

**计算机学院本科生实验报告**

实验课程：程序设计实验

实验项目： 陨石撞飞机游戏设计与实现

指导教师： 张金区

开课时间： 2016年上半年

班级： 2015级实验班

**华 南 师 范 大 学**

计算机 学院 \_程序设计实验\_课实验报告

2015年级 2 班 实验日期 2016-06-19

姓名\_林伟业\_ 学号\_\_20152100121

教师评定\_\_\_\_\_\_\_\_\_\_ 实验题目：陨石撞飞机游戏设计与实现

报告提纲：用MFC开发陨石撞飞机的图形应用

摘要：

用MFC设计一个陨石撞飞机的平面游戏，陨石不断下落，飞机通过键盘的上下左右键移动以躲避陨石。当陨石撞到飞机时，显示游戏结束提示对话框。设计开始要对开发环境VC 6.0的熟悉，需要学会如何添加资源，添加类，编写函数控制位图，主要通过网上的学习，不断搜索才弄到大体。要设计飞机有3次被撞的机会，通过一个全局的变量设置飞机的生命值，没被撞一次，飞机的生命值减少1，直到飞机的生命值为0，游戏结束。有了3次机会，使得游戏的乐趣性增加，效果很满意。当游戏继续进行时，陨石下落的速度不断增加，速度的增加通过给陨石下落的速度设定一个函数，函数值经过时间的积累而增加，使得陨石下落的速度越来越快。陨石下落的速度不断增加，这给游戏添加不少挑战性，飞机存活的时间越长，速度越快，游戏的难度越大。

关键字：陨石，飞机，速度。

1. C++游戏的现状分析

中国C++程序员100％都是自学的。因为中国几乎没有象样的C++培训机构。大家都知道社会培训是赚钱为主要目的。能在那里成才是个大运气。也不象JAVA有SUN公司一手推行。中国大学教育里也根本没有C++教学。关于C/C++基础课的知识，只是让学生入门。但对于实际的C++软件工程应用，大学课本知识实践意义不大。最重要的是，大学里没有合格的C++老师。编程是门技术，叫手艺也行。而大学是搞科研的，多的是科学家，靠出PAPER教科凭职称为生。并不是技术员。没有10万行以上代码经验和成功软件项目经验的人，很难教出中高级C++程序员。

游戏制作业对C++程序员的要求是比较高的。比一般行业软件高很多。因为任何一个MMO都是个中大型软件项目。日使用人次达到几十万上百万，同时使用人数几十万在线的大型网络多人交互软件，集成了超过十几项专业领域编程技术的软件，想不大都不行。

另外，中高级C++程序员能力一般都很强。因为都是自学成才，这需要较高智商，很强的自学能力，解决问题能力，和克服困难的毅力。但是这种人才在游戏行业里人数并不多。抛开凤毛麟角具备整体大型软件架构能力的高级人才不谈。能够承担独立模块开发，具备独立解决问题的能力，写出在空间，性能，可维护性等几个方面达到要求的C++程序员，也是很难找到的。而具备这些能力，同时又具备一些专业领域技术的人（图形引擎，网络引擎，等），更加稀少。

所以，即便是大游戏公司，对于C++游戏程序员的需求，也是永远不能够得到满足。而已经在公司工作的C++程序员，不经过短则3个月，长达1年的锻炼，也无法真正胜任独当一面的C++游戏程序开发工作

首先，基本C++知识，C/C++运用技能，必须极其扎实。并不是游戏软件开发用到什么平时书本上很难得到的知识点或者技能点。游戏C++开发的知识体系和技能体系，利用现有C++书籍就可以满足。比如《C++ PAORGRAMMING LANGUAGE》《C++ PRIMER》《EFFECT C++》《MORE EFFECT C++》等。但是在开始游戏编程之前，把这些书本吃透并且实践熟练的人，非常难遇到。当然能够真做到这点的人，已经是个中高级C++程序员了。我曾经遇到过一个，毕业工作不久就去了GOOGLE

其次，对于面向对象的深入理解。是非常重要的。熟练掌握C/C++语法，编程技巧，只是满足了C++游戏程序员一部分要求。而面向对象观念的深刻理解，会为大型C++项目开发带来翻天覆地的变化。 有时候，在一个10人以上并行开发的C++游戏项目中，面向对象方法会成为救命稻草。《设计模式》以及《设计模式解析》相关的书籍，也是必修课。

最后，好技术人员的本质是自大的。这属于正常。但是如果这种习性影响了自身继续提高和团队合作。以及影响了其成为技术管理者和项目技术核心时，那么这种技术人员的自身价值会大打折扣。 如果早日越过自命不凡这一关，扩大眼界，继续在综合素质上提高。在技术管理能力，架构能力上提高，其自身价值才会继续增长。落到实处，就是待遇和身家增长。

由于游戏开发的本质是团队开发。任何时候，只有对一个团队有价值的技术人员，其本身的价值才有可能得以体现。

一个经验丰富的C/C++程序员。有过中国自主研发在线过百万游戏的成功项目经验。目前有网易出钱让其独立领导队伍搞引擎和游戏产品已经快3年。这样的一个履历，经验，本身就价值很高。他在任何公司都可以找到年薪几十万的工作。如果他具备管理才能。哪怕就是项目管理，或者技术管理能力。对于一个上市公司，他能做出的贡献，应该用百万/年量级来衡量。如果他能找到一个商业人士，负责舵手，并且自己具备一定商业能力，独立创业。吸引的天使风险投资也得几千万（否则做不了事，当然）。如果产品能做成，身家到达几千万是很自然的事。

1. 陨石撞飞机游戏总体设计

在游戏中玩家可以通过键盘的上下左右移动飞机躲避陨石，陨石从游戏界面的上部分不断下落，陨石的下落速度一开始是慢的，随着游戏的继续而玩家还没停止，陨石下落的速度越来越快，从而增加游戏的难度。在游戏上设置了2个模块，模块1是控制飞机的移动，主要是通过KEYDOWN函数实现，监测键盘的动作而处理相应事件，事件就是移动飞机。模块2是控制陨石的下落速度，通过设置两个计时器，一个计时器是控制陨石实际下落速度，另一个计时器是控制上一个计时器刷新的时间，从而避免一个计时器控制下落速度改变成指数型。陨石的数量可以自己控制，通过添加资源位图，陨石的形状也可以按照自己的爱好画。

1. 游戏实现的技术难点与关键算法说明

要熟悉VC6.0的使用才 能顺利开发。游戏中实现陨石下落速度的变化，一开始使用一个 计时器控制，使得陨石下落的速度成指数型的速度变化，慢的的时候，速度变化很慢，但快的时候变化很快，最后使得游戏一开始持续的时间长，到后面陨石下落速度很快的时候，游戏一下就结束了。解决这个问题我是通过使用两个计时器实现陨石下落速度 一个计时器是控制陨石实际下落速度，另一个计时器是控制上一个计时器刷新的时间，从而避免一个计时器控制下落速度改变成指数型。

1. 运行结果与测试

游戏测试结果良好，陨石和飞机除了颜色外还出现有白色的部分，飞机移动的距离适合，游戏较小，开启很快，CPU基本不占用，内存也只占用4236k

1. 总结与体会

对MFC开发还是不熟悉，需要多加练习，熟悉开发环境VC6.0和其中的各种函数使用。游戏的用户体验还有带提升，挑战性也可以继续添加。MFC好难，要学习比较苦难

附：源代码

文件planeView.h 和planeView.cpp 的源代码

// planeView.h : interface of the CPlaneView class

//

/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX\_PLANEVIEW\_H\_\_CF79B933\_ADF8\_4CEA\_8203\_ABF9DD6B3A72\_\_INCLUDED\_)

#define AFX\_PLANEVIEW\_H\_\_CF79B933\_ADF8\_4CEA\_8203\_ABF9DD6B3A72\_\_INCLUDED\_

#if \_MSC\_VER > 1000

#pragma once

#endif // \_MSC\_VER > 1000

class CPlaneView : public CView

{

protected: // create from serialization only

CPlaneView();

DECLARE\_DYNCREATE(CPlaneView)

// Attributes

public:

CPlaneDoc\* GetDocument();

CBitmap m\_plane; // 飞机

CBitmap m\_stone1, m\_stone2, m\_stone3,m\_stone4, m\_stone5, m\_stone6,m\_stone7, m\_stone8, m\_stone9; // 陨石

int planex, planey; // 飞机坐标

int stone1x, stone1y; // 陨石1坐标

int stone2x, stone2y; // 陨石2坐标

int stone3x, stone3y; // 陨石3坐标

int stone4x, stone4y; // 陨石4坐标

int stone5x, stone5y; // 陨石5坐标

int stone6x, stone6y; // 陨石6坐标

int stone7x, stone7y; // 陨石7坐标

int stone8x, stone8y; // 陨石8坐标

int stone9x, stone9y; // 陨石9坐标

int speed; //陨石初始下落速度

// Operations

public:

// Overrides

// ClassWizard generated virtual function overrides

//{{AFX\_VIRTUAL(CPlaneView)

public:

virtual void OnDraw(CDC\* pDC); // overridden to draw this view

virtual BOOL PreCreateWindow(CREATESTRUCT& cs);

protected:

virtual BOOL OnPreparePrinting(CPrintInfo\* pInfo);

virtual void OnBeginPrinting(CDC\* pDC, CPrintInfo\* pInfo);

virtual void OnEndPrinting(CDC\* pDC, CPrintInfo\* pInfo);

//}}AFX\_VIRTUAL

// Implementation

public:

virtual ~CPlaneView();

#ifdef \_DEBUG

virtual void AssertValid() const;

virtual void Dump(CDumpContext& dc) const;

#endif

protected:

// Generated message map functions

protected:

//{{AFX\_MSG(CPlaneView)

afx\_msg void OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags);

afx\_msg void OnTimer(UINT nIDEvent);

afx\_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);

//}}AFX\_MSG

DECLARE\_MESSAGE\_MAP()

private:

void DrawBitmap(CDC \*pDC, CBitmap &oBitmap, int oldx, int oldy, int newx, int newy);

};

#ifndef \_DEBUG // debug version in planeView.cpp

inline CPlaneDoc\* CPlaneView::GetDocument()

{ return (CPlaneDoc\*)m\_pDocument; }

#endif

/////////////////////////////////////////////////////////////////////////////

//{{AFX\_INSERT\_LOCATION}}

// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX\_PLANEVIEW\_H\_\_CF79B933\_ADF8\_4CEA\_8203\_ABF9DD6B3A72\_\_INCLUDED\_)

// planeView.cpp : implementation of the CPlaneView class

//

#include "stdafx.h"

#include "plane.h"

#include "planeDoc.h"

#include "planeView.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

/////////////////////////////////////////////////////////////////////////////

// CPlaneView

IMPLEMENT\_DYNCREATE(CPlaneView, CView)

BEGIN\_MESSAGE\_MAP(CPlaneView, CView)

//{{AFX\_MSG\_MAP(CPlaneView)

ON\_WM\_KEYDOWN()

ON\_WM\_TIMER()

ON\_WM\_CREATE()

//}}AFX\_MSG\_MAP

// Standard printing commands

ON\_COMMAND(ID\_FILE\_PRINT, CView::OnFilePrint)

ON\_COMMAND(ID\_FILE\_PRINT\_DIRECT, CView::OnFilePrint)

ON\_COMMAND(ID\_FILE\_PRINT\_PREVIEW, CView::OnFilePrintPreview)

END\_MESSAGE\_MAP()

/////////////////////////////////////////////////////////////////////////////

// CPlaneView construction/destruction

void CPlaneView::DrawBitmap(CDC \*pDC, CBitmap &oBitmap, int oldx, int oldy, int newx, int newy)

// 将位图对象oBitmap,以(x,y)为坐标显示在设备CDC上

{

// 擦去旧位图

RECT rect; // 矩形结构

BITMAP bm; // 位图结构

oBitmap.GetObject(sizeof(BITMAP), &bm); // 得到位图结构

rect.left = oldx; // rect左上角x坐标

rect.top = oldy; // rect左上角y坐标

rect.right = rect.left + bm.bmWidth; // rect右下角x坐标

rect.bottom = rect.top + bm.bmHeight; // rect右下角y坐标

CBrush brush(GetSysColor(COLOR\_WINDOW));

// 得到窗口系统色的画刷 GetSysColor为SDK函数,前面不能加this->

pDC->FillRect(&rect, &brush); // 擦去原来的位图

// 重绘新位图

CDC memDC; // 内存上的设备

memDC.CreateCompatibleDC(pDC); // 生成与pDC兼容的内存设备memDC

CBitmap \*pOldBitmap = memDC.SelectObject(&oBitmap);

// 将memDC与oBitmap对象相关

// BITMAP bm; // 位图结构

// oBitmap.GetObject(sizeof(BITMAP), &bm); // 得到位图结构

pDC->BitBlt(newx, newy, bm.bmWidth, bm.bmHeight, &memDC, 0, 0, SRCCOPY); // 显示位图

memDC.SelectObject(pOldBitmap); // 恢复memDC原来关联的位图对象

}

CPlaneView::CPlaneView()

{

// TODO: add construction code here

this->planex = 100; this->planey = 100; // 飞机初始位置

this->m\_plane.LoadBitmap(IDB\_BITMAP1); // 装载飞机位图

this->stone1x = rand() % 500; this->stone1y = 10; // 陨石1初始位置

this->m\_stone1.LoadBitmap(IDB\_BITMAP2); // 装载陨石1位图

this->stone2x = rand() % 500; this->stone2y = 10; // 陨石2初始位置

this->m\_stone2.LoadBitmap(IDB\_BITMAP3); // 装载陨石2位图

this->stone3x = rand() % 500; this->stone3y = 10; // 陨石3初始位置

this->m\_stone3.LoadBitmap(IDB\_BITMAP4); // 装载陨石3位图

this->stone4x = rand() % 500; this->stone4y = 10; // 陨石4初始位置

this->m\_stone4.LoadBitmap(IDB\_BITMAP5); // 装载陨石4位图

this->stone5x = rand() % 500; this->stone5y = 10; // 陨石5初始位置

this->m\_stone5.LoadBitmap(IDB\_BITMAP6); // 装载陨石5位图

this->stone6x = rand() % 500; this->stone6y = 10; // 陨石6初始位置

this->m\_stone6.LoadBitmap(IDB\_BITMAP7); // 装载陨石6位图

this->stone7x = rand() % 500; this->stone7y = 10; // 陨石7初始位置

this->m\_stone7.LoadBitmap(IDB\_BITMAP8); // 装载陨石1位图

this->stone8x = rand() % 500; this->stone8y = 10; // 陨石8初始位置

this->m\_stone8.LoadBitmap(IDB\_BITMAP9); // 装载陨石8位图

this->stone9x = rand() % 500; this->stone9y = 10; // 陨石9初始位置

this->m\_stone9.LoadBitmap(IDB\_BITMAP10); // 装载陨石9位图

this->speed = 500;//初始陨石下落速度

}

CPlaneView::~CPlaneView()

{

}

BOOL CPlaneView::PreCreateWindow(CREATESTRUCT& cs)

{

// TODO: Modify the Window class or styles here by modifying

// the CREATESTRUCT cs

return CView::PreCreateWindow(cs);

}

/////////////////////////////////////////////////////////////////////////////

// CPlaneView drawing

void CPlaneView::OnDraw(CDC\* pDC)

{

CPlaneDoc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

// TODO: add draw code for native data here

DrawBitmap(pDC, m\_plane, planex, planey, planex, planey);// 绘制飞机

}

/////////////////////////////////////////////////////////////////////////////

// CPlaneView printing

BOOL CPlaneView::OnPreparePrinting(CPrintInfo\* pInfo)

{

// default preparation

return DoPreparePrinting(pInfo);

}

void CPlaneView::OnBeginPrinting(CDC\* /\*pDC\*/, CPrintInfo\* /\*pInfo\*/)

{

// TODO: add extra initialization before printing

}

void CPlaneView::OnEndPrinting(CDC\* /\*pDC\*/, CPrintInfo\* /\*pInfo\*/)

{

// TODO: add cleanup after printing

}

/////////////////////////////////////////////////////////////////////////////

// CPlaneView diagnostics

#ifdef \_DEBUG

void CPlaneView::AssertValid() const

{

CView::AssertValid();

}

void CPlaneView::Dump(CDumpContext& dc) const

{

CView::Dump(dc);

}

CPlaneDoc\* CPlaneView::GetDocument() // non-debug version is inline

{

ASSERT(m\_pDocument->IsKindOf(RUNTIME\_CLASS(CPlaneDoc)));

return (CPlaneDoc\*)m\_pDocument;

}

#endif //\_DEBUG

/////////////////////////////////////////////////////////////////////////////

// CPlaneView message handlers

void CPlaneView::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)

{

// TODO: Add your message handler code here and/or call default

CClientDC ClientDC(this); // 得到当前客户区域设备

int oldx, int oldy, int newx, int newy;

oldx = planex; oldy = planey; // 旧飞机位置

if (nChar == VK\_UP)

{ // 按↑

newx = oldx;

newy = oldy - 30;

}

else if (nChar == VK\_LEFT)

{ // 按←

newx = oldx - 30;

newy = oldy;

}

else if (nChar == VK\_RIGHT)

{ // 按→

newx = oldx + 30;

newy = oldy;

}

else if (nChar == VK\_DOWN)

{ // 按↓

newx = oldx;

newy = oldy + 30;

}

DrawBitmap(&ClientDC, m\_plane, oldx, oldy, newx, newy);

// 绘制飞机

planex = newx; planey = newy; // 新飞机位置

CView::OnKeyDown(nChar, nRepCnt, nFlags);

}

void CPlaneView::OnTimer(UINT nIDEvent)

{

// TODO: Add your message handler code here and/or call default

switch(nIDEvent)

{

case 1:

{

CClientDC ClientDC(this); // 得到当前客户区域设备

RECT ClientRect; // 矩形结构

this->GetClientRect(&ClientRect); // 得到当前客户区域矩形

int oldx, oldy, newx, newy; // 新旧位置

oldx = stone1x; oldy = stone1y; // 旧石头1位置

newx = oldx; newy = oldy + 10; // 新石头1位置

if (newy > ClientRect.bottom)

{ // 重新生成石头1新位置

newx = rand() % 1000; newy = 0;

}

DrawBitmap(&ClientDC, m\_stone1, oldx, oldy, newx, newy);// 绘制石头1

stone1x = newx; stone1y = newy; // 新石头1位置

oldx = stone2x; oldy = stone2y; // 旧石头2位置

newx = oldx; newy = oldy + 13; // 新石头2位置

if (newy > ClientRect.bottom)

{ // 重新生成石头2新位置

newx = rand() % 1000; newy = 0;

}

DrawBitmap(&ClientDC, m\_stone2, oldx, oldy, newx, newy);// 绘制石头2

stone2x = newx; stone2y = newy; // 新石头2位置

oldx = stone3x; oldy = stone3y; // 旧石头3位置

newx = oldx; newy = oldy + 16; // 新石头3位置

if (newy > ClientRect.bottom)

{ // 重新生成石头3新位置

newx = rand() % 1000; newy = 0;

}

DrawBitmap(&ClientDC, m\_stone3, oldx, oldy, newx, newy);// 绘制石头3

stone3x = newx; stone3y = newy; // 新石头3位置

oldx = stone4x; oldy = stone4y; // 旧石头4位置

newx = oldx; newy = oldy + 10; // 新石头4位置

if (newy > ClientRect.bottom)

{ // 重新生成石头4新位置

newx = rand() % 1000; newy = 0;

}

DrawBitmap(&ClientDC, m\_stone4, oldx, oldy, newx, newy);// 绘制石头4

stone4x = newx; stone4y = newy; // 新石头4位置

oldx = stone5x; oldy = stone5y; // 旧石头5位置

newx = oldx; newy = oldy + 13; // 新石头5位置

if (newy > ClientRect.bottom)

{ // 重新生成石头5新位置

newx = rand() % 1000; newy = 0;

}

DrawBitmap(&ClientDC, m\_stone5, oldx, oldy, newx, newy);// 绘制石头5

stone5x = newx; stone5y = newy; // 新石头5位置

oldx = stone6x; oldy = stone6y; // 旧石头6位置

newx = oldx; newy = oldy + 16; // 新石头6位置

if (newy > ClientRect.bottom)

{ // 重新生成石头6新位置

newx = rand() % 1000; newy = 0;

}

DrawBitmap(&ClientDC, m\_stone6, oldx, oldy, newx, newy);// 绘制石头6

stone6x = newx; stone6y = newy; // 新石头6位置

oldx = stone7x; oldy = stone7y; // 旧石头7位置

newx = oldx; newy = oldy + 10; // 新石头7位置

if (newy > ClientRect.bottom)

{ // 重新生成石头7新位置

newx = rand() % 1000; newy = 0;

}

DrawBitmap(&ClientDC, m\_stone7, oldx, oldy, newx, newy);// 绘制石头7

stone7x = newx; stone7y = newy; // 新石头7位置

oldx = stone8x; oldy = stone8y; // 旧石头8位置

newx = oldx; newy = oldy + 13; // 新石头8位置

if (newy > ClientRect.bottom)

{ // 重新生成石头8新位置

newx = rand() % 1000; newy = 0;

}

DrawBitmap(&ClientDC, m\_stone8, oldx, oldy, newx, newy);// 绘制石头8

stone8x = newx; stone8y = newy; // 新石头8位置

oldx = stone9x; oldy = stone9y; // 旧石头9位置

newx = oldx; newy = oldy + 16; // 新石头9位置

if (newy > ClientRect.bottom)

{ // 重新生成石头9新位置

newx = rand() % 1000; newy = 0;

}

DrawBitmap(&ClientDC, m\_stone9, oldx, oldy, newx, newy);// 绘制石头9

stone9x = newx; stone9y = newy; // 新石头9位置

break;

}

case 2:

{

SetTimer(1,speed,NULL);//重置定时器1

if(speed > 300)

speed -=10;//陨石下落速度加快

else if (speed > 200)

speed -=5;

else if (speed > 100)

speed -= 2;

else

speed -= 1;

break;

}

default:

break;

}

CView::OnTimer(nIDEvent);

}

int CPlaneView::OnCreate(LPCREATESTRUCT lpCreateStruct)

{

if (CView::OnCreate(lpCreateStruct) == -1)

return -1;

// TODO: Add your specialized creation code here

this->SetTimer(1, 500, NULL); // 设备定时器触发的间隔为200毫秒

this->SetTimer(2,1000,NULL);

return 0;

}

最后附上自己开发的程序源代码。本部分长度不限