南開大學
Nankai University

高级语言程序设计

实验报告

南开大学 工科试验班（信息科学与技术）
姓名 刘修铭
学号 2112492
班级 0973
2022 年 4 月 25 日

目 录

高级语言程序设计大作业实验报告

一、作业题目

使用 C++图形化完成 Zouma 小游戏，并用冬奥会元素加以修饰。

二、开发软件

Visual Studio 2019，EasyX

三、课题要求

（1）面向对象

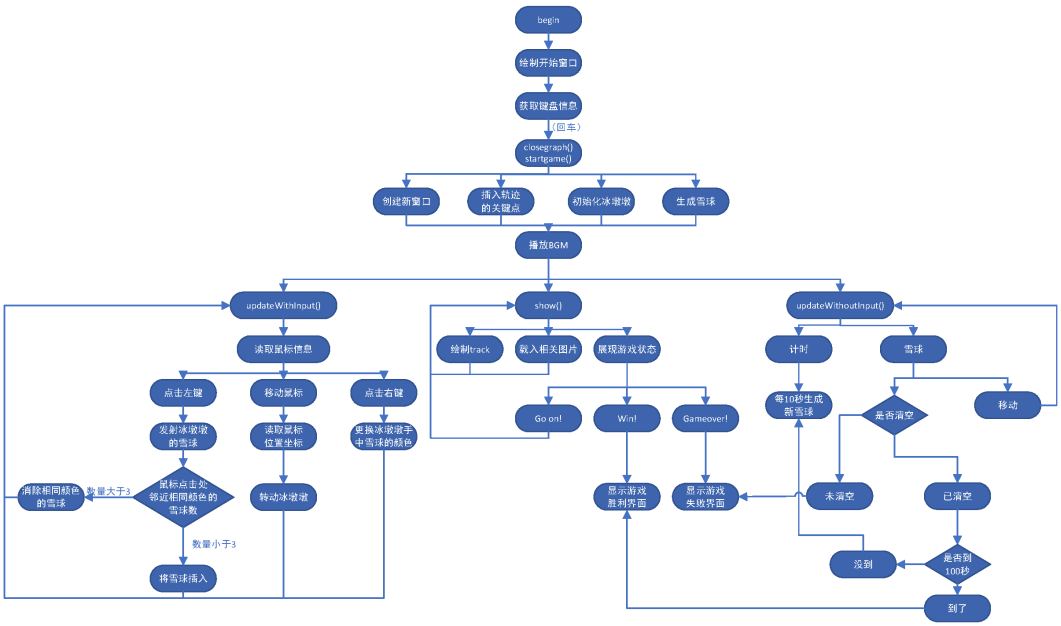
（2）单元测试

（3）模型部分

（4）验证

四、主要流程

（一）整体流程



（二）类视图

Point//轨迹点

x//点的横坐标
y//点的纵坐标
Point(float x0, float y0)
//构造函数
~Point()//析构函数

Ball //雪球类

Point center;//球心坐标
float radius;//雪球半径
int colorId;//雪球的颜色代码
int indexInPath;
//雪球位置在移动轨道中的序号
int direction;//雪球的移动方向
void draw() //画出雪球
void movetoIndexInPath(Path path)
//将雪球定位到Path的allPoints中的
indexInPath序号位置
void initiate(Path path)
//把雪球放在自定义的轨迹起点
void changeIndexbyDirection(Path path)
//让雪球沿着自定义轨迹移动

Path//雪球运动轨迹

vector<Point> keyPoints;
//记录轨迹上的所有点
float sampleInterval
//计算点的间隔
vector<Point> allPoints
//用于雪球移动的关键点
void getAllpoints()
//获得雪球移动的轨迹点
void draw() //画出轨迹
~Path() //析构函数

Cannon // 冰墩墩类

IMAGE im; // 冰墩墩图片
IMAGE im_rotate;// 冰墩墩旋转时的图片
float x, y; // 冰墩墩的中心坐标
Ball ball; // 冰墩墩的专属雪球
float angle; // 旋转角度
void draw() // 绘制相关元素
void setBallPosition()// 生成冰墩墩雪球的坐标
void updateWithMouseMOVE(int mx, int my)
// 让冰墩墩跟随鼠标摇摆
void updateWithRButtonDown()
//当鼠标右键点击时，改变雪球的颜色

（三）核心代码

1.游戏界面绘制

```
#include<graphics.h>
#define WIDTH 1300
#define HEIGHT 910
initgraph(WIDTH, HEIGHT);//开设一个窗口
cleardevice();//清屏
```

2.路径确定

```
void getAllpoints() //获得雪球移动的轨迹点
{
    int i = 0;
    for (i = 0; i < keyPoints.size() - 1; i++)
    {
        float xd = keyPoints[i + 1].x - keyPoints[i].x;
        float yd = keyPoints[i + 1].y - keyPoints[i].y;
        float length = sqrt(xd * xd + yd * yd);
        int num = length / sampleInterval;
```

```
            for (int j = 0; j < num; j++)
            {
                float x_sample = keyPoints[i].x + j * xd / num;
                float y_sample = keyPoints[i].y + j * yd / num;
                allPoints.push_back(Point(x_sample, y_sample));
            }
        }
        allPoints.push_back(Point(keyPoints[i].x, keyPoints[i].y));
    }
    for (int i = 0; i < keyPoints.size() - 1; i++)
    {
        line(keyPoints[i].x, keyPoints[i].y, keyPoints[i + 1].x, keyPoints[i + 1].y);
    }
    //为轨迹类添加一些关键点，用于自定义轨迹的构建
    path.keyPoints.push_back(Point(63, 368));
    path.keyPoints.push_back(Point(63, 755));
    path.keyPoints.push_back(Point(186, 847));
    path.keyPoints.push_back(Point(911, 847));
    path.keyPoints.push_back(Point(911, 733));
    path.keyPoints.push_back(Point(376, 733));
    path.keyPoints.push_back(Point(265, 658));
    path.keyPoints.push_back(Point(265, 265));
    path.keyPoints.push_back(Point(319, 196));
    path.keyPoints.push_back(Point(1038, 196));
    path.keyPoints.push_back(Point(1100, 271));
    path.keyPoints.push_back(Point(1122, 840));
    path.keyPoints.push_back(Point(1244, 840));
    path.keyPoints.push_back(Point(1239, 125));
    path.keyPoints.push_back(Point(1150, 58));
    path.keyPoints.push_back(Point(190, 63));
    path.sampleInterval = Radius / 5;
    path.getAllpoints();
```

3.雪球生成

```
void draw()  //画出雪球
{
    setlinecolor(colors[colorId]);//设置边框颜色
    setfillcolor(colors[colorId]);//设置内部颜色
    fillcircle(center.x, center.y, radius);//以(x,y)为圆心，以radius为半径画圆代表雪球
}
void movetoIndexInPath(Path path)//将雪球定位到Path的allPoints中的indexInPath序号位置
{
    center = path.allPoints[indexInPath];
```

```
}
void initiate(Path path) //把雪球放在自定义的轨迹起点
{
    radius = Radius;
    indexInPath = 0; // 初始化雪球的位置序号
    direction = 0;//初始雪球的速度
    movetoIndexInPath(path); // 移动到轨迹上面的对应序号位置
    colorId = rand() % ColorNum;//随机生成雪球的颜色
}
for (int i = 0; i < 10; i++)
{
    Ball ball;
    ball.initiate(path);//安放雪球到相应位置
    ball.indexInPath = i * (2 * ball.radius / path.sampleInterval);
    ball.movetoIndexInPath(path);
    balls.push_back(ball);
}
```

4.冰墩墩生成及旋转

```
void draw() // 绘制相关元素
{
    rotateimage(&im_rotate, &im, angle, RGB(241, 247, 252), false, true);//旋转冰墩墩
    putimage(x - im.getwidth() / 2, y - im.getheight() / 2, &im_rotate); // 显示旋转后的冰墩墩
    ball.draw(); // 绘制冰墩墩的雪球
}
void setBallPosition() // 生成冰墩墩雪球的坐标
{
    ball.center.x = x + 100 * cos(angle) + 38;
    ball.center.y = y + 100 * sin(angle) - 20;
}
void updateWithMouseMOVE(int mx, int my) // 让冰墩墩跟随鼠标摇摆
{
    float xs = mx - x;
    float ys = my - y;
    float length = sqrt(xs * xs + ys * ys);
    if (length > 4)
    {
        angle = atan2(-ys, xs);
        ball.center.x = x + 100 * xs / length + 38;
        ball.center.y = y + 100 * ys / length - 20;
    }
}
```

```
// 冰墩墩初始化
cannon.im = im_bdd; // 冰墩墩图片
cannon.angle = 0; // 初始角度
cannon.x = 500;  // 中心点坐标
cannon.y = 350;
cannon.ball.radius = Radius - 3; // 冰墩墩雪球的半径
cannon.ball.colorId = rand() % ColorNum; // 冰墩墩雪球颜色
cannon.setBallPosition(); // 设置冰墩墩雪球的位置坐标
```

5.雪球按照既定轨迹的移动

```
void changeIndexbyDirection(Path path)//让雪球沿着自定义轨迹移动
{
    if (direction == 1 && indexInPath + 1 < path.allPoints.size())
    {
        indexInPath++;
    }
    else if (direction == -1 && indexInPath - 1 >= 0)
    {
        indexInPath--;
    }
}
```

6.起始界面到游戏主界面的跳转

```
void begin()//游戏开始界面
{
    initgraph(WIDTH, HEIGHT);//开设一个窗口
    cleardevice();//清屏
    loadimage(&im_login, "login.png");//载入游戏开始的图片
    putimage(0, 0, &im_login);
    int a = 0;
    a = _getch();//读取键盘输入信息

    if (a == 13)
    {
        closegraph();//关闭当前的游戏开始界面
        startgame();//进行游戏主界面的初始化
    }
}
```

7.游戏音效的加入

```
#include<mmsystem.h>
```

```
#include<windows.h>
#pragma comment(lib,"Winmm.lib")
mciSendString(_T("open game_music.mp3"), NULL, 0, NULL);
mciSendString(_T("play game_music.mp3"), NULL, 0, NULL);
mciSendString(_T("close game_music.mp3"), NULL, 0, NULL);
```

五、代码测试

1.游戏开始

2.游戏运行界面

3. 雪球的消除

4. 冰墩墩的旋转

5. 游戏胜利


Zouma
南開大學
Nankai University
WIN!

6. 游戏失败


Zouma
南開大學
Nankai University
GAMEOVER!

六、收获

（一）加深了对 C++面向对象编程的理解。

（二）养成了写代码量较大的项目时随手写注释的习惯。

（三）培养了自己遇到相关问题动手去查询的习惯。

（四）学会了 C++图形化编程中图形绘制函数的使用。

要加入头文件#include<graphics.h>

1.生成图形窗口

initgraph(int WIDTH, int HEIGHT)
WIDTH：窗口宽度,HEIGHT：窗口高度，单位:像素

2.绘制点

void putpixel(int X, int Y, COLORREF COLOR)
X：X 轴坐标，Y：Y 轴坐标
COLOR：点颜色，支持宏、16进制颜色代码
注：窗口左上角坐标为(0, 0)，横轴为 X 轴，纵轴为 Y 轴

3.绘制线

setlinestyle(int LINE_STYLE, int THICKNESS)
setlinecolor(COLORREF COLOR);
line(int X1, int Y1, int X2, int Y2);
X1：第一点 X 轴坐标，Y1：第一点 Y 轴坐标
X2：第二点 X 轴坐标，Y2：第二点 Y 轴坐标
STYLE：线风格，常用如：0 实线，1 点线，2 中心线，3 断续线
THICKNESS：线宽

4.绘制矩形

空心矩形：void rectangle(int X1, int Y1, int X2, int Y2)
注：用法如 line 函数，不再赘述
实心有边框矩形：void fillrectangle(int X1, int Y1, int X2, int Y2)
实心无边框矩形：void solidrectangle(int X1, int Y1, int X2, int Y2)
注：实心矩形颜色使用 setfillcolor(COLORREF COLOR) 设置颜色
有边框矩形（包含空心矩形）使用 setlinecolor(COLORREF COLOR) 设置颜色

5.绘制圆

空心圆：void circle(int X, int Y, int R)
X：圆心 X 轴坐标，Y：圆心 Y 轴坐标，R：半径
实心有边框圆：void fillcircle(int X, int Y, int R)
实心无边框圆：void solidcircle(int X, int Y, int R)

6.文字输出

outtextxy(int X, int Y, char* STR)
X：文字输出位置 X 轴坐标，Y：文字输出位置 Y 轴坐标，STR：输出内容，支持明文字符串
设置字符风格：void settextstyle(int HEIGHT, int WIDTH, char* FONT)
HEIGHT：字符高，WIDTH：字符宽，为 0 则为自适应宽度，通常指定高度，宽度自适应，FONT：字体
设置字符颜色：void settextcolor(COLORREF COLOR)
设置字符背景色：void setbkcolor(COLORREF COLOR)
设置字符背景模式：void setbkmode(int MODE)
MODE：经测试值为 2 时背景为默认或者设置的字符背景色，其他值为透明

7.贴图操作

（1）声明 IMAGE 类型变量

IMAGE IMG;

（2）载入图片

void loadimage(IMAGE* IMG, LPCTSTR FILE, int WIDTH, int HEIGHT, bool RESIZE)
IMG：IMAGE类型变量
FILE：图片文件路径，char* 类型，如果有多层目录结构，间隔符号用 / 或者 \\
文件当前路径为.cpp 文件所在路径
WIDTH：载入图片的宽度
HEIGHT：载入图片的高度
RESIZE：缺省值为 false
注：指明了载入图片宽度和高度，图片将以缩放形式适应设定值

（3）输出图片

void putimage(int X, int Y, const IMAGE* IMG)
X：输出至屏幕位置 X 轴坐标；Y：输出至屏幕位置 Y 轴坐标；IMG：图片变量地址，如& img
void putimage(int X, int Y, int WIDTH, int HEIGHT, const IMAGE* IMG, int IMG_X, int IMG_Y)
X：输出至窗口起始位置 X 轴坐标；Y：输出至窗口起始位置 Y 轴坐标
WIDTH：指定绘制宽度；HEIGHT：指定绘制高度
IMG：图片变量地址，如& img
IMG_X：图片起始打印位置的 X 轴坐标。此时图片大小为载入后的大小
IMG_Y：图片起始打印位置的 Y 轴坐标。
函数解释：从图片的(IMG_X, IMG_Y) 位置开始截取 WIDTH 宽、HEIGHT 高的图形输出至窗口(X, Y) 位置

（五）了解了 C++编程中读取键鼠信息的代码实现方式。

1.键盘

//vs 中可以使用 _kbhit() 函数来获取键盘事件，使用时需要加入 conio.h 头文件

```
//例如
#include <conio.h>
#include <iostream>
using namespace std;
int main()
{
    int ch;
    while (1) {
        if (_kbhit()) {//如果有按键按下，则_kbhit()函数返回真
            ch = _getch();//使用_getch()函数获取按下的键值
            cout << ch;
            if (ch == 27) { break; }//当按下ESC时循环，ESC键的键值时27.
        }
    }
    system("pause");
    return 0;
}
```

键盘 Key Code 对照表

字母和数字键的键码值(keyCode)

按键	键码	按键	键码	按键	键码	按键	键码
A	65	J	74	S	83	1	49
B	66	K	75	T	84	2	50
C	67	L	76	U	85	3	51
D	68	M	77	V	86	4	52
E	69	N	78	W	87	5	53
F	70	O	79	X	88	6	54
G	71	P	80	Y	89	7	55
H	72	Q	81	Z	90	8	56
I	73	R	82	0	48	9	57

数字键盘上的键的键码值(keyCode)				功能键键码值(keyCode)			
按键	键码	按键	键码	按键	键码	按键	键码
0	96	8	104	F1	112	F7	118
1	97	9	105	F2	113	F8	119
2	98	*	106	F3	114	F9	120
3	99	+	107	F4	115	F10	121
4	100	Enter	108	F5	116	F11	122
5	101	-	109	F6	117	F12	123
6	102	.	110				
7	103	/	111				

控制键键码值(keyCode)							
按键	键码	按键	键码	按键	键码	按键	键码
BackSpace	8	Esc	27	Right Arrow	39	_-	189
Tab	9	Spacebar	32	Dw Arrow	40	.>	190
Clear	12	Page Up	33	Insert	45	/?	191
Enter	13	Page Down	34	Delete	46	`~	192
Shift	16	End	35	Num Lock	144	[{	219
Control	17	Home	36	;:	186	/|	220
Alt	18	Left Arrow	37	=+	187	]}	221
Cape Lock	20	Up Arrow	38	,<	188	'"	222

2. 鼠标

```
//vs中通过头文件 windows.h 来插入鼠标信息的结构体 MOUSEMSG
struct MOUSEMSG {
    UINT uMsg;      // 当前鼠标消息
    /*其中鼠标消息包含：
        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
    */
    bool mkCtrl;    // Ctrl 键是否按下
    bool mkShift;   // Shift 键是否按下
    bool mkLButton; // 鼠标左键是否按下
    bool mkMButton; // 鼠标中键是否按下
    bool mkRButton; // 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
};
//GetMouseMsg()仅能够在鼠标有动作的情况下获取鼠标信息，若鼠标无动作，则返回空值，m结构体中则不存在x，y值。
//如果想要实时获取鼠标位置信息时，需要引入 MouseHit()进行判断，存在消息才对 m 赋值。
```

（六）了解了 C++图形化编程中 BGM 的插入与播放。

mciSendString(LPCTSTR lpszCommand, LPTSTR lpszReturnString, UINT cchReturn, HANDLE hwndCallback);

lpszCommand：mci命令字符串

lpszReturnString：反馈信息缓冲区

cchReturn：缓冲区长度

hwndCallback：回调窗口句柄

注：请添加如下行避免mciSendString()无法解析

```
#include <mmsystem.h>
#pragma comment(lib, "winmm.lib")
```

mci常用命令：

1、open：打开设备

open DEVICE_NAME type DEVICE_TYPE alias DEVICE_ALIAS

DEVICE_NAME：设备名，通常是文件名

type DEVICE_TYPE：指明设备类型，常省略

alias DEVICE_ALIAS：设备别名，（理解为指代设备的变量名）可在其他mci命令中使用

2、play：设备播放

play DEVICE_ALIAS from POS1 to POS2 wait | repeat

DEVICE_ALIAS：播放的设备别名

from POS1 to POS2：从POS1磁道播放至POS2磁道，省from表从头播放，省to表播放至尾部

wait | repeat：播放模式，wait播放一次后命令返回，repeat循环播放

3、暂停，恢复，停止，关闭

pause DEVICE_ALIAS

resume DEVICE_ALIAS

stop DEVICE_ALIAS：停止播放非关闭

close DEVICE_ALIAS：关闭设备

例如：

```
#include<Windows.h>
#include<mmsystem.h>
#pragma comment(lib,"Winmm.lib")
using namespace std;
int main()
{
    mciSendString(_T("open game_music.mp3"), NULL, 0, NULL);
    mciSendString(_T("play game_music.mp3"), NULL, 0, NULL);
    mciSendString(_T("close game_music.mp3"), NULL, 0, NULL);
    return 0;
};
```