**面向对象方法程序设计与实践报告**

**题 目 4399狙击小日本**

**序 号 16**

**学 号 20242081177**

**姓 名 王斯民**

**任课教师 谢玲**

**成 绩**

大连理工大学软件学院

2025年7月

**目录**

[评分细则及标准 2](#_Toc143947979)

[1 技术调研报告 3](#_Toc143947980)

[1.1 第一周学习总结（除去代码，不少于1000字） 3](#_Toc143947981)

[1.1.1 内容简介 3](#_Toc143947982)

[1.1.2 难点和解决办法 3](#_Toc143947983)

[1.1.3 学习案例 3](#_Toc143947984)

[1.2 第二周学习总结 3](#_Toc143947985)

[1.2.1 内容简介 3](#_Toc143947986)

[1.2.2 难点和解决办法 3](#_Toc143947987)

[1.2.3 学习案例 4](#_Toc143947988)

[2 项目开发报告 5](#_Toc143947989)

[2.1 项目简介 5](#_Toc143947990)

[2.2 需求分析 5](#_Toc143947991)

[2.2.1 功能模块 5](#_Toc143947992)

[2.2.2 需求说明 6](#_Toc143947993)

[2.3 系统设计 6](#_Toc143947994)

[2.3.1 界面设计（UI） 6](#_Toc143947995)

[2.3.2 流程设计 6](#_Toc143947996)

[2.3.3 类设计 8](#_Toc143947997)

[2.4 系统实现 8](#_Toc143947998)

[2.4.1 实现效果 8](#_Toc143947999)

[2.4.2 核心代码 9](#_Toc143948000)

[2.5 系统测试 11](#_Toc143948001)

[2.6 项目总结 11](#_Toc143948002)

**评分细则及标准**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **考察项目** | **总分** | **评分细则（总分40分）** | **分数** | **得分** |
| **技术调研（第一周）** | 20分 | 调研报告 | 10分 |  |
| 平时表现 | 10分 |
| **技术调研（第二周）** | 20分 | 调研报告 | 10分 |  |
| 平时表现 | 10分 |
| **项目开发评分（总分60分）** | | | |  |
| 问题规模 | 10分 | 创新超额完成指定任务，工作量饱满 | 8-10分 |  |
| 基本完成指定任务，工作量一般 | 6-8分 |
| 指定任务未完成，工作量不足 | 0-5分 |
| 技术难度 | 10分 | 模型设计合理优化，采用合适的方法实现 | 8-10分 |  |
| 模型设计基本正确，采用较合适的方法实现 | 6-8分 |
| 模型设计存在问题，采用方法不合理 | 0-5分 |
| 实现程度 | 20分 | 系统实现完整，界面友好，测试全面无误 | 8-10分 |  |
| 系统实现完整，界面友好，存在少许错误 | 6-8分 |
| 系统实现不完整，界面不友好，存在错误 | 0-5分 |
| 报告质量 | 10分 | 报告完整、格式统一、结构清晰、图表正确 | 8-10分 |  |
| 报告较为规范、结构较为清晰、图表基本正确 | 6-8分 |
| 内容不完整不规范、结构不清晰、图表有错误 | 0-5分 |
| 项目汇报 | 10分 | 阐述清晰准确，回答问题准确到位 | 8-10分 |  |
| 阐述基本准确，回答问题基本准确 | 6-8分 |
| 阐述不够清晰完整，回答问题不准确 | 0-5分 |
| 最终得分 | 100分 |  | | |

相关材料（电子报告及项目源码压缩包）的百度网盘链接：

# 1 技术调研报告

## 1.1 第一周学习总结

### 1.1.1 内容简介

本周主要学习了 EasyX 绘图库的基础知识及开发环境搭建，包括以下几个方面：

1. **开发环境搭建**
   * 在 Windows 平台下安装 Visual Studio，并配置仅“使用 C++ 的桌面开发”工作负载。
   * 下载并安装 EasyX Graphics Library，了解其安装位置及与 Visual Studio 的集成方式。
2. **EasyX 基本概念**
   * **绘图窗口与设备**：掌握 initgraph 初始化绘图窗口，了解 EX\_NOMINIMIZE、EX\_NOCLOSE、EX\_SHOWCONSOLE 等窗口属性标志；熟悉 IMAGE 对象的概念及 SetWorkingImage 的使用。
   * **坐标体系**：区分物理坐标（以像素为单位，原点在左上角）和逻辑坐标（以点为单位，可通过 setorigin、setaspectratio 改变原点和比例）。
   * **颜色表示**：理解预定义常量、十六进制数、RGB 宏和 HSL/HSV 转换四种方式，通过 setlinecolor、setfillcolor 及填充函数设置颜色。
3. **绘图函数**
   * 掌握 EasyX 提供的 33 个基本绘图函数，包括圆形、矩形、圆角矩形、扇形、多边形、线条、贝塞尔曲线、区域填充及像素操作等，并通过示例代码熟悉各函数参数和用法。
4. **文字输出**
   * 熟悉 outtextxy、drawtext 输出文字的方法，了解 settextcolor、settextstyle、gettextstyle、textwidth、textheight 等文字样式和度量函数，并通过 LOGFONT 结构体自定义字体属性。
5. **双缓冲绘图**
   * 学习使用 BeginBatchDraw、FlushBatchDraw 和 EndBatchDraw 实现双缓冲，掌握帧率控制和屏幕闪烁消除的方法。
6. **随堂练习**
   * 完成了绘制五子棋棋盘、动态时钟等练习，加深对循环、时间函数、图形绘制和文本输出的综合运用。

### 1.1.2 难点和解决办法

在学习过程中，遇到了以下主要困难与对应解决方法：

1. **开发环境配置烦琐**
   * **问题**：Visual Studio 安装包体积较大，安装和重启后仍需手动配置 include 和 lib 路径；EasyX 安装后识别不到 VS 版本。
   * **解决**：参考 EasyX 官方文档和社区经验，确保在 VS “属性→VC++ 目录” 中正确添加 …\include 和 …\lib\x64（或 x86）路径；安装后重启 Visual Studio，并在“生成日志”中查看链接器是否正确载入 EasyXa.lib / EasyXw.lib。
2. **坐标系切换与变换理解困难**
   * **问题**：同时理解物理坐标与逻辑坐标、以及如何利用 setorigin 和 setaspectratio 实现坐标平移、缩放和翻转，需要一定的空间想象能力。
   * **解决**：通过设置不同的原点和缩放系数练习绘制同一图形（如圆形、矩形），并在注释中标注各步坐标变化；绘制坐标轴辅助线，直观对比不同坐标系下图形位置和形状。
3. **图形填充与边界处理**
   * **问题**：floodfill 填充有时会出现“漏边”或“溢出”现象，需指定正确的边界颜色和填充类型。
   * **解决**：在填充前先用 setlinecolor 明确绘制边框，用 solidcircle、fillpolygon 等填充函数验证参数效果；对比 FLOODFILLBORDER 和 FLOODFILLSURFILL 不同的填充机制，选择合适类型。
4. **双缓冲时序控制与帧率计算**
   * **问题**：在动画循环中计算每帧耗时并进行精确补时，对 Windows API GetTickCount 和 Sleep 配合使用不够熟练。
   * **解决**：阅读 MSDN 文档并参考案例，在循环内记录 beginTime、endTime，计算 elapsedTime 后用 Sleep(max(0, targetFrameTime – elapsedTime)) 保持帧率；测试时打印日志验证实际帧间隔。

### 1.1.3 学习案例

 **第一个课后习题**

#include <conio.h>  
#include <graphics.h>  
#include <string>  
using namespace std;  
  
int main()  
{  
 constexpr auto board\_lines=15;  
 constexpr auto cell\_size=40;  
 constexpr auto margin=20;  
 constexpr int width=margin\*2+cell\_size\*(board\_lines-1);  
 constexpr int height=width;  
 POINT star\_points[]={  
 {3,3},{11,3},{7,7},{3,11},{11,11}  
 };  
  
 initgraph(width,height);  
 setbkcolor(RGB(205,133,63));  
 cleardevice();  
 setlinecolor(BLACK);  
 for(auto i=0;i<board\_lines;i++)  
 {  
 const int x=margin+i\*cell\_size;  
 line(x,margin,x,margin+(board\_lines-1)\*cell\_size);  
 line(margin,x,margin+(board\_lines-1)\*cell\_size,x);  
 }  
 for(auto i=0;i<2;i++)  
 {  
 line(margin+i,margin+i,margin+(board\_lines-1)\*cell\_size-i,margin+i);  
 line(margin+i,margin+(board\_lines-1)\*cell\_size-i,  
 margin+(board\_lines-1)\*cell\_size-i,margin+(board\_lines-1)\*cell\_size-i);  
 line(margin+i,margin+i,margin+i,margin+(board\_lines-1)\*cell\_size-i);  
 line(margin+(board\_lines-1)\*cell\_size-i,margin+i,  
 margin+(board\_lines-1)\*cell\_size-i,margin+(board\_lines-1)\*cell\_size-i);  
 }  
 setfillcolor(BLACK);  
 for(const auto&[x, y]:star\_points)  
 {  
 constexpr auto starRadius=6;  
 const int cx=margin+x\*cell\_size;  
 const int cy=margin+y\*cell\_size;  
 solidcircle(cx,cy,starRadius);  
 }  
  
 settextstyle(18,0,"Courier");  
 LOGFONT f;  
 gettextstyle(&f);  
 f.lfQuality=ANTIALIASED\_QUALITY;  
 f.lfOutPrecision=OUT\_TT\_PRECIS;  
 settextstyle(&f);  
 setbkmode(TRANSPARENT);  
 settextcolor(BLACK);  
  
 for(auto i=0;i<board\_lines;i++)  
 {  
 const int y\_pos=margin+(board\_lines-1-i)\*cell\_size;  
 string numStr=to\_string(i+1);  
 RECT r={margin-25,y\_pos-8,margin-5,y\_pos+8};  
 drawtext(numStr.c\_str(),&r,DT\_RIGHT|DT\_VCENTER|DT\_SINGLELINE);  
 }  
  
 constexpr int boardBottom=margin+(board\_lines-1)\*cell\_size;  
 for(auto i=0;i<board\_lines;i++)  
 {  
 const auto letter=static\_cast<char>('A'+i);  
 string basic\_string(1,letter);  
 const int x\_pos=margin+i\*cell\_size;  
 RECT r={x\_pos-10,boardBottom+5,x\_pos+10,boardBottom+25};  
 drawtext(basic\_string.c\_str(),&r,DT\_CENTER|DT\_VCENTER|DT\_SINGLELINE);  
 }  
  
 \_getch();  
 closegraph();  
 return 0;  
}

 **第二个课后习题**

#include <conio.h>  
#include <graphics.h>  
#include <bits/stdc++.h>  
using namespace std;  
  
double to\_rad(const double deg)  
{  
 return deg\*M\_PI/180.0;  
}  
  
void draw\_clockface(const int cx,const int cy,const int r)  
{  
 setlinecolor(BLACK);  
 setlinestyle(PS\_SOLID,2);  
 circle(cx,cy,r);  
  
 for(auto i=0;i<60;i++)  
 {  
 const double ang=to\_rad(i\*6);  
 const int len=i%5==0?25:20;  
 const int x1=cx+static\_cast<int>((r-10)\*sin(ang));  
 const int y1=cy-static\_cast<int>((r-10)\*cos(ang));  
 const int x2=cx+static\_cast<int>((r-len)\*sin(ang));  
 const int y2=cy-static\_cast<int>((r-len)\*cos(ang));  
 setlinestyle(PS\_SOLID,i%5==0?3:1);  
 line(x1,y1,x2,y2);  
 }  
  
 settextstyle(30,0,"微软雅黑");  
 LOGFONT f;  
 gettextstyle(&f);  
 f.lfQuality=ANTIALIASED\_QUALITY;  
 f.lfOutPrecision=OUT\_TT\_PRECIS;  
 settextstyle(&f);  
 setbkmode(TRANSPARENT);  
 settextcolor(BLACK);  
 for(auto h=1;h<=12;h++)  
 {  
 const double angDeg=h\*30.0-90;  
 const double ang=to\_rad(angDeg);  
 const int tx=cx+static\_cast<int>((r-40)\*cos(ang));  
 const int ty=cy+static\_cast<int>((r-40)\*sin(ang));  
  
 char buf[3];  
 sprintf(buf,"%d",h);  
  
 const int tw=textwidth(buf);  
 const int th=textheight(buf);  
 outtextxy(tx-tw/2,ty-th/2,buf);  
 }  
}  
  
void draw\_hands(const int cx,const int cy,const int r,  
 const double h,const double m,const double s)  
{  
 const double aH=to\_rad(h\*30+m\*0.5);  
 setlinecolor(BLACK);  
 setlinestyle(PS\_SOLID,6);  
 line(cx,cy,  
 cx+static\_cast<int>((r-80)\*sin(aH)),  
 cy-static\_cast<int>((r-80)\*cos(aH)));  
  
 const double aM=to\_rad(m\*6+s\*6.0/60);  
 setlinestyle(PS\_SOLID,4);  
 line(cx,cy,  
 cx+static\_cast<int>((r-50)\*sin(aM)),  
 cy-static\_cast<int>((r-50)\*cos(aM)));  
  
 const double aS=to\_rad(s\*6);  
 setlinecolor(RED);  
 setlinestyle(PS\_SOLID,2);  
 const int sx=cx+static\_cast<int>((r-30)\*sin(aS));  
 const int sy=cy-static\_cast<int>((r-30)\*cos(aS));  
 line(cx,cy,sx,sy);  
  
 const int bx=cx-static\_cast<int>(20\*sin(aS));  
 const int by=cy+static\_cast<int>(20\*cos(aS));  
 line(cx,cy,bx,by);  
  
 setfillcolor(BLACK);  
 solidcircle(cx,cy,5);  
 setfillcolor(WHITE);  
 solidcircle(cx,cy,2);  
}  
  
int main()  
{  
 constexpr auto W=500,H=500;  
 initgraph(W,H);  
 setbkcolor(WHITE);  
  
 BeginBatchDraw();  
 while(!\_kbhit())  
 {  
 constexpr auto R=200;  
 SYSTEMTIME st;  
 GetLocalTime(&st);  
 const HWND hWnd=GetHWnd();  
 char timeStr[9];  
 sprintf(timeStr,"%02d:%02d:%02d",st.wHour,st.wMinute,st.wSecond);  
 SetWindowText(hWnd,timeStr);  
 const double smoothSec=st.wSecond+st.wMilliseconds/1000.0;  
  
 cleardevice();  
  
 draw\_clockface(W/2,H/2,R);  
 draw\_hands(W/2,H/2,R,st.wHour%12,st.wMinute,smoothSec);  
  
 FlushBatchDraw();  
 Sleep(10);  
 }  
 EndBatchDraw();  
 closegraph();  
 return 0;  
}

## 1.2 第二周学习总结

### 1.2.1 内容简介

进阶学习主要围绕 EasyX 的图像处理与消息（事件）处理两大模块：

#### 一、图像处理

1. **基础图像读写**
   * loadimage：将外部文件（BMP、JPEG、PNG、GIF、TIF 等）加载到 IMAGE 对象或直接绘图窗口，可指定缩放尺寸与是否调整设备大小。
   * saveimage：将当前绘图设备或 IMAGE 对象保存为各种格式文件。
2. **图像绘制与变换**
   * putimage：将 IMAGE 对象绘制到目标设备，支持指定绘制区域和源图像区域，并可使用三元光栅操作码（如 SRCCOPY、SRCAND、SRCPAINT）实现透明贴图。
   * rotateimage：按指定弧度旋转 IMAGE 内容，可设置背景色、自动调整尺寸和高质量插值。
   * Resize：调整 IMAGE 对象或绘图窗口尺寸。
3. **像素级操作**
   * GetImageBuffer：获取设备的像素缓存指针，可直接读写像素、绘制渐变、翻转图像等。
   * GetImageHDC：获取设备的 GDI 句柄，配合 Windows GDI 函数（如 TransparentBlt、AlphaBlend、Ellipse、LineTo 等）实现高级效果。
4. **动画制作**
   * 通过按帧加载一系列图片，并在主循环中按固定间隔切换显示，结合双缓冲技术 (BeginBatchDraw／FlushBatchDraw／EndBatchDraw) 实现平滑动画。

#### 二、消息处理

1. **消息获取**
   * getmessage（阻塞式）与 peekmessage（非阻塞式），可按类别（鼠标、键盘、字符、窗口）过滤消息，了解消息缓冲区长度与刷新。
2. **鼠标事件**
   * 通过 WM\_MOUSEMOVE、WM\_LBUTTONDOWN、WM\_LBUTTONUP、WM\_RBUTTONDOWN 等消息，实现拖拽、点击绘图、跟随效果和退出控制。
3. **键盘事件**
   * 利用 WM\_KEYDOWN、WM\_KEYUP 和 msg.vkcode（虚拟键码）识别常用按键，控制角色移动、暂停／继续、调整速度，并处理组合键（如 Ctrl + 鼠标）。
4. **综合练习**
   * 结合鼠标与键盘输入，实现“鼠标+键盘绘图”与“弹球”小游戏，掌握事件循环、状态存储与边界检测等技术。

### 1.2.2 难点和解决办法

 **大尺寸图片加载与性能优化**

* **难点**：直接使用 loadimage(NULL, ...) 将大型图片加载到窗口时，CPU 占用高且出现卡顿。
* **解决**：先将图片加载到 IMAGE 对象，再按需 putimage；对同一张大图多次绘制时，使用双缓冲并尽量避免在循环内频繁调用 loadimage。

 **透明贴图的多种实现方式**

* **难点**：使用三元光栅操作码进行透明合成，需自行制作掩码／前景图，流程复杂；而 Windows GDI 方式（TransparentBlt、AlphaBlend）又需要掌握 HDC、BLENDFUNCTION 等细节。
* **解决**：优先采用 AlphaBlend 结合 PNG 的内建 alpha 通道，实现“一步到位”的透明效果；仅在特殊需求下（如只需二值透明）使用掩码法与 SRCAND／SRCPAINT。

 **像素级缓存操作易越界**

* **难点**：直接写 GetImageBuffer 得到的指针数组时，若不注意宽高，一次性写入会导致内存越界或图像失真。
* **解决**：始终使用 IMAGE 对象的 getwidth()、getheight() 确定像素总数，并在循环中以 y \* width + x 的方式索引；对外部传入的坐标或尺寸进行合法性检查。

 **消息循环与高帧率控制**

* **难点**：在消息处理与绘制之间平衡，若先绘制再处理消息，会出现输入延迟；若先处理再绘制，则动画帧率不稳。
* **解决**：采用“每帧先 peekmessage 处理所有消息，再进行数据更新和绘图”模式，并记录 GetTickCount 计算每帧耗时，结合 Sleep 精确补时，保证 60 FPS 同时快速响应输入。

### 1.2.3 学习案例

 **第一个课后习题**

#include <conio.h>  
#include <graphics.h>  
#include <windows.h>  
  
COLORREF drawColor=WHITE;  
  
int main()  
{  
 initgraph(1024,768);  
 setbkcolor(BLACK);  
 cleardevice();  
  
 ExMessage msg{};  
  
 while(true)  
 {  
 if(peekmessage(&msg, EM\_MOUSE|EM\_KEY))  
 {  
 if(msg.message==WM\_LBUTTONDOWN||msg.message==WM\_RBUTTONDOWN)  
 {  
 const bool ctrlPressed=GetAsyncKeyState(VK\_CONTROL)&0x8000;  
  
 setfillcolor(drawColor);  
 setlinecolor(drawColor);  
  
 if(msg.message==WM\_LBUTTONDOWN)  
 {  
 const int size=ctrlPressed?20:10;  
 fillrectangle(msg.x-size/2,msg.y-size/2,msg.x+size/2,msg.y+size/2);  
 }  
 else if(msg.message==WM\_RBUTTONDOWN)  
 {  
 const int radius=ctrlPressed?20:10;  
 fillcircle(msg.x,msg.y,radius);  
 }  
 }  
  
 if(msg.message==WM\_KEYDOWN)  
 {  
 switch(msg.vkcode)  
 {  
 case 'C':  
 cleardevice();  
 break;  
 case 'R':  
 drawColor=RED;  
 break;  
 case 'G':  
 drawColor=GREEN;  
 break;  
 case 'B':  
 drawColor=BLUE;  
 break;  
 case 'W':  
 drawColor=WHITE;  
 break;  
 case VK\_ESCAPE:  
 closegraph();  
 return 0;  
 default:;  
 }  
 }  
 }  
 }  
}

 **第二个课后习题**

#include <conio.h>  
#include <graphics.h>  
#include <windows.h>  
#include <bits/stdc++.h>  
  
constexpr auto WIN\_WIDTH=1024;  
constexpr auto WIN\_HEIGHT=768;  
  
constexpr auto BALL\_RADIUS=10;  
int ball\_x=WIN\_WIDTH/2;  
int ball\_y=WIN\_HEIGHT/2;  
auto ball\_vx=4;  
auto ball\_vy=4;  
  
constexpr auto PADDLE\_WIDTH=100;  
constexpr auto PADDLE\_HEIGHT=10;  
int paddle\_x=(WIN\_WIDTH-PADDLE\_WIDTH)/2;  
constexpr int PADDLE\_Y=WIN\_HEIGHT-30;  
constexpr auto PADDLE\_SPEED=8;  
  
auto isPaused=true;  
  
void draw\_frame()  
{  
 cleardevice();  
 solidcircle(ball\_x,ball\_y,BALL\_RADIUS);  
 rectangle(paddle\_x,PADDLE\_Y,  
 paddle\_x+PADDLE\_WIDTH,  
 PADDLE\_Y+PADDLE\_HEIGHT);  
}  
  
void update\_ball()  
{  
 ball\_x+=ball\_vx;  
 ball\_y+=ball\_vy;  
  
 if(ball\_x-BALL\_RADIUS<=0||ball\_x+BALL\_RADIUS>=WIN\_WIDTH)  
 {  
 ball\_vx=-ball\_vx;  
 ball\_x+=ball\_vx;  
 }  
 if(ball\_y-BALL\_RADIUS<=0)  
 {  
 ball\_vy=-ball\_vy;  
 ball\_y+=ball\_vy;  
 }  
}  
  
void check()  
{  
 if(ball\_y+BALL\_RADIUS>=PADDLE\_Y&&  
 ball\_y+BALL\_RADIUS<=PADDLE\_Y+PADDLE\_HEIGHT&&  
 ball\_x>=paddle\_x&&  
 ball\_x<=paddle\_x+PADDLE\_WIDTH)  
 {  
 ball\_vy=-ball\_vy;  
 ball\_y+=ball\_vy;  
 }  
}  
  
void handle\_input()  
{  
 static auto pDown=false;  
  
 if(GetAsyncKeyState('P')&0x8000)  
 {  
 if(!pDown)  
 {  
 isPaused=!isPaused;  
 pDown=true;  
 }  
 }  
 else  
 {  
 pDown=false;  
 }  
  
 if(!isPaused)  
 {  
 if(GetAsyncKeyState('A')&0x8000||GetAsyncKeyState(VK\_LEFT)&0x8000)  
 {  
 paddle\_x=std::max(paddle\_x-PADDLE\_SPEED,0);  
 }  
 if(GetAsyncKeyState('D')&0x8000||GetAsyncKeyState(VK\_RIGHT)&0x8000)  
 {  
 paddle\_x=std::min(paddle\_x+PADDLE\_SPEED,WIN\_WIDTH-PADDLE\_WIDTH);  
 }  
 }  
}  
  
int main()  
{  
 initgraph(WIN\_WIDTH,WIN\_HEIGHT);  
  
 BeginBatchDraw();  
 while(true)  
 {  
 handle\_input();  
 if(!isPaused)  
 {  
 update\_ball();  
 check();  
 }  
 draw\_frame();  
 FlushBatchDraw();  
  
 if(ball\_y-BALL\_RADIUS>WIN\_HEIGHT)  
 {  
 break;  
 }  
  
 Sleep(10);  
 }  
 EndBatchDraw();  
  
 settextstyle(40,0,"微软雅黑");  
 LOGFONT f;  
 gettextstyle(&f);  
 f.lfQuality=ANTIALIASED\_QUALITY;  
 f.lfOutPrecision=OUT\_TT\_PRECIS;  
 setbkmode(TRANSPARENT);  
 settextstyle(&f);  
 setcolor(WHITE);  
 const auto s="游戏结束";  
 const int tw=textwidth(s);  
 const int th=textheight(s);  
 const int tx=(WIN\_WIDTH-tw)/2;  
 const int ty=(WIN\_HEIGHT-th)/2;  
 outtextxy(tx,ty,s);  
  
 \_getch();  
 closegraph();  
 return 0;  
}

# 2 项目开发报告

## 2.1 项目简介

### 1. 游戏初始化系统

#### 1.1 DPI自适应

* **功能描述**：自动适配不同DPI设置的显示器
* **实现方式**：调用Windows API设置DPI感知上下文

void dpi\_awareness() {

if(auto lib=LoadLibraryA("user32.dll")) {

using Func=BOOL(WINAPI\*)(DPI\_AWARENESS\_CONTEXT);

if(auto fn=(Func)GetProcAddress(lib,"SetProcessDpiAwarenessContext"))

fn(DPI\_AWARENESS\_CONTEXT\_PER\_MONITOR\_AWARE\_V2);

FreeLibrary(lib);

}

}

#### 1.2 窗口创建

* **功能描述**：创建1600×900游戏窗口
* **特性**：
  + 窗口居中显示
  + 标题设置为"4399狙击小日本"

#### 1.3 资源加载

* **图像资源**：
  + 背景图片(bg.png)
  + 门元素(door.png)
  + 敌人模型(enemy.png)
  + 狙击镜遮罩(mask.png)
* **音频资源**：
  + 射击音效(shoot.mp3)
  + 背景音乐(bgm.mp3)
  + 爆头音效(headshot.mp3)
  + 音乐盒音效(mvp\_\*.mp3)

### 2. 视角控制系统

#### 2.1 鼠标视角控制

* **功能描述**：通过鼠标移动控制游戏视角
* **灵敏度调节**：支持0.1-5.0范围调节
* **实现方式**：

speed = std::clamp(speed+em.wheel/120\*0.1,0.1,5.0);

off\_x -= static\_cast<int>((p.x-WIN\_WIDTH/8)\*speed);

off\_y -= static\_cast<int>((p.y-WIN\_HEIGHT/8)\*speed);

#### 2.2 视角边界限制

* **功能描述**：防止视角移出场景边界
* **实现方式**：

wx=std::clamp(800-off\_x,450,1920);

wy=std::clamp(900+off\_y,340,1000);

### 3. 射击战斗系统

#### 3.1 射击机制

* **触发方式**：鼠标左键点击
* **音效反馈**：播放射击音效

mciSendString(\_T("play shoot from 0"),nullptr,0,nullptr);

#### 3.2 命中判定

* **头部命中**：
  + 条件：准星在敌人上半部分
  + 效果：播放爆头音效，显示"击中头部"
* **身体命中**：
  + 条件：准星在敌人下半部分
  + 效果：显示"击中身体"
* **判定代码**：

const bool hitX=wx>=ewx\_pos&&wx<=ewx\_pos+EN\_W;

const bool head=wy<=ewy\_pos&&wy>=ewy\_pos-EN\_W;

if(const bool body=wy<=ewy\_pos-EN\_W&&wy>=ewy\_pos-EN\_H;hitX&&(head||body))

#### 3.3 音乐盒系统

* **切换方式**：鼠标右键点击
* **功能**：循环切换不同MVP音乐
* **实现**：

void switch\_mvp() {

current\_mvp\_index=(current\_mvp\_index+1)%mvp\_names.size();

}

### 4. 敌人行为系统

#### 4.1 移动模式

* **基础移动**：从右向左匀速移动
* **重置机制**：移出屏幕后随机位置重生

ewx\_pos=ewx\_pos<=450?1370+rand()%4000:ewx\_pos-static\_cast<int>(1000\*dt);

### 5. 游戏状态系统

#### 5.1 暂停/继续

* **触发方式**：空格键
* **功能**：
  + 暂停时显示鼠标指针
  + 继续时隐藏指针并重置位置

if(GetAsyncKeyState(VK\_SPACE)&0x8000) {

paused^=1;

just\_unpaused=!paused;

}

#### 5.2 游戏退出

* **触发方式**：ESC键
* **功能**：安全关闭游戏窗口和资源

### 6. 渲染显示系统

#### 6.1 分层渲染

* **渲染顺序**：
  1. 背景层
  2. 敌人层
  3. 门层
  4. 狙击镜遮罩层

putimage(off\_x,off\_y,&bg);

putimage\_alpha(ex,ey,&enemy);

putimage\_alpha(off\_x,off\_y,&door);

putimage\_alpha(0,0,&mask);

#### 6.2 Alpha混合

* **功能描述**：实现透明效果渲染

inline void putimage\_alpha(int x, int y, IMAGE\* img) {

AlphaBlend(GetImageHDC(nullptr), x, y, img->getwidth(), img->getheight(),

GetImageHDC(img), 0, 0, img->getwidth(), img->getheight(),

{AC\_SRC\_OVER, 0, 255, AC\_SRC\_ALPHA});

}

#### 6.3 状态显示

* **显示内容**：
  + 当前音乐盒名称
  + 灵敏度数值
  + 实时FPS
* **位置**：窗口右上角

#### 6.4 操作提示

* **显示内容**：
  + 空格:暂停
  + ESC:退出
  + 滚轮:调灵敏度
  + 左键:开枪
  + 右键:切换音乐盒
* **位置**：窗口左上角

#### 6.5 击中提示

* **显示方式**：右侧渐消提示框
* **持续时间**：2秒

hit\_events.erase(std::remove\_if(hit\_events.begin(),hit\_events.end(),

[&](auto&e) {

return std::chrono::duration<double>(now2-e.time).count()>2.0;

}),

hit\_events.end());

### 7. 音频管理系统

#### 7.1 背景音乐

* **播放模式**：循环播放
* **音量设置**：400

mciSendString(\_T("setaudio bgm volume to 400"),nullptr,0,nullptr);

mciSendString(\_T("play bgm repeat from 0"),nullptr,0,nullptr);

#### 7.2 音效触发

| **事件类型** | **音效名称** | **音量设置** |
| --- | --- | --- |
| 射击 | shoot.mp3 | 1000 |
| 爆头 | headshot.mp3 | 1000 |
| 命中 | mvp音效 | 800 |

### 8. 输入处理系统

#### 8.1 鼠标输入

* **左键**：射击
* **右键**：切换音乐盒
* **滚轮**：调整灵敏度

#### 8.2 键盘输入

* **空格键**：暂停/继续
* **ESC键**：退出游戏

### 9. 性能管理系统

#### 9.1 帧率控制

* **功能**：实时计算并显示FPS
* **实现**：

if(++frame\_count,std::chrono::duration\_cast<std::chrono::milliseconds>(now-last\_time).count()>=1000)

{

fps=frame\_count;

frame\_count=0;

last\_time=now;

}

#### 9.2 批量绘制

* **功能**：减少画面闪烁
* **实现**：

BeginBatchDraw();

FlushBatchDraw();

EndBatchDraw();

## 三、特殊功能处理

### 1. 鼠标指针控制

* **问题**：EasyX无法直接隐藏鼠标
* **解决方案**：

SetCursor(nullptr);

while(ShowCursor(FALSE)>=0) {}

while(ShowCursor(TRUE)<0) {}

### 2. 坐标转换系统

* **世界坐标**：(off\_x, off\_y)
* **屏幕坐标**：(wx, wy)
* **转换关系**：

wx = 800 - off\_x;

wy = 900 + off\_y;

### 3. 击中事件管理

* **数据结构**：vector<HitEvent>
* **事件属性**：
  + 文本内容
  + 生成时间戳
* **生命周期**：2秒自动清除

## 四、游戏运行流程

1. 初始化DPI感知
2. 创建游戏窗口
3. 加载资源（图像/音频）
4. 扫描音乐盒文件
5. 进入主游戏循环：
   * 处理输入事件
   * 更新游戏状态
   * 计算视角偏移
   * 检测命中
   * 分层渲染场景
   * 显示状态信息
6. 退出时释放资源

## 2.2 需求分析

### 2.2.1 功能模块

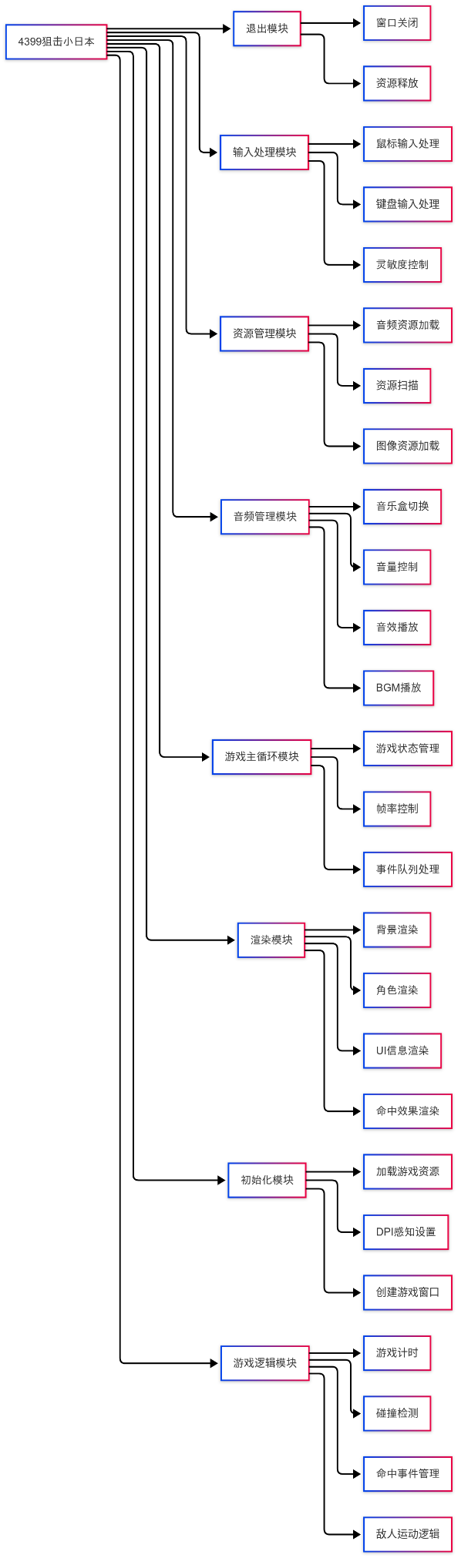


图2.1 功能模块图

### 2.2.2 需求说明

## 一、游戏概述

本游戏是一款基于EasyX图形库开发的第一人称射击游戏。玩家通过鼠标控制视角，狙击移动的敌人目标。游戏包含完整的视听系统、交互控制和状态显示，提供沉浸式的射击体验。

## 二、核心功能模块

## 2.3 系统设计

### 2.3.1 界面设计（UI）



图2.2 界面原型图

### 2.3.2 流程设计

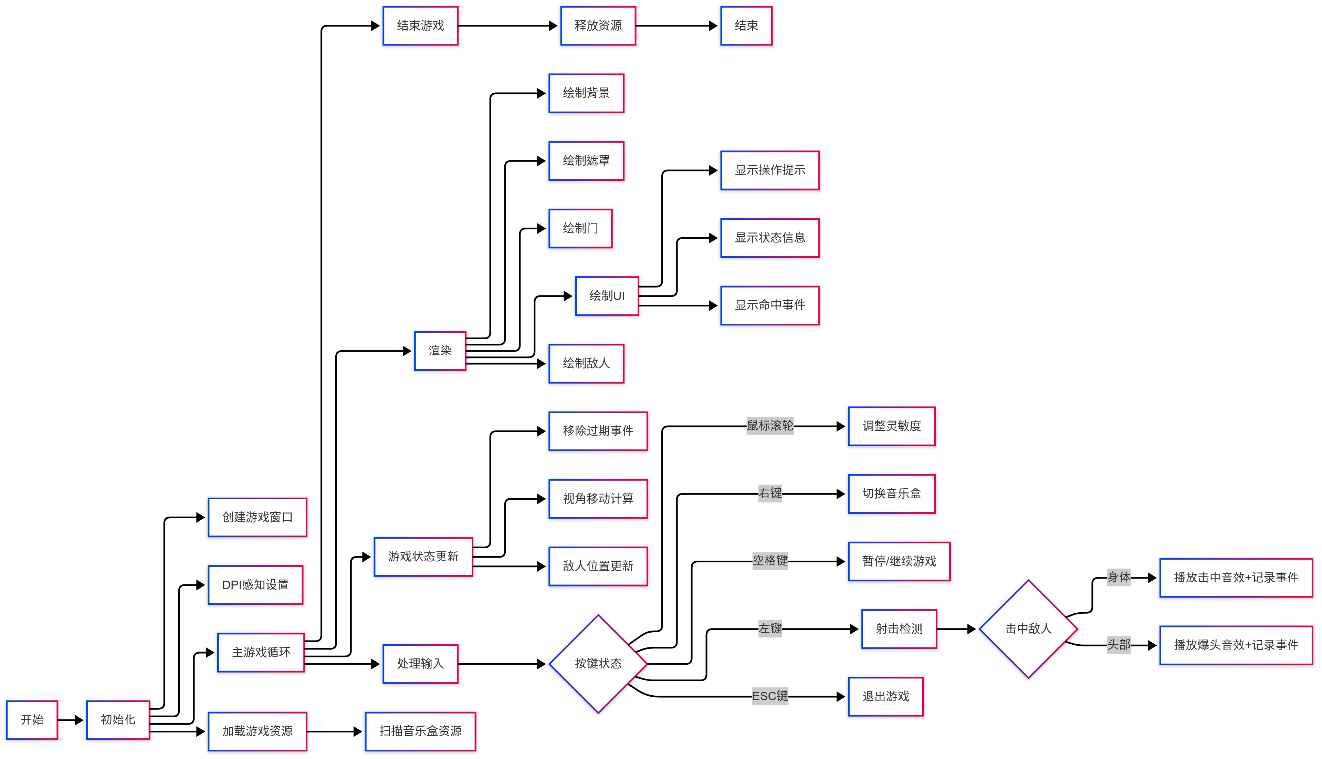


图2.3 系统流程图



### 2.3.3 类设计

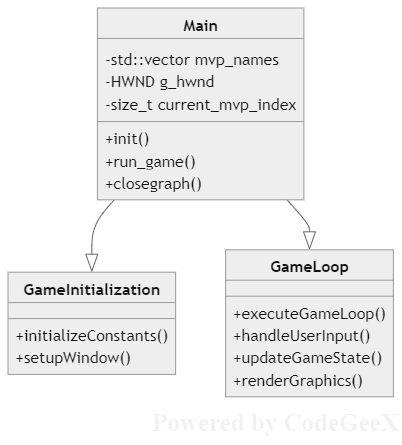


图2.4 类关系图

## 2.4 系统实现

### 2.4.1 实现效果



图2.5 游戏界面



图2.6 游戏击中反馈



图2.7 游戏镜头随机移动

### 2.4.2 核心代码

constexpr auto WIN\_WIDTH=1600;  
constexpr auto WIN\_HEIGHT=900;  
  
inline void putimage\_alpha(const int x,const int y,IMAGE\* img)  
{  
 const int w=img->getwidth();  
 const int h=img->getheight();  
 AlphaBlend(GetImageHDC(nullptr),x,y,w,h,  
 GetImageHDC(img),0,0,w,h,{AC\_SRC\_OVER,0,255,AC\_SRC\_ALPHA});  
}  
  
static IMAGE mask,door,bg,enemy;  
static auto running=true,paused=false,just\_unpaused=false;  
static std::vector<std::string> mvp\_names;  
static size\_t current\_mvp\_index=0;  
  
struct HitEvent  
{  
 std::string text;  
 std::chrono::high\_resolution\_clock::time\_point time;  
};  
  
static std::vector<HitEvent> hit\_events;  
static HWND g\_hwnd;  
  
void dpi\_awareness()  
{  
 if(const auto lib=LoadLibraryA("user32.dll"))  
 {  
 using Func=BOOL(WINAPI\*)(DPI\_AWARENESS\_CONTEXT);  
 if(const auto fn=reinterpret\_cast<Func>(GetProcAddress(lib,"SetProcessDpiAwarenessContext")))  
 fn(DPI\_AWARENESS\_CONTEXT\_PER\_MONITOR\_AWARE\_V2);  
 FreeLibrary(lib);  
 }  
}  
  
HWND create\_window()  
{  
 initgraph(WIN\_WIDTH,WIN\_HEIGHT);  
 const HWND h=GetHWnd();  
 const int sw=GetSystemMetrics(SM\_CXSCREEN);  
 const int sh=GetSystemMetrics(SM\_CYSCREEN);  
 SetWindowPos(h,nullptr,(sw-WIN\_WIDTH)/2,(sh-WIN\_HEIGHT)/2,  
 WIN\_WIDTH,WIN\_HEIGHT, SWP\_NOZORDER);  
 SetWindowText(h,"4399狙击小日本");  
 return h;  
}  
  
void scan\_mvp\_assets()  
{  
 mvp\_names.clear();  
 WIN32\_FIND\_DATAA fd;  
 if(const auto h=FindFirstFileA("assets\\mvp\_\*.mp3",&fd);h!=INVALID\_HANDLE\_VALUE)  
 {  
 do  
 {  
 std::string prefix="mvp\_",suffix=".mp3";  
 if(std::string fn=fd.cFileName;fn.rfind(prefix,0)==0&&fn.size()>prefix.size()+suffix.size())  
 mvp\_names.push\_back(fn.substr(prefix.size(),fn.size()-prefix.size()-suffix.size()));  
 }  
 while(FindNextFileA(h,&fd));  
 FindClose(h);  
 }  
 mvp\_names.insert(mvp\_names.begin(),"");  
 current\_mvp\_index=0;  
}  
  
void load\_assets()  
{  
 loadimage(&mask,"assets/mask.png",WIN\_WIDTH,WIN\_HEIGHT);  
 loadimage(&door,"assets/door.png",WIN\_WIDTH\*1.5,WIN\_HEIGHT\*1.5);  
 loadimage(&bg,"assets/bg.png",WIN\_WIDTH\*1.5,WIN\_HEIGHT\*1.5);  
 loadimage(&enemy,"assets/enemy.png");  
 mciSendString(\_T("open assets\\shoot.mp3 alias shoot"),nullptr,0,nullptr);  
 mciSendString(\_T("open assets\\bgm.mp3 alias bgm"),nullptr,0,nullptr);  
 mciSendString(\_T("open assets\\headshot.mp3 alias headshot"),nullptr,0,nullptr);  
 scan\_mvp\_assets();  
 mciSendString(\_T("setaudio bgm volume to 400"),nullptr,0,nullptr);  
 mciSendString(\_T("play bgm repeat from 0"),nullptr,0,nullptr);  
}  
  
void switch\_mvp()  
{  
 mciSendString(\_T("stop mvp"),nullptr,0,nullptr);  
 mciSendString(\_T("close mvp"),nullptr,0,nullptr);  
 current\_mvp\_index=(current\_mvp\_index+1)%mvp\_names.size();  
 if(!mvp\_names[current\_mvp\_index].empty())  
 {  
 TCHAR cmd[256];  
 \_stprintf(cmd, \_T("open assets\\mvp\_%s.mp3 alias mvp"),mvp\_names[current\_mvp\_index].c\_str());  
 mciSendString(cmd,nullptr,0,nullptr);  
 }  
}  
  
void handle\_input()  
{  
 static auto space\_down=false,esc\_down=false;  
 if(GetAsyncKeyState(VK\_SPACE)&0x8000)  
 {  
 if(!space\_down)  
 {  
 paused^=1;  
 just\_unpaused=!paused;  
 space\_down=true;  
 }  
 }  
 else  
 space\_down=false;  
 if(GetAsyncKeyState(VK\_ESCAPE)&0x8000)  
 {  
 if(!esc\_down)  
 {  
 running=false;  
 esc\_down=true;  
 }  
 }  
 else  
 esc\_down=false;  
}  
  
void display\_stats(const int fps,const double speed)  
{  
 settextcolor(GREEN);  
 settextstyle(30,0,"微软雅黑");  
 LOGFONT f;  
 gettextstyle(&f);  
 f.lfQuality=ANTIALIASED\_QUALITY;  
 f.lfOutPrecision=OUT\_TT\_PRECIS;  
 setbkmode(OPAQUE);  
 setbkcolor(BLACK);  
 settextstyle(&f);  
 TCHAR buf[128];  
 const char\* name=mvp\_names[current\_mvp\_index].empty()?"(无)":mvp\_names[current\_mvp\_index].c\_str();  
 \_stprintf(buf, \_T("当前音乐盒:《%s》 灵敏度:%.2f FPS:%d"),name,speed,fps);  
 SIZE sz;  
 GetTextExtentPoint32(GetImageHDC(nullptr),buf, \_tcslen(buf),&sz);  
 outtextxy(WIN\_WIDTH-sz.cx-10,10,buf);  
 settextcolor(WHITE);  
 for(auto i=0,y=10;i<7;++i,y+=30)  
 {  
 const TCHAR\* tips[]={  
 \_T("(easyx库重置了鼠标状态因此无法隐藏指针)"),  
 \_T(" 提示"),  
 \_T("空格:暂停"),  
 \_T("ESC:退出"),  
 \_T("滚轮:调灵敏度"),  
 \_T("左键:开枪"),  
 \_T("右键:切换音乐盒")  
 };  
 outtextxy(10,y,tips[i]);  
 }  
}  
  
void run\_game()  
{  
 srand(static\_cast<unsigned>(time(nullptr)));  
 BeginBatchDraw();  
 auto cursorVisible=false;  
 SetCursor(nullptr);  
 while(ShowCursor(FALSE)>=0) {}  
 POINT center{WIN\_WIDTH/8,WIN\_HEIGHT/8};  
 ClientToScreen(g\_hwnd,&center);  
 SetCursorPos(center.x,center.y);  
  
 int off\_x=-361,off\_y=-227;  
 int wx=800-off\_x,wy=900+off\_y;  
 auto ewx\_pos=1370;  
 using Clock=std::chrono::high\_resolution\_clock;  
 auto last\_time=Clock::*now*(),last\_frame=last\_time;  
 auto frame\_count=0,fps=0;  
 auto speed=1.0;  
 const int BG\_H=bg.getheight();  
 const int EN\_W=enemy.getwidth(),EN\_H=enemy.getheight();  
  
 while(true)  
 {  
 handle\_input();  
 if(!running)  
 {  
 break;  
 }  
  
 constexpr auto ewy\_pos=800;  
 auto now=Clock::*now*();  
 const float dt=std::chrono::duration<float>(now-last\_frame).count();  
 last\_frame=now;  
  
 ExMessage em{};  
 while(peekmessage(&em, EM\_MOUSE))  
 {  
 if(em.message==WM\_MOUSEWHEEL)  
 {  
 speed=std::clamp(speed+em.wheel/120\*0.1,0.1,5.0);  
 }  
 else if(em.message==WM\_RBUTTONDOWN)  
 {  
 switch\_mvp();  
 }  
 else if(em.message==WM\_LBUTTONDOWN&&!paused)  
 {  
 mciSendString(\_T("play shoot from 0"),nullptr,0,nullptr);  
 mciSendString(\_T("setaudio mvp volume to 800"),nullptr,0,nullptr);  
 const bool hitX=wx>=ewx\_pos&&wx<=ewx\_pos+EN\_W;  
 const bool head=wy<=ewy\_pos&&wy>=ewy\_pos-EN\_W;  
 if(const bool body=wy<=ewy\_pos-EN\_W&&wy>=ewy\_pos-EN\_H;hitX&&(head||body))  
 {  
 if(head)  
 {  
 mciSendString(\_T("play headshot from 0"),nullptr,0,nullptr);  
 hit\_events.push\_back({"击中头部",Clock::*now*()});  
 }  
 else  
 {  
 hit\_events.push\_back({"击中身体",Clock::*now*()});  
 }  
 mciSendString(\_T("play mvp from 0"),nullptr,0,nullptr);  
 }  
 }  
 }  
  
 if(paused&&!cursorVisible)  
 {  
 while(ShowCursor(TRUE)<0) {}  
 cursorVisible=true;  
 }  
 else if(!paused&&cursorVisible)  
 {  
 while(ShowCursor(FALSE)>=0) {}  
 cursorVisible=false;  
 }  
  
 if(++frame\_count,std::chrono::duration\_cast<std::chrono::milliseconds>(now-last\_time).count()>=1000)  
 {  
 fps=frame\_count;  
 frame\_count=0;  
 last\_time=now;  
 }  
  
 if(!paused)  
 {  
 cleardevice();  
 if(just\_unpaused)  
 {  
 SetCursorPos(center.x,center.y);  
 just\_unpaused=false;  
 continue;  
 }  
 POINT p;  
 GetCursorPos(&p);  
 ScreenToClient(g\_hwnd,&p);  
 off\_x-=static\_cast<int>((p.x-WIN\_WIDTH/8)\*speed);  
 off\_y-=static\_cast<int>((p.y-WIN\_HEIGHT/8)\*speed);  
 wx=std::clamp(800-off\_x,450,1920);  
 wy=std::clamp(900+off\_y,340,1000);  
 SetCursorPos(center.x,center.y);  
 off\_x=800-wx;  
 off\_y=wy-900;  
 ewx\_pos=ewx\_pos<=450?1370+rand()%4000:ewx\_pos-static\_cast<int>(1000\*dt);  
 const int ex=off\_x+ewx\_pos;  
 const int ey=off\_y+(BG\_H-ewy\_pos);  
 putimage(off\_x,off\_y,&bg);  
 putimage\_alpha(ex,ey,&enemy);  
 putimage\_alpha(off\_x,off\_y,&door);  
 putimage\_alpha(0,0,&mask);  
 }  
  
 display\_stats(fps,speed);  
  
 auto now2=Clock::*now*();  
 hit\_events.erase(std::remove\_if(hit\_events.begin(),hit\_events.end(),  
 [&](auto&e)  
 {  
 return std::chrono::duration<double>(now2-e.time).count()>2.0;  
 }),  
 hit\_events.end());  
 constexpr int startX=WIN\_WIDTH-200;  
 setlinecolor(YELLOW);  
 setbkmode(TRANSPARENT);  
 settextstyle(30,0,"微软雅黑");  
 for(size\_t i=0;i<hit\_events.size();++i)  
 {  
 constexpr auto startY=40;  
 auto&[text, time]=hit\_events[i];  
 const char\* txt=text.c\_str();  
 const size\_t len=text.length();  
 SIZE sz;  
 GetTextExtentPoint32(GetImageHDC(nullptr),txt,len,&sz);  
 const int boxW=sz.cx+12;  
 const int boxH=sz.cy+12;  
 const int y=startY+static\_cast<int>(i)\*(boxH+4);  
 rectangle(startX,y,startX+boxW,y+boxH);  
 outtextxy(startX+6,y+6,txt);  
 }  
 setbkmode(OPAQUE);  
  
 FlushBatchDraw();  
 }  
  
 EndBatchDraw();  
 if(!cursorVisible)  
 {  
 while(ShowCursor(TRUE)<0) {}  
 }  
}  
  
void init()  
{  
 dpi\_awareness();  
 g\_hwnd=create\_window();  
 load\_assets();  
}

## 2.5 系统测试

表2.1 系统测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例编号 | 测试项 | 测试步骤 | 输入信号 | 预期结果 | 测试结果 |
| 1 | 游戏启动 | 运行程序 | 无 | 显示1600×900游戏窗口，标题为"4399狙击小日本" | 通过 |
| 2 | 视角移动 | 移动鼠标 | 鼠标位移 | 游戏场景随鼠标移动方向平滑滚动 | 通过 |
| 3 | 灵敏度调整 | 滚动鼠标滚轮 | 滚轮上/下滚动 | 灵敏度按0.1步进增减(0.1-5.0范围) | 通过 |
| 4 | 射击命中判定 | 1. 将准星对准敌人 2. 点击鼠标左键 | 鼠标左键单击 | 播放射击音效，显示"击中头部/身体"提示 | 通过 |
| 5 | 音乐盒切换 | 点击鼠标右键 | 鼠标右键单击 | 切换MVP音乐并显示当前音乐盒名称 | 通过 |
| 6 | 暂停/继续功能 | 按下空格键 | 键盘空格键 | 游戏暂停/继续，鼠标指针显示/隐藏 | 通过 |
| 7 | 敌人移动 | 不进行任何操作，观察敌人 | 无 | 敌人从右向左持续移动 | 通过 |
| 8 | 击中事件显示 | 1. 击中敌人 2. 等待2秒 | 时间流逝 | "击中头部/身体"提示在2秒后自动消失 | 通过 |
| 9 | 游戏退出 | 按下ESC键 | 键盘ESC键 | 游戏窗口关闭，程序退出 | 通过 |
| 10 | 边界检测 | 将视角移动到场景边缘 | 鼠标持续向边界移动 | 场景停止滚动，不会超出地图边界 | 通过 |
| 11 | 音频系统 | 连续快速点击鼠标左键 | 多次鼠标左键单击 | 射击音效正常播放，无卡顿 | 通过 |
| 12 | 帧率显示 | 观察游戏窗口右上角 | 无 | 实时显示当前FPS值 | 通过 |
| 13 | 图层遮挡 | 1. 使敌人移动到门后 2. 射击门后敌人位置 | 鼠标左键单击 | 击中门后的敌人 | 通过 |
| 14 | 长时间运行 | 持续运行游戏10分钟 | 时间流逝 | 游戏稳定运行，无内存泄漏或崩溃 | 通过 |

## 2.6 项目总结

一、完成情况  
成功实现了基于EasyX库的第一人称射击游戏核心功能。主要完成的内容：

1. 搭建了1600×900分辨率的游戏窗口，并实现了DPI适配
2. 开发了鼠标控制视角移动系统，支持0.1-5.0灵敏度调节
3. 实现了射击命中判定系统，区分头部和身体命中效果
4. 创建了动态敌人移动系统，敌人从右向左移动
5. 完成分层渲染系统（背景/敌人/门/遮罩层）
6. 实现多通道音频系统（BGM/射击/爆头/MVP音效）
7. 设计音乐盒动态切换机制
8. 开发UI系统（实时FPS/灵敏度/音乐盒显示）
9. 实现击中提示系统（2秒渐消效果）
10. 操作指引界面

代码量329行，核心射击玩法完整。

二、技术难点与解决方案

1. DPI缩放问题：  
   问题：高DPI显示器导致元素错位  
   解决方案：通过dpi\_awareness()函数动态加载DPI感知上下文
2. 坐标系统问题：  
   问题：世界坐标与屏幕坐标转换困难  
   解决方案：建立双坐标系系统

（1）(off\_x, off\_y)表示世界偏移

（2）(wx, wy)表示视角中心

1. Alpha通道实现：  
   问题：EasyX原生Alpha支持不足  
   解决方案：自定义putimage\_alpha()函数

三、存在的不足

1. 性能问题：
   * 每帧全场景重绘，未实现优化
   * 音频播放缺乏缓冲机制，频繁触发时卡顿
   * 击中事件使用vector存储，未做对象池优化
2. 游戏设计缺陷：
   * 敌人移动采用线性模式，缺乏变化
   * 没有关卡进度和分数系统
3. 代码结构问题：
   * 缺乏错误处理机制（资源加载失败会导致崩溃）
4. 交互体验不足：
   * 灵敏度调整缺乏视觉反馈
   * 没有穿门伤害衰减
5. 平台限制：
   * 深度依赖Windows API
   * 无法跨平台运行
   * 仅支持1600×900特定分辨率