

int x1, x2, y1, y2;

x1 = q.front(); q.pop();

y1 = q.front(); q.pop();

x2 = q.front(); q.pop();

y2 = q.front(); q.pop();

clearellipse(x1 - 2, y1 - 2, x2 + 2, y2 + 2);

}

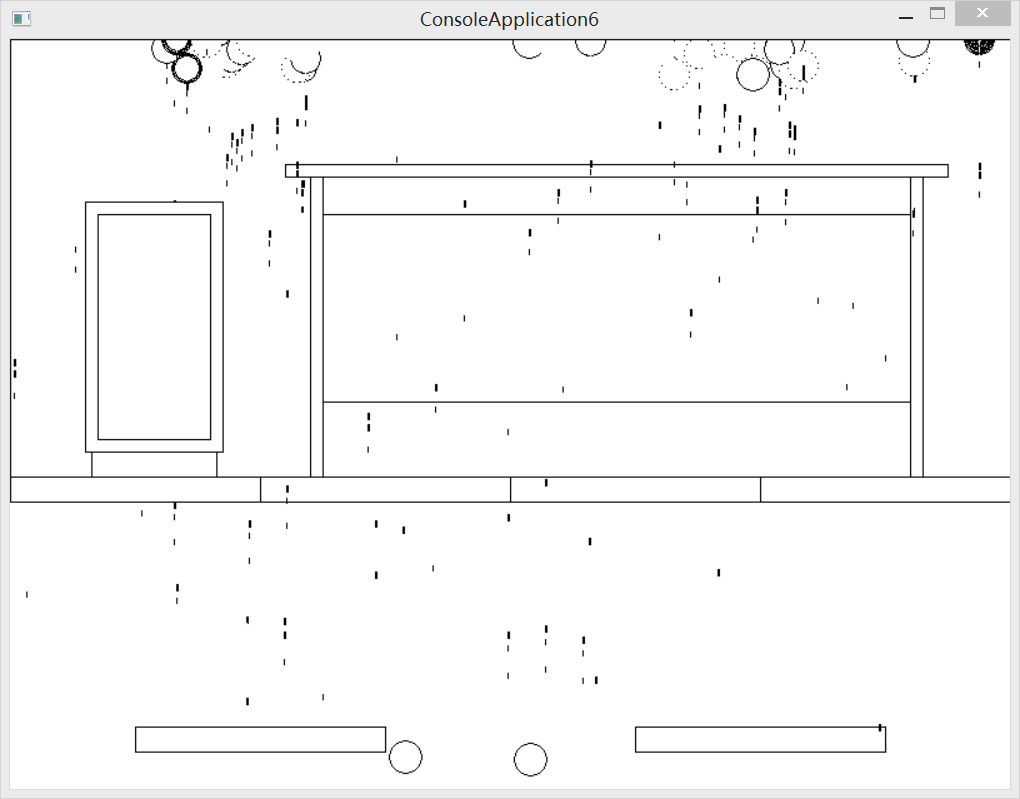
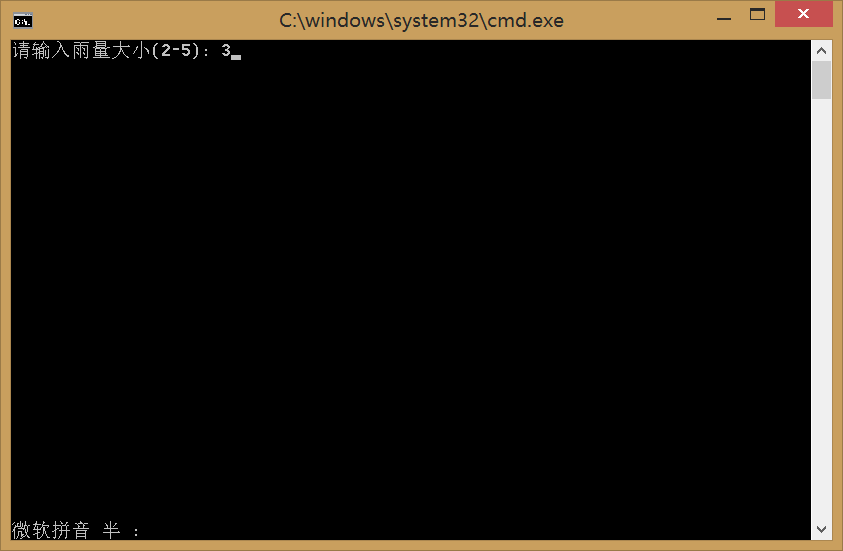
Sleep(40);

}

return true;

}

### 5.3.2 运行程序



### 5.3.3 错误原因

线程之间因资源占用问题相互干扰，所以造成雨滴显示异常。对于此问题我们最终没有得到解决方案，所以我们放弃开辟单独线程控制水圈延迟消失的设计方案。下面将进入2.x版本。

# 5.4 下雨模拟效果2.0版本

### 5.4.1 加入一滴雨，两滴雨效果

为达到水圈出现明显效果，在下大雨之前加入了一滴雨，两滴雨的效果，代码如下：

void run()

{

//一滴雨

RAIN one;

one.x = 400;

one.y = 0;

one.step = 0.005;

//雨落

rectangle((int)one.x, one.y, (int)one.x + 1, one.y + 4);

while (one.y < 500)

{

clearrectangle((int)one.x, one.y, (int)one.x + 1, one.y + 4);

line(0, 370, 800, 370);

background();

one.y += (int)one.step;

one.step += 0.04;

rectangle((int)one.x, one.y, (int)one.x + 1, one.y + 4);

Sleep(10);

}

rectangle((int)one.x, one.y, (int)one.x + 1, one.y + 4);

//水波

int a = 2; int b = 10;

int x1, y1, x2, y2;

x1 = (int)one.x - (b / 2);

y1 = (int)one.y - (a / 2);

x2 = (int)one.x + (b / 2);

y2 = (int)one.y + (a / 2);

for (int i = 0; i < 20; i++) //对水波的出现、消失进行循环

{

arc(x1 - 1, y1 - 1, x2 + 1, y2 + 1, 60, 60);

Sleep(50);

clearellipse(x1 - 1, y1 - 1, x2 + 1, y2 + 1);//擦除水波

x1 = x1 - 1;

y1 = y1 - 1;

x2 = x2 + 1;

y2 = y2 + 1;

}

Sleep(50);

clearellipse(x1 - 1, y1 - 1, x2 + 1, y2 + 1);

//两滴雨

RAIN two1, two2;

two1.x = 200;

two2.x = 600;

two1.y = 0;

two2.y = 0;

two1.step = 0.005;

two2.step = 0.005;

//雨落

rectangle((int)two1.x, two1.y, (int)two1.x + 1, two1.y + 4);

rectangle((int)two2.x, two2.y, (int)two2.x + 1, two2.y + 4);

while (two1.y < 500)

{

clearrectangle((int)two1.x, two1.y, (int)two1.x + 1, two1.y + 4);

clearrectangle((int)two2.x, two2.y, (int)two2.x + 1, two2.y + 4);

line(0, 370, 800, 370);

background();

two1.y += (int)two1.step;

two2.y += (int)two2.step;

two1.step += 0.04;

two2.step += 0.04;

rectangle((int)two1.x, two1.y, (int)two1.x + 1, two1.y + 4);

rectangle((int)two2.x, two2.y, (int)two2.x + 1, two2.y + 4);

Sleep(10);

}

rectangle((int)two1.x, two1.y, (int)two1.x + 1, two1.y + 4);

rectangle((int)two2.x, two2.y, (int)two2.x + 1, two2.y + 4);

//水波

x1 = (int)two1.x - (b / 2);

y1 = (int)two1.y - (a / 2);

x2 = (int)two1.x + (b / 2);

y2 = (int)two1.y + (a / 2);

for (int i = 0; i < 20; i++) //对水波的出现、消失进行循环

{

arc(x1 - 1, y1 - 1, x2 + 1, y2 + 1, 60, 60);

arc(x1 + 400 - 1, y1 - 1, x2 + 400 + 1, y2 + 1, 60, 60);

Sleep(50);

clearellipse(x1 - 1, y1 - 1, x2 + 1, y2 + 1);//擦除水波

clearellipse(x1 + 400 - 1, y1 - 1, x2 + 400 + 1, y2 + 1);

x1 = x1 - 1;

y1 = y1 - 1;

x2 = x2 + 1;

y2 = y2 + 1;

}

Sleep(50);

clearellipse(x1 - 1, y1 - 1, x2 + 1, y2 + 1);

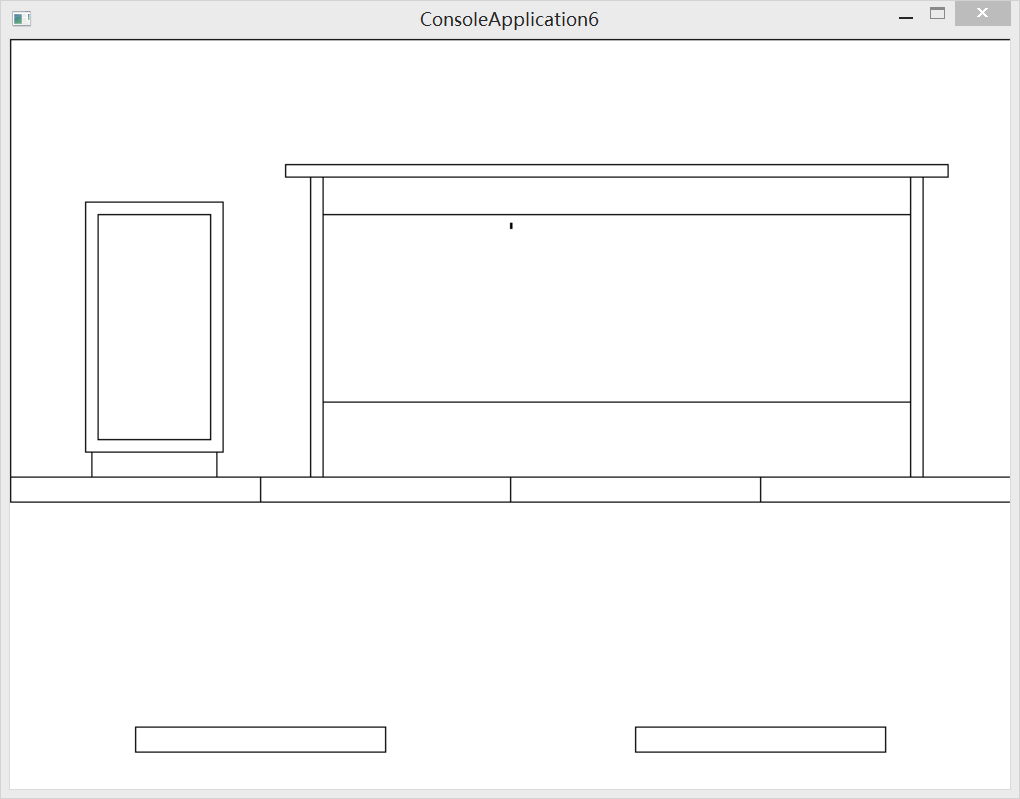
clearellipse(x1 + 400 - 1, y1 - 1, x2 + 400 + 1, y2 + 1);

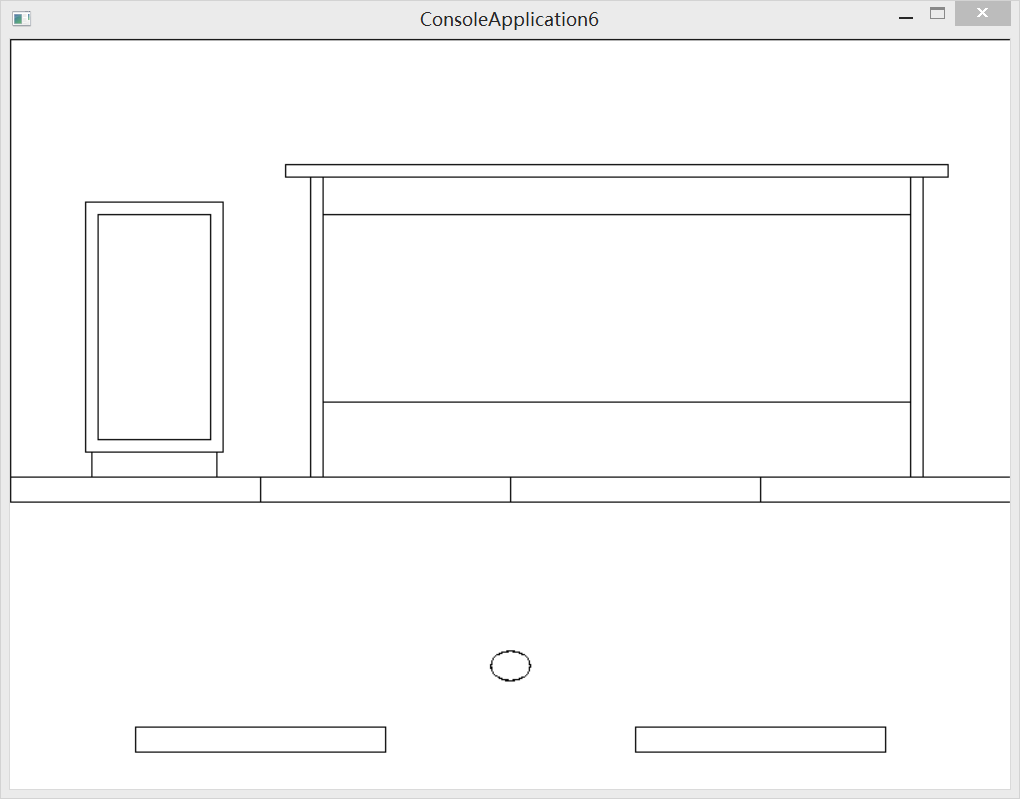
Sleep(100);

}

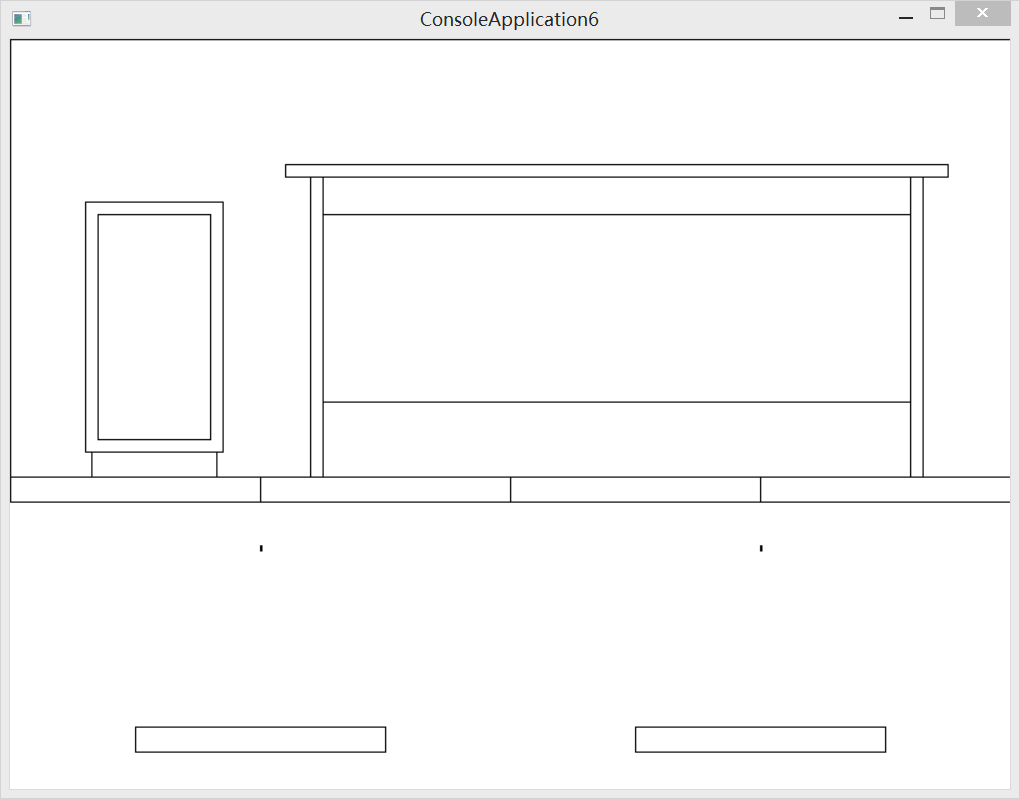
### 5.4.2 效果显示

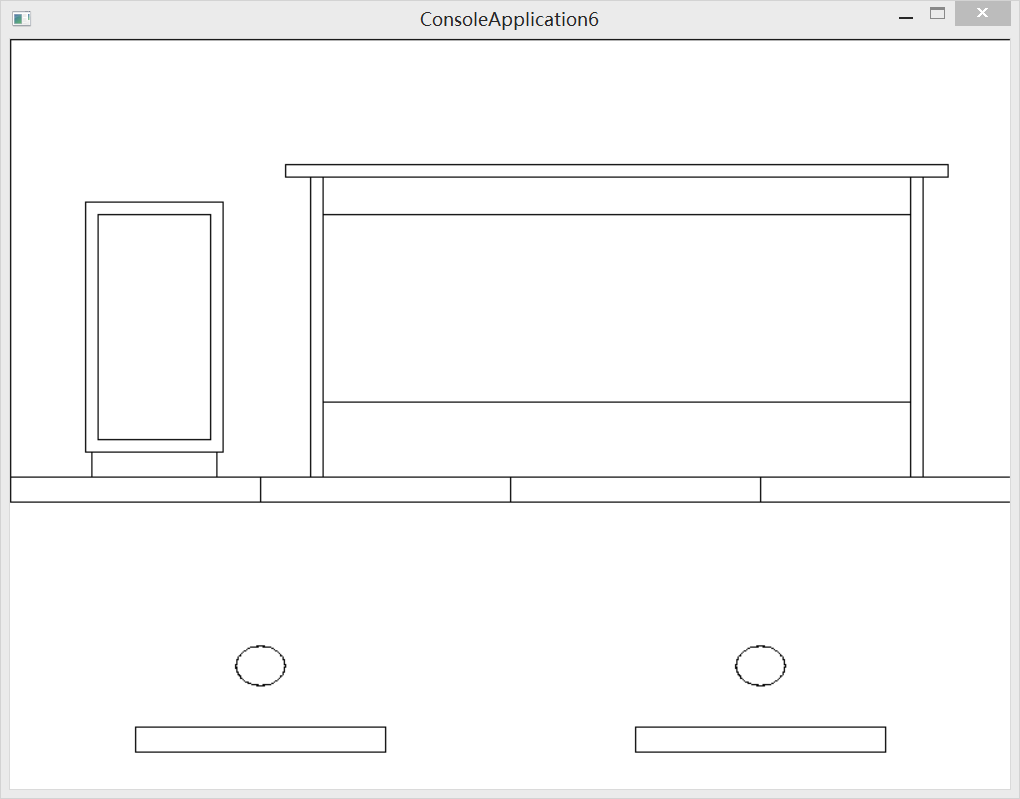
一滴雨效果：





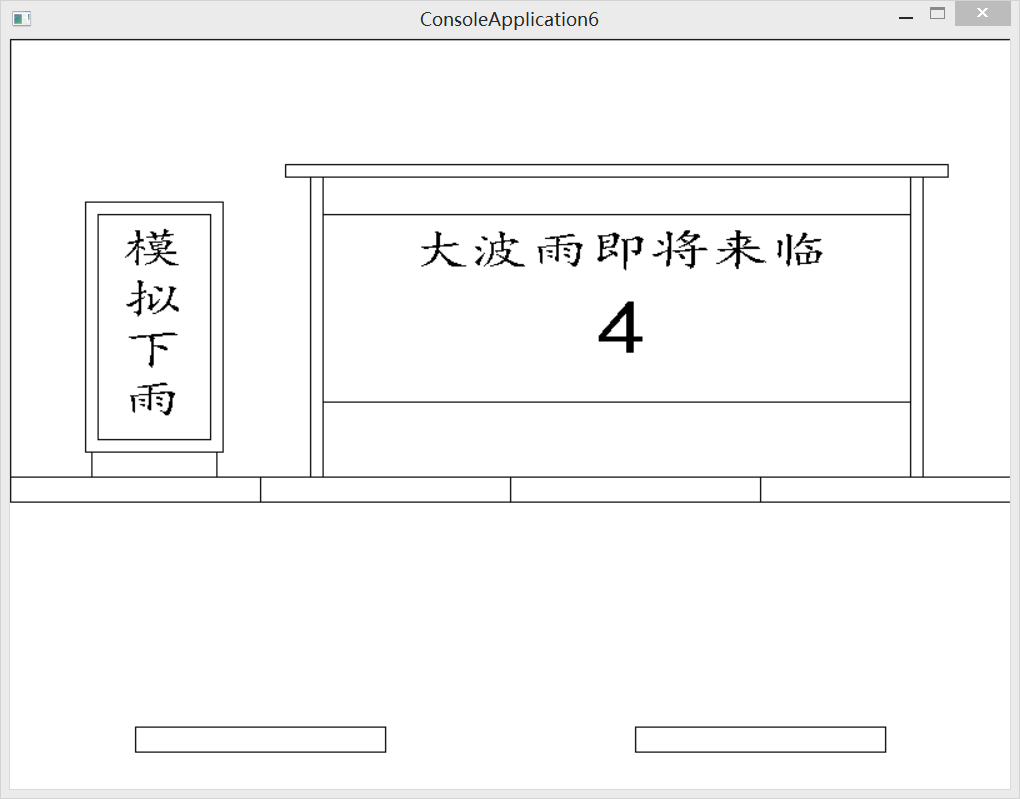
两滴雨效果

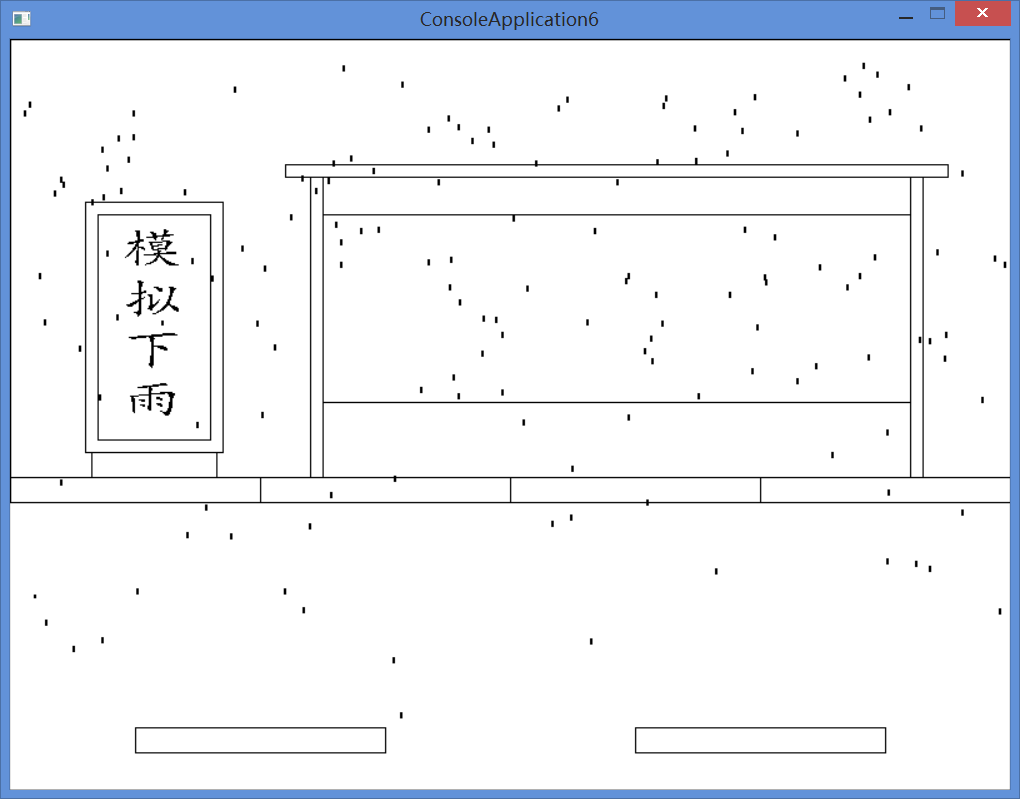




## 5.5 下雨效果模拟2.1版本

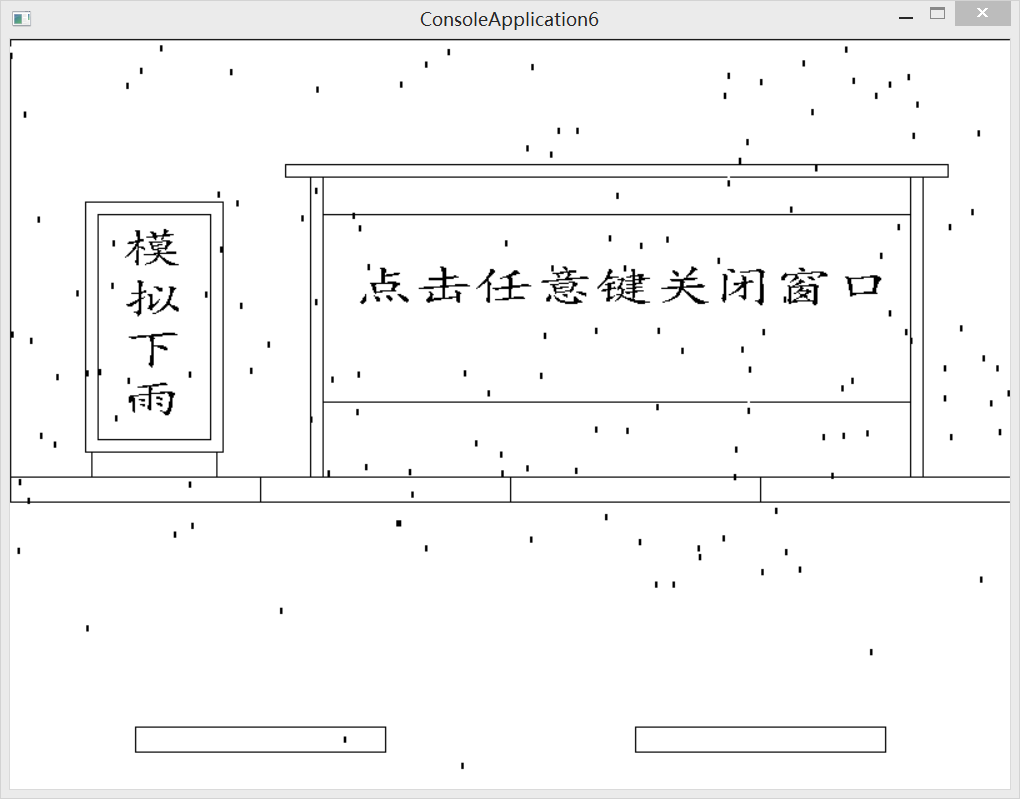
加入文本显示，使背景更加人性化





## 5.6 下雨效果模拟2.2版本

加入“点击任意键关闭窗口”，和“关闭窗口倒计时”。





## 5.7 下雨效果模拟2.2.1版本

加入了音乐文件，并且运用线程控制音乐的播放

#include "mmsystem.h"//导入声音头文件

#pragma comment(lib,"winmm.lib")//导入声音头文件库

/\*\*\*\*\*\*\*\*\*线程-背景音乐\*\*\*\*\*\*\*\*\*/

DWORD WINAPI MyThread1(PVOID pvParam)

{

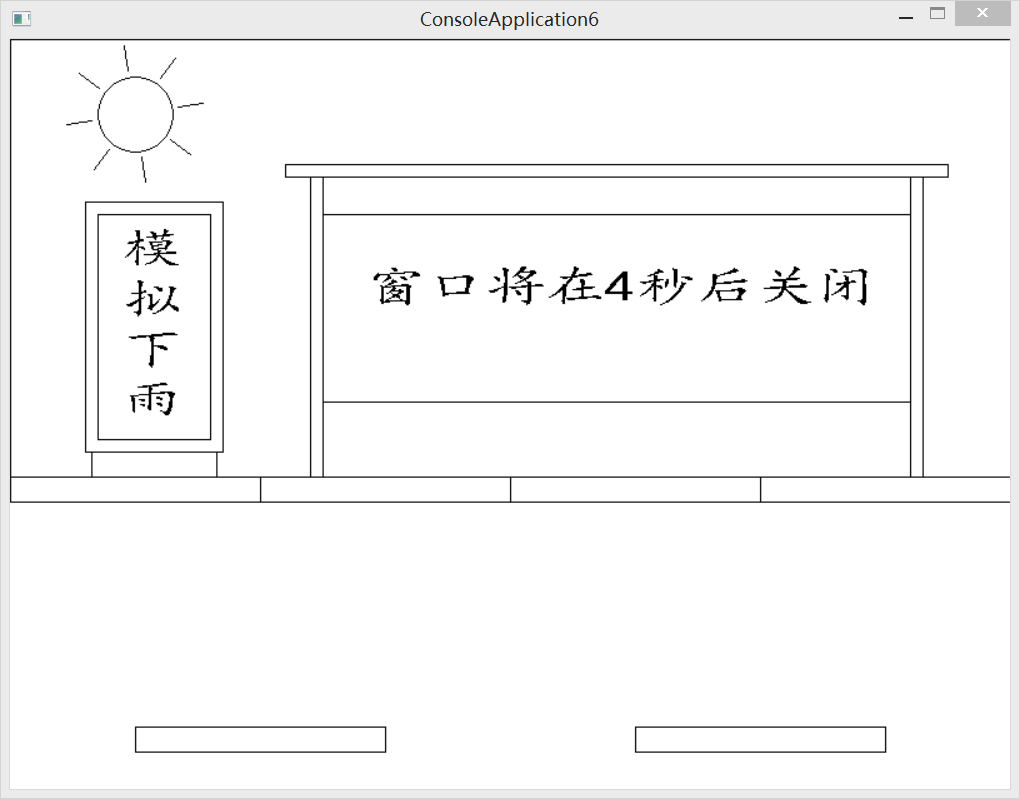
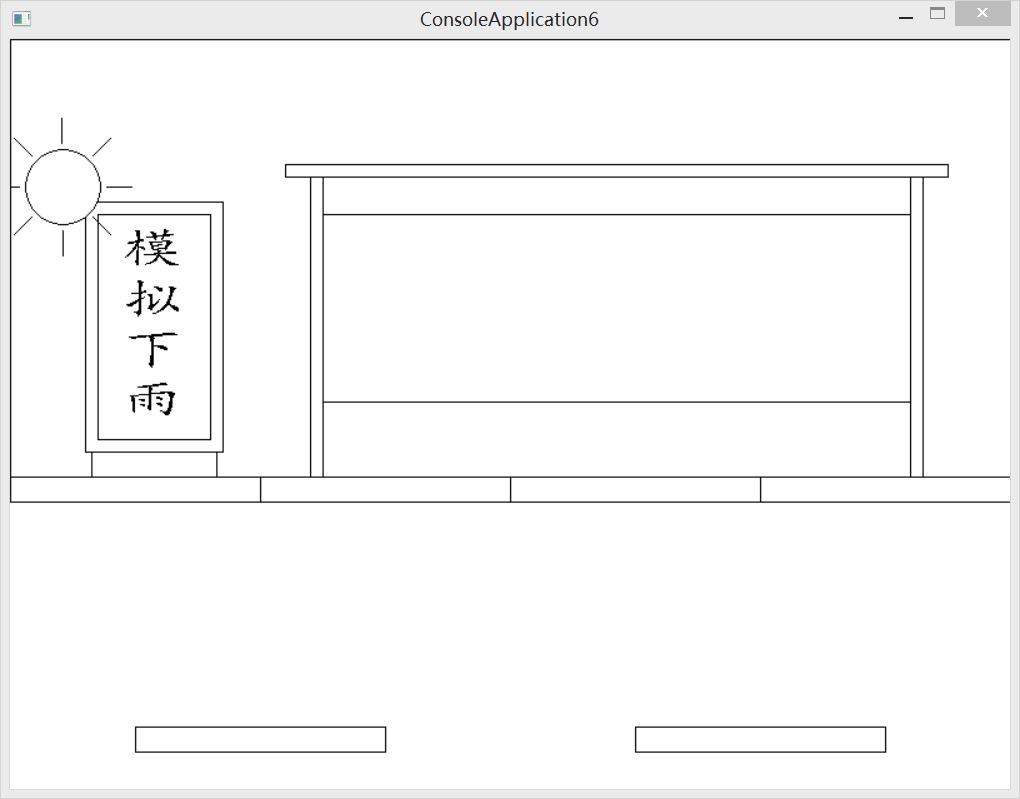
PlaySound(TEXT("F:\\my document\\SmallTerm\\28\\test2\\云之彼端约定的地方.wav"), NULL, SND\_LOOP | SND\_SYNC);

return 0;

}

## 5.8 下雨效果模拟2.3版本

加入了一滴雨，两滴雨效果的文字显示。加入了雨停后太阳升起函数和太阳旋转函数。



## 5.9 下雨效果模拟2.3.1版本

增添控制太阳旋转的线程

/\*\*\*\*\*\*\*\*\*线程-太阳旋转\*\*\*\*\*\*\*\*\*/

DWORD WINAPI MyThread1(PVOID pvParam)

{

while (true)

{

sunrotate();

}

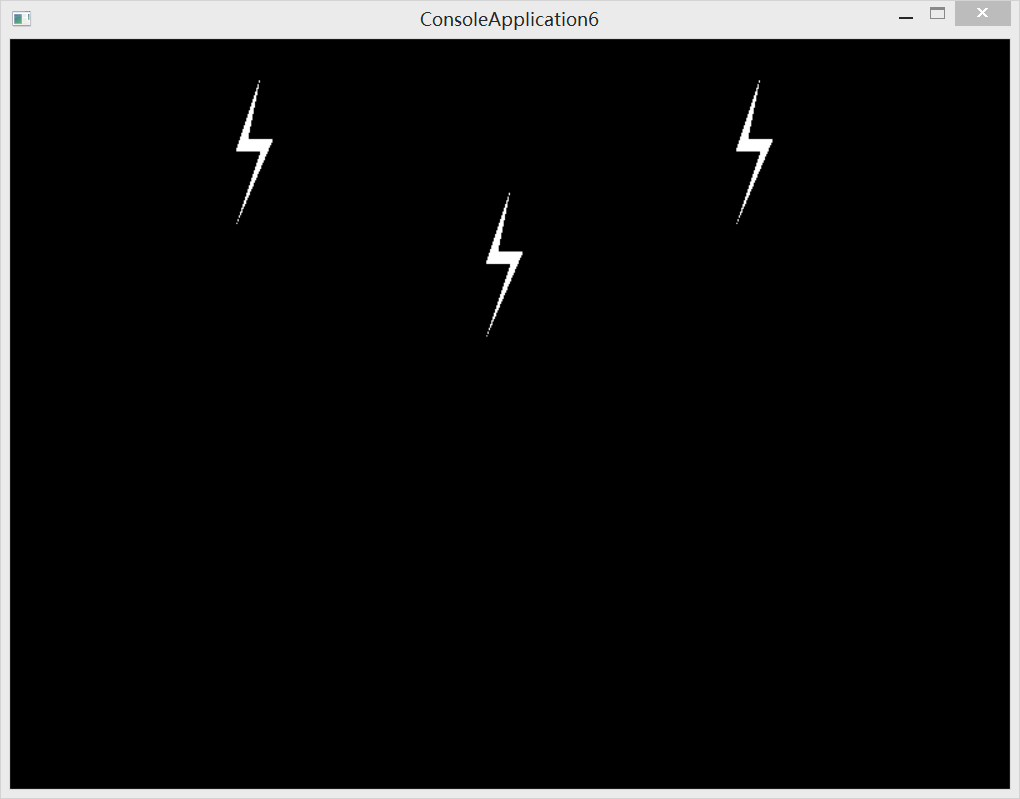
return 0;

}

加入闪屏效果，为制作闪电做准备。

## 5.10 下雨效果模拟2.4版本

加入闪电效果

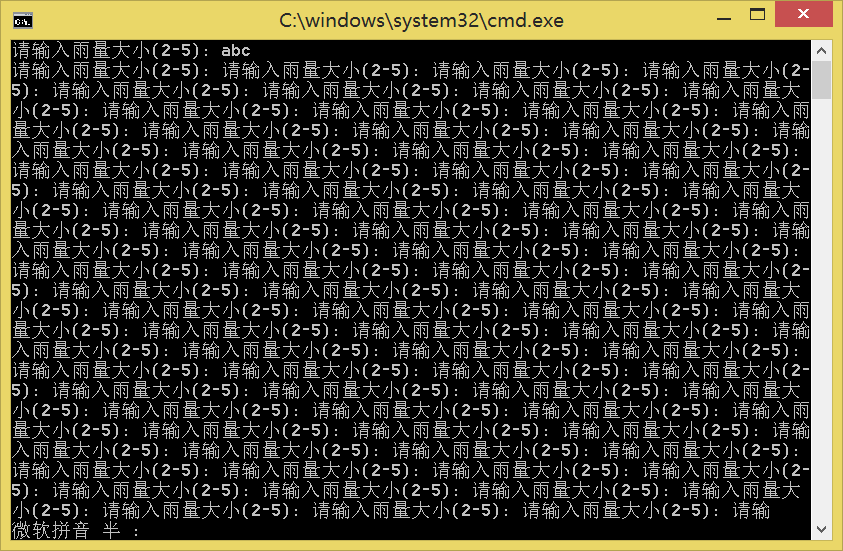


## 5.11 下雨效果模拟2.4.1

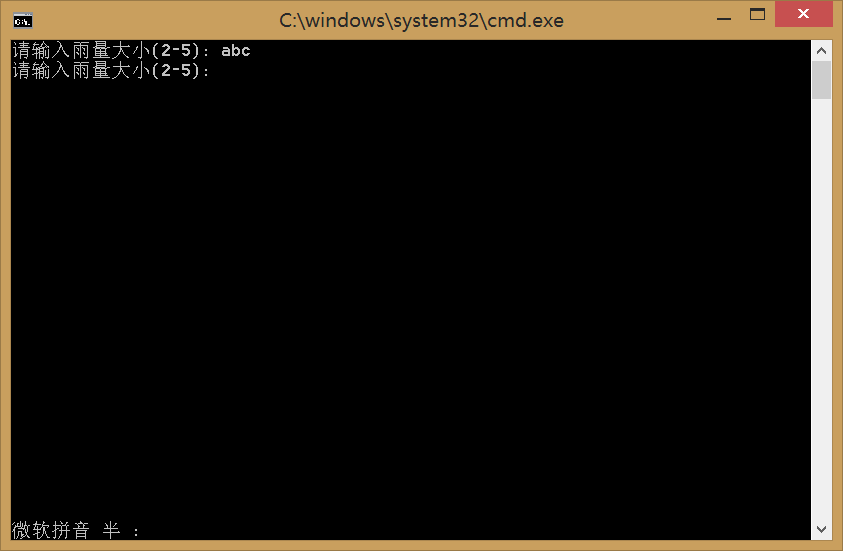
规范代码格式，完善注释。

修复一处开始输入的BUG，当输入字符时显示异常。

改正前：



改正后：



# 附录：

## 模拟下雨效果 v3.0Final

/\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 模拟下雨效果\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*头文件\*\*\*\*\*\*\*\*\*/

#include <graphics.h>

#include <ctime>

#include <conio.h>

#include <iostream>

#include <cmath>

#include "mmsystem.h" //导入声音头文件

#pragma comment(lib,"winmm.lib") //导入声音头文件库

#define PI 3.1415926 //π值

using namespace *std*;

/\*\*\*\*\*\*\*\*\*全局变量\*\*\*\*\*\*\*\*\*/

int maxrain; // 雨点总数

int first = 370; //雨点初始随机位置相关参量

double v; //雨点初始下落随机速度相关参量

/\*\*\*\*\*\*\*\*\*雨点结构体\*\*\*\*\*\*\*\*\*/

struct RAIN

{

int x; //雨点x坐标

int y; //雨点y坐标

double step; //雨点速度

};

RAIN rain[600]; //初始化雨点结构体数组

/\*\*\*\*\*\*\*\*\*函数声明\*\*\*\*\*\*\*\*\*/

*DWORD* *WINAPI* MyThread3(*PVOID* pvParam);

*DWORD* *WINAPI* MyThread2(*PVOID* pvParam);

*DWORD* *WINAPI* MyThread1(*PVOID* pvParam);

void InitRain(int i);

void print\_lightning();

void background();

void print\_wave(int i);

void MoveRain(int i);

void run\_few();

void countdown();

void end\_print();

void init();

void running();

void sunup();

void sunrotate();

/\*\*\*\*\*\*\*\*\*主函数\*\*\*\*\*\*\*\*\*/

void *main*()

{

init(); //初始化

run\_few(); //模拟少量雨点效果

countdown(); //倒计时显示-5s

print\_lightning(); //显示闪电效果

running(); //模拟下雨效果

end\_print(); //结束下雨效果

*closegraph*(); // 关闭窗口

}

/\*\*\*\*\*\*\*\*\*线程-背景音乐\*\*\*\*\*\*\*\*\*/

*DWORD* *WINAPI* MyThread3(*PVOID* pvParam)

{

*PlaySound*(*TEXT*("F:\\my document\\SmallTerm\\28\\test2\\Rainymood.wav"), *NULL*, *SND\_LOOP* | *SND\_ASYNC*);

return 0;

}

/\*\*\*\*\*\*\*\*\*线程-太阳旋转\*\*\*\*\*\*\*\*\*/

*DWORD* *WINAPI* MyThread1(*PVOID* pvParam)

{

while (true)

{

sunrotate();

}

return 0;

}

/\*\*\*\*\*\*\*\*\*线程-鸟鸣音乐\*\*\*\*\*\*\*\*\*/

*DWORD* *WINAPI* MyThread2(*PVOID* pvParam)

{

*PlaySound*(*TEXT*("F:\\my document\\SmallTerm\\28\\test2\\鸟鸣声.wav"), *NULL*, *SND\_LOOP* | *SND\_ASYNC*);

return 0;

}

/\*\*\*\*\*\*\*\*\*雨点初始化\*\*\*\*\*\*\*\*\*/

void InitRain(int i)

{

rain[i].y = *rand*() % first; //随机雨点出现的y坐标--控制在0~30

rain[i].x = *rand*() % 800; //随即雨点出现的x坐标

rain[i].step = (*rand*() % 3000) / 3000.0 + v\*1.2; //随机雨点初始速度

}

/\*\*\*\*\*\*\*\*\*闪电\*\*\*\*\*\*\*\*\*/

void print\_lightning()

{

*Sleep*(50);

*setfillcolor*(*WHITE*);

*setbkcolor*(*BLACK*);

*cleardevice*();

int i;//设置循环计数变量i

for (i = 0; i < 3; i++)

{

/\*第一轮闪电\*/

//绘制闪电1

*POINT* pts1[] = { { 200, 30 }, { 180, 90 }, { 200, 90 }, { 180, 150 }, { 210, 80 }, { 190, 80 } };

*solidpolygon*(pts1, 6);

//绘制闪电2

*POINT* pts2[] = { { 400, 120 }, { 380, 180 }, { 400, 180 }, { 380, 240 }, { 410, 170 }, { 390, 170 } };

*solidpolygon*(pts2, 6);

//绘制闪电3

*POINT* pts3[] = { { 600, 30 }, { 580, 90 }, { 600, 90 }, { 580, 150 }, { 610, 80 }, { 590, 80 } };

*solidpolygon*(pts3, 6);

*Sleep*(200);

*clearrectangle*(180, 30, 210, 150);//擦除闪电

*clearrectangle*(380, 120, 410, 240);

*clearrectangle*(580, 30, 610, 150);

*Sleep*(200);//产生闪烁效果

}

for (i = 0; i < 3; i++)

{

/\*第二轮闪电\*/

//绘制闪电4

*POINT* pts4[] = { { 200, 120 }, { 180, 180 }, { 200, 180 }, { 180, 240 }, { 210, 170 }, { 190, 170 } };

*solidpolygon*(pts4, 6);

//绘制闪电5

*POINT* pts5[] = { { 400, 30 }, { 380, 90 }, { 400, 90 }, { 380, 150 }, { 410, 80 }, { 390, 80 } };

*solidpolygon*(pts5, 6);

//绘制闪电6

*POINT* pts6[] = { { 600, 120 }, { 580, 180 }, { 600, 180 }, { 580, 240 }, { 610, 170 }, { 590, 170 } };

*solidpolygon*(pts6, 6);

*Sleep*(200);

*clearrectangle*(180, 120, 210, 240);//擦除闪电

*clearrectangle*(380, 30, 410, 150);

*clearrectangle*(580, 120, 610, 240);

*Sleep*(200);//产生闪烁效果

}

*setbkcolor*(*WHITE*);

*cleardevice*();

background();

}

/\*\*\*\*\*\*\*\*\*背景绘制\*\*\*\*\*\*\*\*\*/

void background()

{

// 背景色

*setlinecolor*(*BLACK*);

// 站牌

*rectangle*(0, 0, 800, 370);

// 边框

*rectangle*(60, 130, 170, 330);

// 内部

*rectangle*(70, 140, 160, 320);

// 底部

*rectangle*(65, 330, 165, 350);

// 等车棚

// 顶部

*rectangle*(220, 100, 750, 110);

// 柱子

*rectangle*(240, 110, 250, 350);

*rectangle*(720, 110, 730, 350);

// 公告

*rectangle*(250, 140, 720, 290);

//路边

*rectangle*(0, 350, 200, 370);

*rectangle*(200, 350, 400, 370);

*rectangle*(400, 350, 600, 370);

*rectangle*(600, 350, 800, 370);

//马路线

*rectangle*(100, 550, 300, 570);

*rectangle*(500, 550, 700, 570);

*settextstyle*(35, 24, *\_T*("楷体"));

*settextcolor*(*BLACK*);

*TCHAR* s[] = *\_T*("模");

*outtextxy*(90, 150, s);

*TCHAR* ss[] = *\_T*("拟");

*outtextxy*(90, 190, ss);

*TCHAR* sss[] = *\_T*("下");

*outtextxy*(90, 230, sss);

*TCHAR* ssss[] = *\_T*("雨");

*outtextxy*(90, 270, ssss);

}

/\*\*\*\*\*\*\*\*\*水波的产生及消失\*\*\*\*\*\*\*\*\*/

void print\_wave(int i)

{

int a = 10; int b = 30;

int x1, y1, x2, y2;

x1 = (int)rain[i].x - (b / 2);

y1 = (int)rain[i].y - (a / 2);

x2 = (int)rain[i].x + (b / 2);

y2 = (int)rain[i].y + (a / 2);

for (int i = 0; i < 10; i++) //对水波的出现、消失进行循环

{

*arc*(x1 - 1, y1 - 1, x2 + 1, y2 + 1, 60, 60);

*clearellipse*(x1 - 1, y1 - 1, x2 + 1, y2 + 1);//擦除水波

x1 = x1 - 1;

y1 = y1 - 1;

x2 = x2 + 1;

y2 = y2 + 1;

}

*clearellipse*(x1 - 1, y1 - 1, x2 + 1, y2 + 1);

}

/\*\*\*\*\*\*\*\*\*移动雨点（更新雨点位置）\*\*\*\*\*\*\*\*\*/

void MoveRain(int i)

{

*clearrectangle*((int)rain[i].x, rain[i].y, (int)rain[i].x + 1, rain[i].y + 4); // 擦掉原来的雨点

*line*(0, 370, 800, 370);

rain[i].y += (int)rain[i].step; //计算雨点新位置

rain[i].step += 0.1; //雨点加速度 0.1

*srand*(i\*i\*(unsigned)*time*(*NULL*)); //重设随机种子

if (rain[i].y > *rand*() % 230 + 380) //在落地点绘制水波并重新初始化雨点

{

print\_wave(i);

InitRain(i);

}

// 画新雨点

else {

*rectangle*((int)rain[i].x, rain[i].y, (int)rain[i].x + 1, rain[i].y + 4);

}

}

/\*\*\*\*\*\*\*\*\*对少量雨点下落的模拟\*\*\*\*\*\*\*\*\*/

void run\_few()

{

//一滴雨

RAIN one;

one.x = 400;

one.y = 0;

one.step = 0.005;

//雨落

*rectangle*((int)one.x, one.y, (int)one.x + 1, one.y + 4);

while (one.y < 500)

{

*clearrectangle*((int)one.x, one.y, (int)one.x + 1, one.y + 4);

*line*(0, 370, 800, 370);

background();

*TCHAR* s[] = *\_T*("一滴雨效果");

*outtextxy*(360, 180, s);

one.y += (int)one.step;

one.step += 0.04;

*rectangle*((int)one.x, one.y, (int)one.x + 1, one.y + 4);

*Sleep*(10);

}

*rectangle*((int)one.x, one.y, (int)one.x + 1, one.y + 4);

//水波

int a = 2; int b = 10;

int x1, y1, x2, y2;

x1 = (int)one.x - (b / 2);

y1 = (int)one.y - (a / 2);

x2 = (int)one.x + (b / 2);

y2 = (int)one.y + (a / 2);

for (int i = 0; i < 20; i++) //对水波的出现、消失进行循环

{

*arc*(x1 - 1, y1 - 1, x2 + 1, y2 + 1, 60, 60);

*Sleep*(50);

*clearellipse*(x1 - 1, y1 - 1, x2 + 1, y2 + 1);//擦除水波

x1 = x1 - 1;

y1 = y1 - 1;

x2 = x2 + 1;

y2 = y2 + 1;

}

*Sleep*(50);

*clearellipse*(x1 - 1, y1 - 1, x2 + 1, y2 + 1);

*clearrectangle*(260, 150, 710, 280);

//两滴雨

RAIN two1, two2;

two1.x = 200;

two2.x = 600;

two1.y = 0;

two2.y = 0;

two1.step = 0.005;

two2.step = 0.005;

//雨落

*rectangle*((int)two1.x, two1.y, (int)two1.x + 1, two1.y + 4);

*rectangle*((int)two2.x, two2.y, (int)two2.x + 1, two2.y + 4);

while (two1.y < 500)

{

*clearrectangle*((int)two1.x, two1.y, (int)two1.x + 1, two1.y + 4);

*clearrectangle*((int)two2.x, two2.y, (int)two2.x + 1, two2.y + 4);

*line*(0, 370, 800, 370);

background();

*TCHAR* s[] = *\_T*("两滴雨效果");

*outtextxy*(360, 180, s);

two1.y += (int)two1.step;

two2.y += (int)two2.step;

two1.step += 0.04;

two2.step += 0.04;

*rectangle*((int)two1.x, two1.y, (int)two1.x + 1, two1.y + 4);

*rectangle*((int)two2.x, two2.y, (int)two2.x + 1, two2.y + 4);

*Sleep*(10);

}

*rectangle*((int)two1.x, two1.y, (int)two1.x + 1, two1.y + 4);

*rectangle*((int)two2.x, two2.y, (int)two2.x + 1, two2.y + 4);

//水波

x1 = (int)two1.x - (b / 2);

y1 = (int)two1.y - (a / 2);

x2 = (int)two1.x + (b / 2);

y2 = (int)two1.y + (a / 2);

for (int i = 0; i < 20; i++) //对水波的出现、消失进行循环

{

*arc*(x1 - 1, y1 - 1, x2 + 1, y2 + 1, 60, 60);

*arc*(x1 + 400 - 1, y1 - 1, x2 + 400 + 1, y2 + 1, 60, 60);

*Sleep*(50);

*clearellipse*(x1 - 1, y1 - 1, x2 + 1, y2 + 1);//擦除水波

*clearellipse*(x1 + 400 - 1, y1 - 1, x2 + 400 + 1, y2 + 1);

x1 = x1 - 1;

y1 = y1 - 1;

x2 = x2 + 1;

y2 = y2 + 1;

}

*Sleep*(50);

*clearellipse*(x1 - 1, y1 - 1, x2 + 1, y2 + 1);

*clearellipse*(x1 + 400 - 1, y1 - 1, x2 + 400 + 1, y2 + 1);

*clearrectangle*(260, 150, 710, 280);

*Sleep*(100);

}

/\*\*\*\*\*\*\*\*\*开始-倒计时\*\*\*\*\*\*\*\*\*/

void countdown()

{

*settextstyle*(35, 24, *\_T*("楷体"));

*TCHAR* s[] = *\_T*("大雨即将来临");

*outtextxy*(320, 151, s);

*settextstyle*(60, 37, *\_T*("楷体"));

for (int f = 5; f >= 0; f--)

{

*\_stprintf*(s, *\_T*("%d"), f);

*outtextxy*(455, 200, s);

*Sleep*(1200);

*clearrectangle*(450, 190, 520, 250);

}

*clearrectangle*(310, 150, 700, 200);

*HANDLE* hThread3 = *CreateThread*(*NULL*, 0, MyThread3, *NULL*, 0, *NULL*);

*Sleep*(20);

}

/\*\*\*\*\*\*\*\*\*结束-倒计时\*\*\*\*\*\*\*\*\*/

void end\_print()

{

*cleardevice*(); //清屏

*PlaySound*(*NULL*, *NULL*, *SND\_PURGE*); //停止音乐播放

background(); //重新绘制背景

*TCHAR* ss[] = *\_T*("天晴了");

*outtextxy*(295, 180, ss);

sunup(); //日出

*HANDLE* hThread1 = *CreateThread*(*NULL*, 0, MyThread1, *NULL*, 0, *NULL*); //创建太阳发光线程

*Sleep*(3000);

*TCHAR* s[] = *\_T*("窗口将在5秒后关闭");

*outtextxy*(285, 180, s);

*Sleep*(1200);

for (int f = 4; f >= 0; f--)

{

*\_stprintf*(s, *\_T*("%d"), f); //向字符串写入设置数据格式

*outtextxy*(475, 180, s);

*Sleep*(990);

}

}

/\*\*\*\*\*\*\*\*\*初始化\*\*\*\*\*\*\*\*\*/

void init()

{

char read\_in[100];

while (1) //雨量大小选择

{

*cout* << "请输入雨量大小(2-5): ";

*cin* >> read\_in;

if (read\_in[0] >= '2' && read\_in[0] <= '5'&&read\_in[1] == '\0')

{

break;

}

}

maxrain = (read\_in[0] - '0') \* 80; //对雨点数量初始化

v = ((read\_in[0] - '0') + 3)\*0.8; //对雨点随机速度参量初始化

*srand*((unsigned)*time*(*NULL*)); // 随机种子

*initgraph*(800, 600); // 创建窗口

*setbkcolor*(*WHITE*); //设置窗口背景为白色

*cleardevice*();

*setfillcolor*(*BLACK*); //设置雨点填充颜色为黑色

*setlinecolor*(*BLACK*); //设置线条颜色为黑色

background();

}

/\*\*\*\*\*\*\*\*\*模拟下雨效果\*\*\*\*\*\*\*\*\*/

void running()

{

for (int i = 0; i < maxrain; i++) // 初始化所有雨滴

{

InitRain(i);

}

first = 30; //更改雨点初始位置的y坐标

// 雨滴循环下落，随机位置

while (!*kbhit*())

{

for (int i = 0; i < maxrain; i++)

MoveRain(i); //遍历雨点，更新其位置

background(); //对背景进行重绘

*settextstyle*(30, 19, *\_T*("楷体"));

*TCHAR* s[] = *\_T*("按下键盘任意键结束");

*outtextxy*(285, 180, s);

*Sleep*(10);

}

}

/\*\*\*\*\*\*\*\*\*太阳升起\*\*\*\*\*\*\*\*\*/

void sunup()

{

*setfillcolor*(*WHITE*);

for (int i = 0; i < 160; i++)

{

*fillcircle*(-60 + i, 220 - i, 30);

*line*((int)(-60 + 35 \* *cos*(0) + i), (int)(220 + 35 \* *sin*(0) - i), (int)(-60 + 55 \* *cos*(0) + i), (int)(220 + 55 \* *sin*(0) - i));

*line*((int)(-60 + 35 \* *cos*(PI / 4) + i), (int)(220 + 35 \* *sin*(PI / 4) - i), (int)(-60 + 55 \* *cos*(PI / 4) + i), (int)(220 + 55 \* *sin*(PI / 4) - i));

*line*((int)(-60 + 35 \* *cos*(PI / 2) + i), (int)(220 + 35 \* *sin*(PI / 2) - i), (int)(-60 + 55 \* *cos*(PI / 2) + i), (int)(220 + 55 \* *sin*(PI / 2) - i));

*line*((int)(-60 + 35 \* *cos*(PI / 4 \* 3) + i), (int)(220 + 35 \* *sin*(PI / 4 \* 3) - i), (int)(-60 + 55 \* *cos*(PI / 4 \* 3) + i), (int)(220 + 55 \* *sin*(PI / 4 \* 3) - i));

*line*((int)(-60 + 35 \* *cos*(PI) + i), (int)(220 + 35 \* *sin*(PI) - i), (int)(-60 + 55 \* *cos*(PI) + i), (int)(220 + 55 \* *sin*(PI) - i));

*line*((int)(-60 + 35 \* *cos*(PI / 4 \* 5) + i), (int)(220 + 35 \* *sin*(PI / 4 \* 5) - i), (int)(-60 + 55 \* *cos*(PI / 4 \* 5) + i), (int)(220 + 55 \* *sin*(PI / 4 \* 5) - i));

*line*((int)(-60 + 35 \* *cos*(PI / 2 \* 3) + i), (int)(220 + 35 \* *sin*(PI / 2 \* 3) - i), (int)(-60 + 55 \* *cos*(PI / 2 \* 3) + i), (int)(220 + 55 \* *sin*(PI / 2 \* 3) - i));

*line*((int)(-60 + 35 \* *cos*(PI / 4 \* 7) + i), (int)(220 + 35 \* *sin*(PI / 4 \* 7) - i), (int)(-60 + 55 \* *cos*(PI / 4 \* 7) + i), (int)(220 + 55 \* *sin*(PI / 4 \* 7) - i));

*Sleep*(50);

*clearrectangle*(-60 + i - 60, 220 - i - 60, -60 + i + 60, 220 - i + 60);

if (i==80)

{

*HANDLE* hThread2 = *CreateThread*(*NULL*, 0, MyThread2, *NULL*, 0, *NULL*);

}

background();

}

}

/\*\*\*\*\*\*\*\*\*太阳旋转\*\*\*\*\*\*\*\*\*/

void sunrotate()

{

for (int i = 0; i < 5; i++)

{

*circle*(100, 60, 30);

*line*((int)(100 + 35 \* *cos*(PI / 5 \* i)), (int)(60 + 35 \* *sin*(PI / 5 \* i)), (int)(100 + 55 \* *cos*(PI / 5 \* i)), (int)(60 + 55 \* *sin*(PI / 5 \* i)));

*line*((int)(100 + 35 \* *cos*(PI / 5 \* i + PI / 4)), (int)(60 + 35 \* *sin*(PI / 5 \* i + PI / 4)), (int)(100 + 55 \* *cos*(PI / 5 \* i + PI / 4)), (int)(60 + 55 \* *sin*(PI / 5 \* i + PI / 4)));

*line*((int)(100 + 35 \* *cos*(PI / 5 \* i + PI / 2)), (int)(60 + 35 \* *sin*(PI / 5 \* i + PI / 2)), (int)(100 + 55 \* *cos*(PI / 5 \* i + PI / 2)), (int)(60 + 55 \* *sin*(PI / 5 \* i + PI / 2)));

*line*((int)(100 + 35 \* *cos*(PI / 5 \* i + PI / 4 \* 3)), (int)(60 + 35 \* *sin*(PI / 5 \* i + PI / 4 \* 3)), (int)(100 + 55 \* *cos*(PI / 5 \* i + PI / 4 \* 3)), (int)(60 + 55 \* *sin*(PI / 5 \* i + PI / 4 \* 3)));

*line*((int)(100 + 35 \* *cos*(PI / 5 \* i + PI)), (int)(60 + 35 \* *sin*(PI / 5 \* i + PI)), (int)(100 + 55 \* *cos*(PI / 5 \* i + PI)), (int)(60 + 55 \* *sin*(PI / 5 \* i + PI)));

*line*((int)(100 + 35 \* *cos*(PI / 5 \* i + PI / 4 \* 5)), (int)(60 + 35 \* *sin*(PI / 5 \* i + PI / 4 \* 5)), (int)(100 + 55 \* *cos*(PI / 5 \* i + PI / 4 \* 5)), (int)(60 + 55 \* *sin*(PI / 5 \* i + PI / 4 \* 5)));

*line*((int)(100 + 35 \* *cos*(PI / 5 \* i + PI / 2 \* 3)), (int)(60 + 35 \* *sin*(PI / 5 \* i + PI / 2 \* 3)), (int)(100 + 55 \* *cos*(PI / 5 \* i + PI / 2 \* 3)), (int)(60 + 55 \* *sin*(PI / 5 \* i + PI / 2 \* 3)));

*line*((int)(100 + 35 \* *cos*(PI / 5 \* i + PI / 4 \* 7)), (int)(60 + 35 \* *sin*(PI / 5 \* i + PI / 4 \* 7)), (int)(100 + 55 \* *cos*(PI / 5 \* i + PI / 4 \* 7)), (int)(60 + 55 \* *sin*(PI / 5 \* i + PI / 4 \* 7)));

*Sleep*(200);

*clearrectangle*(40, 1, 160, 120);

}

}