浙江大学

C++程 序 设 计

大 程 序 报 告

学生姓名 ：_____ 学号：__

2018~2019 春夏学期 2019 年 6 月 7 日

1、题目描述和题目要求

题目描述：设计和实现一款带有图形界面的游戏或其他应用。

题目要求：要求基于 graphics_lib，设计和实现图形界面，至少运用以下技术和知识：

1.菜单、按钮、鼠标、键盘、文件；

2.体现 c++的内容；

选题：**基于 FLTK 的 2048 游戏**

功能表对应要求如下：

考评项		分数	说明
菜单系统		10	本项目菜单系统包括**主界面菜单系统和子界面菜单系统**
图标工具栏		5	菜单中常用功能/命令，主界面菜单系统包括**与各个子界面的链接以及游戏使用说明和退出按钮**，共 5 个；子界面菜单系统包括**上下左右、重新开始、存档、读档、退出**等按钮，三个界面一共 24 个，其中，每个窗口的退出和存档读档可以通过按钮实现，也可以通过图标工具栏实现
快捷键		5	与菜单功能对应，**所有的按键都有快捷键，在界面中都有标注，在此不一一说明，**另外各个游戏界面和游戏说明界面可以按下快捷键 N 退出
状态信息栏		5	在每个级别的游戏窗口的顶部显示当前游戏的最高分数和游戏的状态（Begin、Game Continue、Game Over、Game Win、Restart）
功能（35）	2048 基本游戏（三个	20	实现 2048 三个级别的游戏的基本算法和基本功能，游戏可以重新开始

	等级）		
	GUI 界面	10	包括游戏主界面以及游戏成功或者失败的界面，并且可以实时看到当前时间
	游戏的随时存档与读档	5	游戏可以随时存档读档，三个级别的游戏互不冲突
模板		5	使用模板进行回调函数的调用
文件	文件读取	10	保存当前游戏的图元信息到文本文件（存档）；从文件中读取图形（读档），用户可以随时存档与读档
	多文件组织	5	系统程序必须采用多文件组织的方式，项目包含多个.h 和.cpp 每个类都对应一个.h 与.cpp，并且有一个集成实现游戏功能的命名空间 Gamegraph_lib
大程序报告	分析和设计	5	软件简介，功能结构，全局、函数及重要算法说明，源程序中功能、函数、文件的组织关系
	部署和运行	5	编译安装、运行测试、用户使用手册
	运行结果	5	效果展示
	分析	5	系统开发亮点和应用知识点总结
总分	100		

2、需求分析和项目简介

2048 是一款益智小游戏，可供休闲娱乐，Game_2048 为基于 FLTK 和 graph_lib 制作的 2048 游戏，在基本的 2048 游戏的基础上增加了中等难度和高等难度，一共三个级别，用户进入主界面后可以根据菜单选入不同的级别，菜单中还包含“About 2048”栏目对游戏进行简介，游戏建议开启键盘大写锁定用快捷键操作，每个界面都可以通过快捷键 N 进行返回和切换，支持用户随时存档读档，实时输出游戏最高分和游戏状态，十分方便

3、总体设计

3.1 功能模块设计

本游戏功能包括以下部分：

（1） 三个级别的 2048 游戏，分别为 4*4,6*6,8*8 的 2048 游戏，对应 Game_base、Game_medium、Game_hard

（2） 简单的用户使用说明界面

（3） 支持用户随时重新开始

（4） 支持用户随时存档读档，三个关卡不冲突

（5） 用户达到 2048 会弹出胜利窗口，但是用户仍然可以继续游戏

（6） 整个游戏所以按钮都支持快捷键，建议开启大写锁定体验，每个按钮的快捷键都在按钮的最后有说明，关闭每个游戏界面采用快捷键 N

功能模块包含：

（1）全局变量模块

（2）游戏功能实现模块（包括基本游戏功能、重新开始、存档读档）

（3）整个游戏主界面和子界面系统模块

（3）绘制游戏方框界面的模块

（4）用户胜利或失败的 GUI 模块

对应的文件组织：

一个全局变量头文件、

一个主要集成三个级别游戏实现和游戏呈现的命名空间 Gamegraph_lib（包含.h 和.cpp）、

一个用于绘制游戏方框界面的类 Rect_with_text（包含.h 和.cpp）、

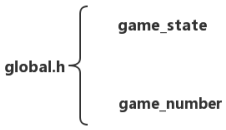
一个用于实现用户胜利或失败的 GUI 界面的类 Guiwindow(包含.h 和.cpp)、以及提供的 Graph.h、**Window.h（对该头文件和 cpp 进行了修改，后面会详细说明）**、Simple_window.h、GUI.h、Point.h、std_lib_facilities.h 以及对应的 cpp

各个功能的实现将在以下部分根据代码进行详细介绍

3.2 结构设计

3.2.1 全局变量头文件 global.h：

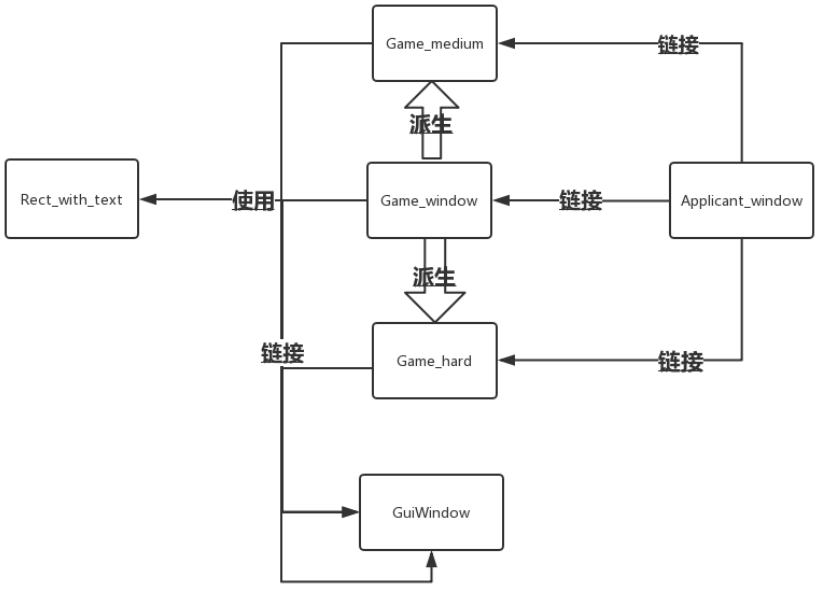
将游戏状态设为常量、游戏数字集合为一个枚举




```
const int GAME_OVER = 1;  //the state
const int GAME_WIN = 2;
const int GAME_CONTINUE = 3;

enum GameNum//game numbers
{
    Game_2 = 2,
    Game_4 = 4,
    Game_8 = 8,
    Game_16 = 16,
    Game_32 = 32,
    Game_64 = 64,
    Game_128 = 128,
    Game_256 = 256,
    Game_512 = 512,
    Game_1024 = 1024,
    Game_2048 = 2048,
};
```

3.2.2 类结构

类结构的设计如下图所示：



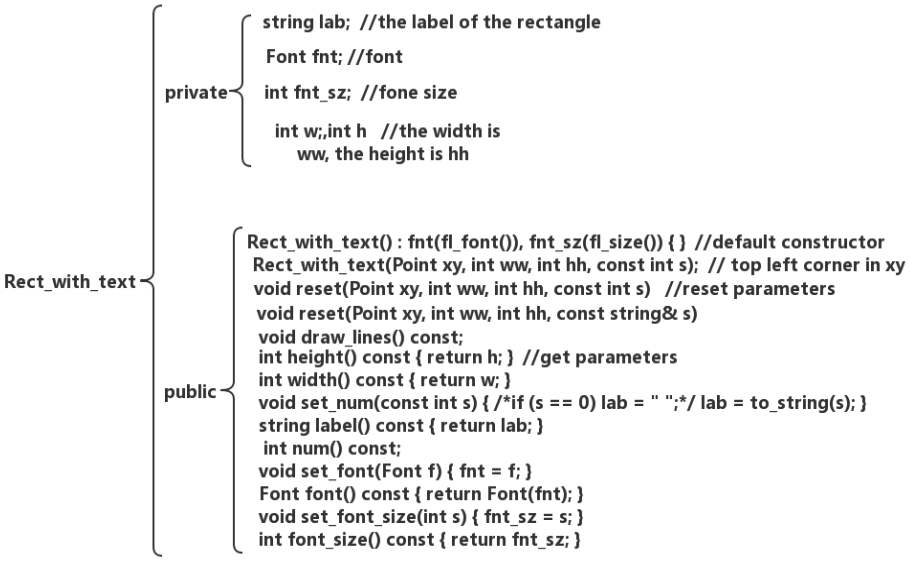
3.2.2.1 用于绘制游戏方框界面（即带有数字的方框）的类 Rect_with_text（包含.h 和.cpp）：

①包含的头文件

```
#include "Graph.h"
#include "Window.h"
#include "Simple_window.h"
#include <string>
#include <cstring>
```

②类声明

```
//Draw our game grid
class Rect_with_text : public Shape
```



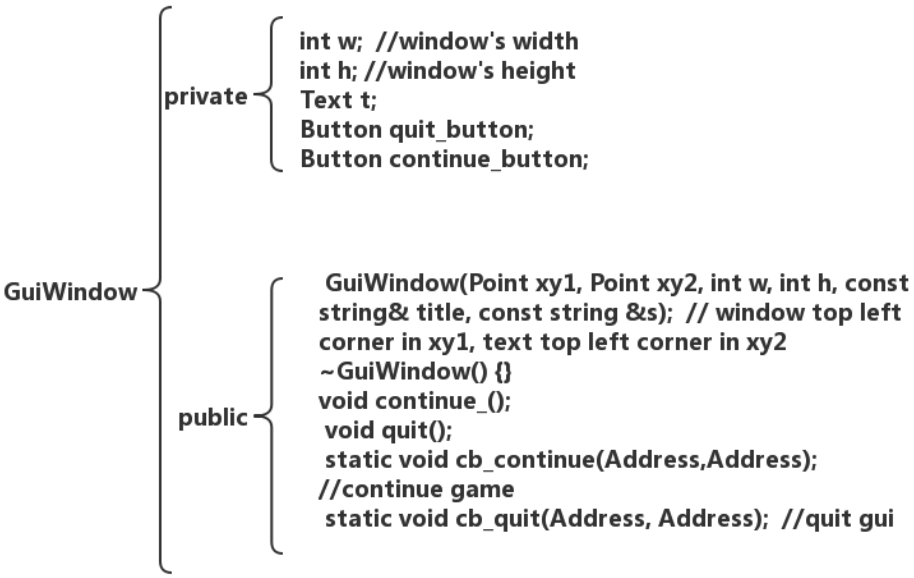
3.2.2.2 用于实现用户胜利或失败的 GUI 界面的类

①包含的头文件

```
#include "Graph.h"
#include "GUI.h"
#include "Simple_window.h"
#include "Point.h"
```

②类声明

```
//GUI when fail or win
class GuiWindow : public Simple_window
```



3.2.2.3 集成三个级别游戏实现和游戏呈现的命名空间 Gamegraph_lib

Gamegraph_lib 包含一个头文件 Gamegraph.h 和一个 cpp 文件 Gamegraph.cpp 包含四个类 Game_window、Game_medium、Game_hard、Applicant_window，前三个类分别为三个级别游戏的实现，最后一个类为进入三个级别游戏的 GUI 界面，将这四个类集成为一个命名空间，使结构更加清晰

Game_medium、Game_hard 均由 Game_window 派生

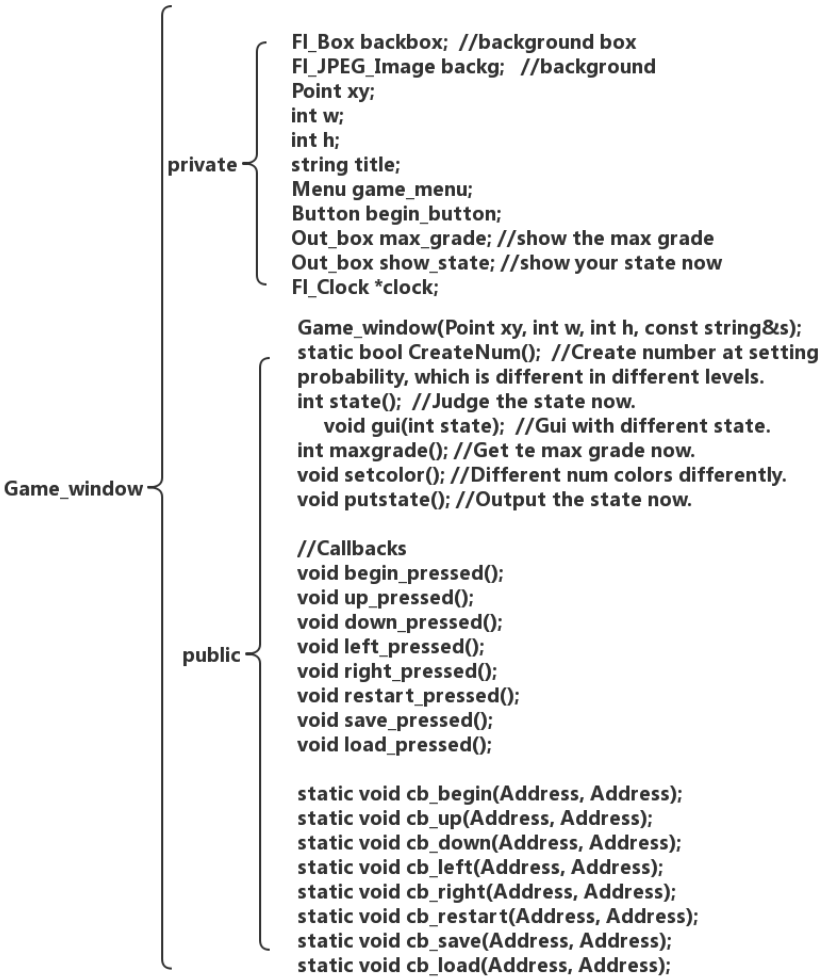
①Gamegraph_lib 包含的头文件

```
#include <string>
#include <memory>
#include <iostream>
#include <ctime>  //产生随机数需要使用
#include <fstream>  //存档与读档使用
#include "Graph.h"
#include "GUI.h"
#include "Simple_window.h"
#include "FL/forms.H"  //用于添加时钟
#include <FL/Fl_Clock.h>
#include <FL/Fl_JPEG_Image.H>  //用于添加背景图片
#include "Rect_with_text.h"
#include "global.h"
#include "GuiWindow.h"
```

②Game_window 类

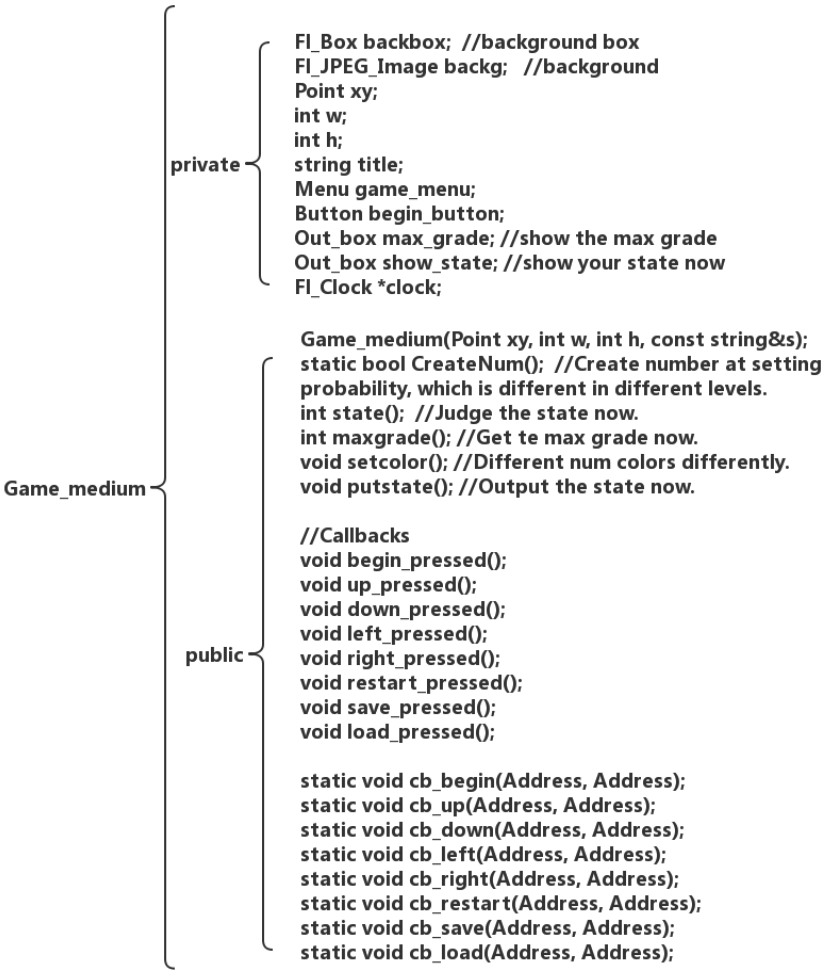
类声明

```
class Game_window : public Simple_window
```



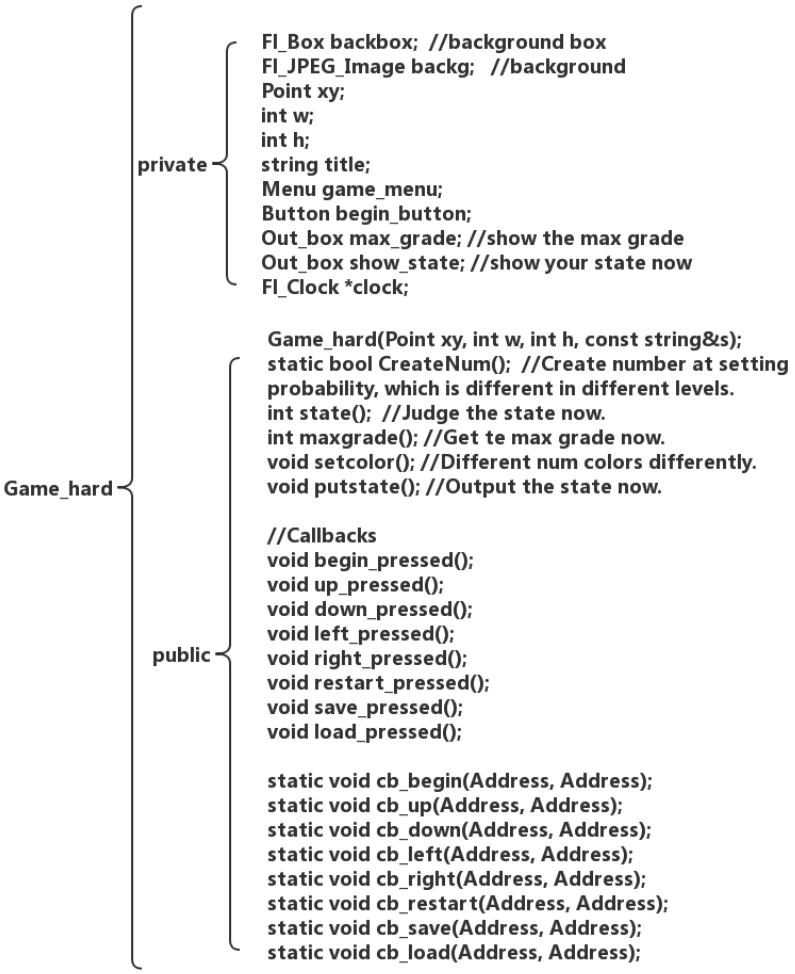
③Game_medium 类
类声明

```
1.  //Derive from Game_window, but several realization and rules are different.
2.      class Game_medium :public Game_window
```



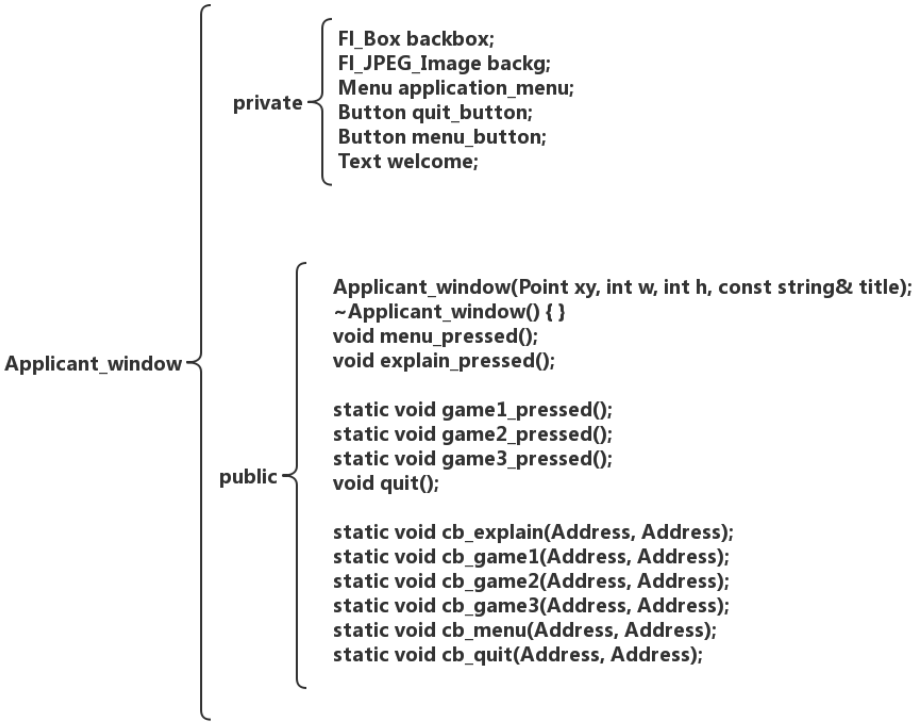
④Game_hard 类
类声明

```
1.  //Derive from Game_window, but several realization and rules are different.
2.      class Game_hard :public Game_window
```



⑤ Applicant_window 类

```
1.  //GUI linking to all game_windows
2.      class Applicant_window : public Window
```



3.3 函数功能描述

3.3.1Rect_with_text 类成员函数

1.构造函数

函数原型：Rect_with_text::Rect_with_text(Point xy, int ww, int hh, const int s);

功能描述：构造函数

参数描述： Point xy 为左上角点的坐标，int ww 为格子宽，int hh 为格子长，const int s 为格子中的数字

返回值描述：无

函数算法描述：无

```
1.  Rect_with_text::Rect_with_text(Point xy, int ww, int hh, const int s) : w(ww), h(hh), lab(t
    o_string(s)), fnt(fl_font()), fnt_sz(fl_size())
2.  {
3.      add(xy);
4.      if (h <= 0 || w <= 0) error("Bad rectangle: non-positive side");
5.  }
```

2.绘制函数

函数原型：void Rect_with_text::draw_lines() const

功能描述：绘制带有数字的格子

参数描述： 无

返回值描述：void（无）

函数算法描述：无

```
void Rect_with_text::draw_lines() const
{
    if (fill_color().visibility()) {    // fill
        fl_color(fill_color().as_int());
        fl_rectf(point(0).x, point(0).y, w, h);
    }

    if (color().visibility()) {    // lines on top of fill
        fl_color(color().as_int());
        fl_rect(point(0).x, point(0).y, w, h);
    }
    int ofnt = fl_font();
    int osz = fl_size();
    fl_font(fnt.as_int(), fnt_sz);
    fl_draw(lab.c_str(), point(0).x + w / 3, point(0).y + 2 * h / 3);
    fl_font(ofnt, osz);
}
```

3.获取格子的数字
函数原型： int Rect_with_text::num() const
功能描述：获取格子的数字
参数描述： 无
返回值描述：int
函数算法描述：使用 stoi 函数将字符串转化为数字

```
int Rect_with_text::num() const   //get the number of the rectangle
{
    if (lab == " ")
    {
        return 0;
    }
    else
    {
        return stoi(lab, 0, 10);
    }
}
```

3.3.2 GUIWindow 类成员函数

1.构造函数
函数原型：GuiWindow::GuiWindow(Point xy1, Point xy2, int w, int h, const string&

title, const string &s)
功能描述：构造函数
参数描述： Point xy1 为界面的左上角的坐标，xy2 为文本的左上角的坐标，w,h,title 分别为界面的宽、高、标题，s 为文本字符串
返回值描述：无
函数算法描述：无

```
GuiWindow::GuiWindow(Point xy1, Point xy2, int w, int h, const string& title, const string &s) :
    Simple_window(xy1, w, h, title),
    t(xy2, s),
    continue_button(Point(x_max() / 2 - 80, 80), 70, 20, "Continue &C", cb_continue),
    quit_button(Point(x_max() - 155, 80), 70, 20, "Quit &Q", cb_quit)
{
    t.set_font_size(16);
    attach(t);
    attach(quit_button);
    attach(continue_button);
}
```

2.回调函数
函数原型： void GuiWindow::cb_quit(Address, Address pw)（以此为例）、
void GuiWindow::quit()
功能描述：回调函数
参数描述：对象的地址
返回值描述：无
函数算法描述：对于事件处理，采用模板类的方法，将回调链接到对应的处理函数
模板类定义如下：

```
template<class W> W& reference_to(Address pw)
// treat an address as a reference to a W
{
    return *static_cast<W*>(pw);
}
```

```
void GuiWindow::cb_quit(Address, Address pw)    // "the usual"
{
    reference_to<GuiWindow>(pw).quit();
}

void GuiWindow::cb_continue(Address, Address pw)
{
    reference_to<GuiWindow>(pw).continue_();
```

```
}

void GuiWindow::quit()
{
    hide();        // curious FLTK idiom for delete window
}

void GuiWindow::continue_()
{
    hide();
}
```

3.3.3 Game_window 类成员函数

1.构造函数

函数原型： Game_window::Game_window(Point xy, int w, int h, const string&s)

功能描述：构造函数

参数描述：Point xy 为窗口左上角坐标，w,h,s 分别为窗口的宽、高和标题

返回值描述：无

函数算法描述：无

```
Game_window::Game_window(Point xy, int w, int h, const string&s) :
    Simple_window(xy, w, h, s),
    game_menu(Point(0, 0), 100, 20, Menu::vertical, "Menu"),
    begin_button(Point(0, 0), 100, 20, "Begin &B", cb_begin),
    max_grade(Point(200, 0), 80, 20, "Max grade:"),
    show_state(Point(380, 0), 120, 20, "State now:"),
    backbox(0, 20, 600, 400),
    backg("2048b.jpg")
{
    backbox.image(backg);
    add(backbox);
    for (int i = 0; i < 4; i++) {        // i is the horizontal coordinate
        for (int j = 0; j < 4; j++) {    // j is the vertical coordinate
            gamebase[i][j].reset(Point(x_max() / 2 - 100 + 50 * j, y_max() / 2 - 100 + 50 * i), 50, 50, 0);
            gamebase[i][j].set_font_size(14);
            gamebase[i][j].set_fill_color(FL_WHITE);
            attach(gamebase[i][j]);
        }
    }
    clock = fl_add_clock(FL_ANALOG_CLOCK, 0, 140, 100, 100, "Clock");
    //clock->draw();
    add(clock);
    game_menu.attach(new Button(Point(0, 0), 0, 0, " ↑ &W", cb_up));
```

```
    game_menu.attach(new Button(Point(0, 0), 0, 0, " ↓ &S", cb_down));
    game_menu.attach(new Button(Point(0, 0), 0, 0, "← &A", cb_left));
    game_menu.attach(new Button(Point(0, 0), 0, 0, "→ &D", cb_right));
    game_menu.attach(new Button(Point(0, 0), 0, 0, "Restart &R", cb_restart));
    game_menu.attach(new Button(Point(0, 0), 0, 0, "Save &E", cb_save));
    game_menu.attach(new Button(Point(0, 0), 0, 0, "Load &L", cb_load));
    attach(game_menu);
    attach(begin_button);
    game_menu.hide();
    attach(show_state);
    show_state.put("Init!");
    attach(max_grade);
    max_grade.put("no grade");
}
```

2. 着色
函数原型： void Game_window::setcolor()
功能描述：不同数字的格子对应不同的颜色
参数描述：无
返回值描述：无
函数算法描述：无

```
int ROW = 4;
int COL = 4;
Rect_with_text gamebase[4][4];
void Game_window::setcolor()  //different numbers has different colors
{
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            switch (gamebase[i][j].num())
            {
            case 0: gamebase[i][j].set_fill_color(FL_WHITE); break;
            case 2: gamebase[i][j].set_fill_color(FL_YELLOW); break;
            case 4: gamebase[i][j].set_fill_color(FL_DARK_YELLOW); break;
            case 8: gamebase[i][j].set_fill_color(FL_CYAN); break;
            case 16: gamebase[i][j].set_fill_color(FL_DARK_CYAN); break;
            case 32: gamebase[i][j].set_fill_color(FL_GREEN); break;
            case 64: gamebase[i][j].set_fill_color(FL_DARK_GREEN); break;
            case 128: gamebase[i][j].set_fill_color(FL_RED); break;
            case 256: gamebase[i][j].set_fill_color(FL_DARK_RED); break;
            case 512: gamebase[i][j].set_fill_color(FL_BLUE); break;
            case 1024: gamebase[i][j].set_fill_color(FL_DARK_CYAN); break;
```

```
            case 2048: gamebase[i][j].set_fill_color(FL_MAGENTA); break;
            }
        }
    }
}
```

3. 获取最大分数
函数原型： int Game_window::maxgrade()
功能描述：获取最大分数
参数描述：无
返回值描述：int
函数算法描述：依次比较获取最大分数

```
int Game_window::maxgrade()  //find the max grade
{
    int max = gamebase[0][0].num();
    for (int i = 0; i < ROW; i++)
    {
        for (int j = 0; j < COL; j++)
        {
            if (max < gamebase[i][j].num())
            {
                max = gamebase[i][j].num();
            }
        }
    }
    return max;
}
```

4. 随机产生数字
函数原型： bool Game_window::CreateNum()
功能描述：空白处随机产生数字 2、4
参数描述：无
返回值描述：bool（是否创建随机数）
函数算法描述：当有格子没有数字（指有格子数字为 0）时，在所有方框中，通过产生随机数，从所有空格子中选择一个填充数字，通过 whitch = rand() % 3 控制产生 2 和产生 4 的概率，这里选择 1/3 个概率产生 4,2/3 的概率产生 2

```
bool Game_window::CreateNum()
{
    int x = -1;
    int y = -1;
    int times = 0;
    int maxTimes = ROW * COL;
```

```
    // Two-thirds of the probability generates two, and one-third generates four.
    int whitch = rand() % 3;
    srand((unsigned int)time(0));
    do
    {
        x = rand() % ROW;
        y = rand() % COL;
        ++times;
    } while (gamebase[x][y].num() != 0 && times <= maxTimes);

    // The lattice is full.
    if (times >= maxTimes)
    {
        return false;
    }
    else
    {
        GameNum num;
        if (whitch == 0)
        {
            num = Game_4;
        }
        else if (whitch)
        {
            num = Game_2;
        }
        gamebase[x][y].set_num(num);
    }
    return true;
}
```

5.上下左右事件处理

函数原型（以向上为例）： `void Game_window::cb_up(Address, Address pw)`

`void Game_window::up_pressed()`

功能描述：用户操作向上事件处理

参数描述：cb_up 传入对象地址，up_pressed 无参数

返回值描述：无

函数算法描述：向上时，对每个格子，如果上面一个格子为空，那么上面的格子置为下面格子的数，以此类推，如果上面格子的数字当前格子的数字相同，则合并，数字*2，以此类推，其他动作类似

```
void Game_window::cb_up(Address, Address pw)
{
    reference_to<Game_window>(pw).up_pressed();
}
```

```
5.
6.  void Game_window::up_pressed()
7.  {
8.      for (int row = 1; row < ROW; ++row)
9.      {
10.         for (int crow = row; crow >= 1; --crow)
11.         {
12.             for (int col = 0; col < COL; ++col)
13.             {
14.                 // The last is empty
15.                 if (gamebase[crow - 1][col].num() == 0)
16.                 {
17.                     gamebase[crow - 1][col].set_num(gamebase[crow][col].num());
18.                     gamebase[crow][col].set_num(0);
19.                 }
20.                 else
21.                 {
22.                     //merge
23.                     if (gamebase[crow - 1][col].num() == gamebase[crow][col].num())
24.                     {
25.                         gamebase[crow - 1][col].set_num(gamebase[crow - 1][col].num() * 2);

26.                         gamebase[crow][col].set_num(0);
27.                     }
28.
29.                 }
30.             }
31.         }
32.     }
33.     CreateNum();  // create numbers randomly
34.     setcolor();
35.     this->redraw();
36.     putstate();  //show the state and max grade
37.     max_grade.put(to_string(maxgrade()));
38.     gui(state());
39. }
```

6. 判断当前状态
函数原型（以向上为例）： `int Game_window::state()`
功能描述：判断当前状态
参数描述：无
返回值描述：int 状态
函数算法描述：若有格子数字为 2048，则成功

若所有格子均有数字，并且横向纵向都无法合并，则失败
需要说明的是，由于产生数字为随机，故当剩下最后一个格子时，若没有产生数字，也算游戏失败，若产生了可以继续合并的数字，游戏可以继续

```
int Game_window::state()
{
    //赢得游戏

    for (int i = 0; i < ROW; ++i)
    {
        for (int j = 0; j < COL; ++j)
        {
            if (gamebase[i][j].num() == 2048)
            {
                return GAME_WIN;
                break;
            }
        }
    }
    //横向检查
    for (int i = 0; i < ROW; ++i)
    {
        for (int j = 0; j < COL - 1; ++j)
        {
            if (gamebase[i][j].num() == 0 || (gamebase[i][j].num() == gamebase[i][j + 1].num()))
            {
                return GAME_CONTINUE;
                break;
            }
        }
    }
    //纵向检查
    for (int j = 0; j < COL; ++j)
    {
        for (int i = 0; i < ROW - 1; ++i)
        {
            if (gamebase[i][j].num() == 0 || (gamebase[i][j].num() == gamebase[i + 1][j].num()))
            {
                return GAME_CONTINUE;
                break;
            }
        }
    }
```

```
    //不符合上述两种状况，游戏结束
    return GAME_OVER;
}
```

7. 输出当前状态
函数原型：void Game_window::putstate()
功能描述：输出当前状态
参数描述：无
返回值描述：无
函数算法描述：无

```
void Game_window::putstate()  //show state in the out box
{
    switch (state())
    {
    case 1: show_state.put("Game Over!");
    case 2: show_state.put("Game Win!");
    case 3: show_state.put("Game Continue!");
    default:
        break;
    }
}
```

8. 状态交互
函数原型： void Game_window::gui(int state)
功能描述：状态交互
参数描述：int 状态
返回值描述：无
函数算法描述：无

```
void Game_window::gui(int state)  //GUI if win or fail
{
    if (state == GAME_OVER)
    {
        show_state.put("Game Over!");
        GuiWindow win(Point(650, 350), Point(90, 50), 350, 150, "2048", "Fail to get 2048!
 You lost!");
        win.wait_for_button();
    }
    if (state == GAME_WIN)
    {
        show_state.put("Game Win!");
```

```
12.         GuiWindow win(Point(350, 250), Point(110, 50), 350, 150, "2048", "Got 2048!  You wi
    n!");
13.         win.wait_for_button();
14.     }
15.     if (state == GAME_CONTINUE)
16.     {
17.         show_state.put("Game Continue!");
18.     }
19. }
```

9. 重新开始
函数原型： void Game_window::restart_pressed()
功能描述：重新开始
参数描述：无
返回值描述：无
函数算法描述：重新开始时，游戏方框重画，随机选择两个空格子随机产生数字，着色，状态变为“Restart”，分数清空

```
1.  void Game_window::restart_pressed()
2.  {
3.      for (int i = 0; i < ROW; i++) {      // i is the horizontal coordinate
4.          for (int j = 0; j < COL; j++) {  // j is the vertical coordinate
5.              gamebase[i][j].reset(Point(x_max() / 2 - 100 + 50 * j, y_max() / 2 - 100 + 50 *
     i), 50, 50, 0);
6.              attach(gamebase[i][j]);
7.          }
8.      }
9.      CreateNum();  // Selecting randomly two lattices to generate numbers
10.     CreateNum();
11.     setcolor();
12.     show_state.put("Restart!");
13.     max_grade.put("no grade");
14.     this->redraw();
15. }
```

10. 存档与读档
函数原型： void Game_window::save_pressed()
void Game_window::load_pressed()
功能描述：存档与读档
参数描述：无
返回值描述：无
函数算法描述：通过存档和读档，实现当前 2048 游戏图元信息的存储和提取用户可以多次存档，但只记录最后一次存档的数据，存档在项目的相对路径

mygrade.txt 下，用户可以随时读档，读档时，可以在控制台看到读到的每个数字，读档将当前状态刷新为上次存档的图元信息，重新着色

```
1.  void Game_window::save_pressed()
2.  {
3.      ofstream saveFile;
4.      saveFile.open("mygrade.txt", ios::out|ios::trunc);
5.      int count = 0;
6.      for (int i = 0; i < ROW; i++)
7.      {
8.          for (int j = 0; j < COL; j++)
9.          {
10.             saveFile << " " << gamebase[i][j].num() << " ";
11.             count++;
12.             if (count % 4 == 0)
13.             {
14.                 saveFile << endl;
15.             }
16.         }
17.     }
18.     saveFile << endl;
19.     saveFile.close();
20. }
21.
22. void Game_window::load_pressed()
23. {
24.     ifstream loadFile;
25.     int num[16];
26.     loadFile.open("mygrade.txt");
27.     for (int i = 0; i < 16; i++)
28.     {
29.         loadFile >> num[i];
30.         cout << num[i] << " ";
31.     }
32.     int k = 0;
33.     for (int i = 0; i < ROW; i++)
34.     {
35.         for (int j = 0; j < COL; j++)
36.         {
37.             gamebase[i][j].set_num(num[k]);
38.             k++;
39.         }
40.     }
41.     setcolor();
42.     this->redraw();
```

```
43. }
```

3.3.4 Game_medium 类成员函数

由 Game_window 派生，构造函数类似，不同的是回调函数和成员函数的实现略有不同，此处不一一赘述，仅将不同之处加以阐明

1. 构造函数中方框的绘制不同
2. 随机产生数的函数中，产生 2 和产生 4 的概率进行了改变
3. 对于各个回调函数，由于执行的数组对象不同，都需要进行一定的变动
4. 存档和读档大小设置不同

3.3.5 Game_hard 类成员函数

由 Game_window 派生，构造函数类似，不同的是回调函数和成员函数的实现略有不同，此处不一一赘述，仅将不同之处加以阐明

1. 构造函数中方框的绘制不同
2. 随机产生数的函数中，产生 2 和产生 4 的概率进行了改变
3. 对于各个回调函数，由于执行的数组对象不同，都需要进行一定的变动
4. 存档和读档大小设置不同

3.3.6 Applicant_window 类成员函数

1.构造函数

函数原型： `Applicant_window::Applicant_window(Point xy, int w, int h, const string& title)`

功能描述：构造函数

参数描述：Point xy 为窗口左上角坐标，w,h,s 分别为窗口的宽、高和标题

返回值描述：无

函数算法描述：无

```
1.   Applicant_window::Applicant_window(Point xy, int w, int h, const string& title)
2.       :Window(xy, w, h, title),
3.       application_menu(Point(x_max() / 2 - 75, 130), 150, 50, Menu::vertical, "Menu"),
4.       menu_button(Point(x_max() / 2 - 75, y_max() / 2 - 60), 150, 50, "application menu &A",
    cb_menu),
5.       quit_button(Point(x_max() - 70, 0), 70, 20, "Quit &Q", cb_quit),
6.       welcome(Point(x_max() / 2 - 75, 90), "Welcome to 2048!"),
7.        backbox(0, 0, 600, 400),
8.        backg("2048i.jpg")
9.   {
10.      backbox.image(backg);
11.      add(backbox);
12.      welcome.font_size();
13.      welcome.set_font(FL_TIMES_ITALIC);
14.      attach(welcome);
15.      attach(quit_button);
16.      application_menu.attach(new Button(Point(0, 0), 0, 0, "GAME_base &B", cb_game1));
17.      application_menu.attach(new Button(Point(0, 0), 0, 0, "GAME_mediun &M", cb_game2));
18.      application_menu.attach(new Button(Point(0, 0), 0, 0, "GAME_hard &H", cb_game3));
19.      application_menu.attach(new Button(Point(0, 0), 0, 0, "About 2048 &E", cb_explain));
```

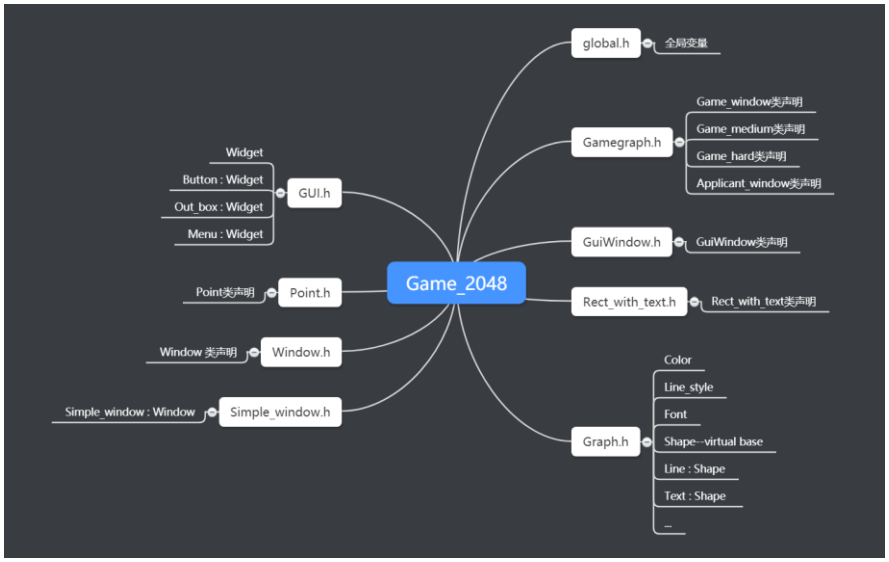
```
    attach(application_menu);
    application_menu.hide();
    attach(menu_button);
}
```

2.链接其他界面的回调函数（以链接 Game_base 为例，其他类似）
函数原型： void Applicant_window::cb_game1(Address, Address pw)
void Applicant_window::game1_pressed()
功能描述：回调函数
参数描述：cb_game1 传入对象地址，game1_pressed 无参数
返回值描述：无
函数算法描述：无

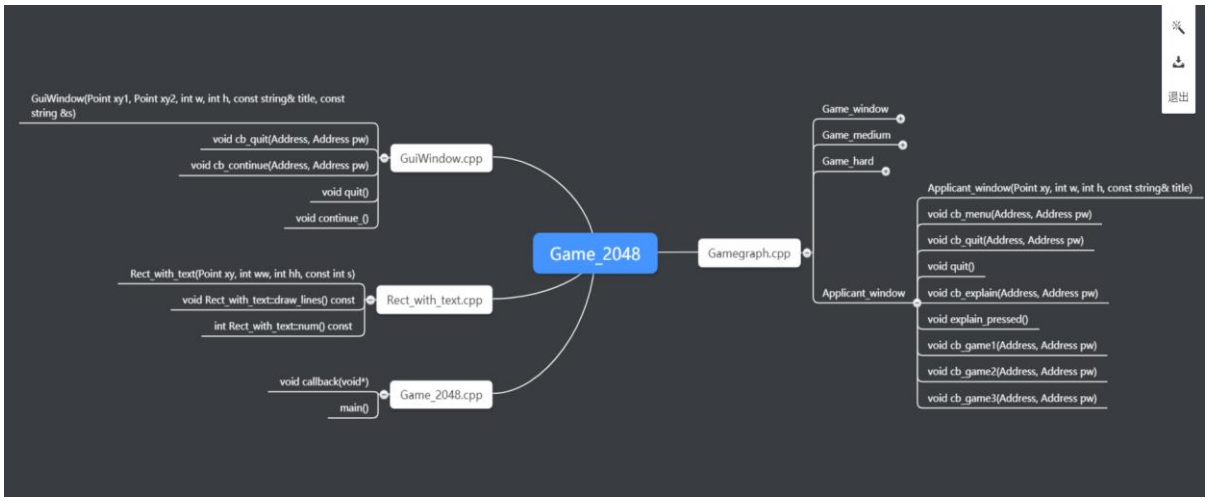
```
void Applicant_window::cb_game1(Address, Address pw)      // "the usual"
{
    reference_to<Applicant_window>(pw).game1_pressed();
}

void Applicant_window::game1_pressed()     // "the usual"
{
    int w = 50, h = 50;
    Game_window wingame(Point(500, 250), 600, 400, "2048");
    wingame.wait_for_button();
}
```

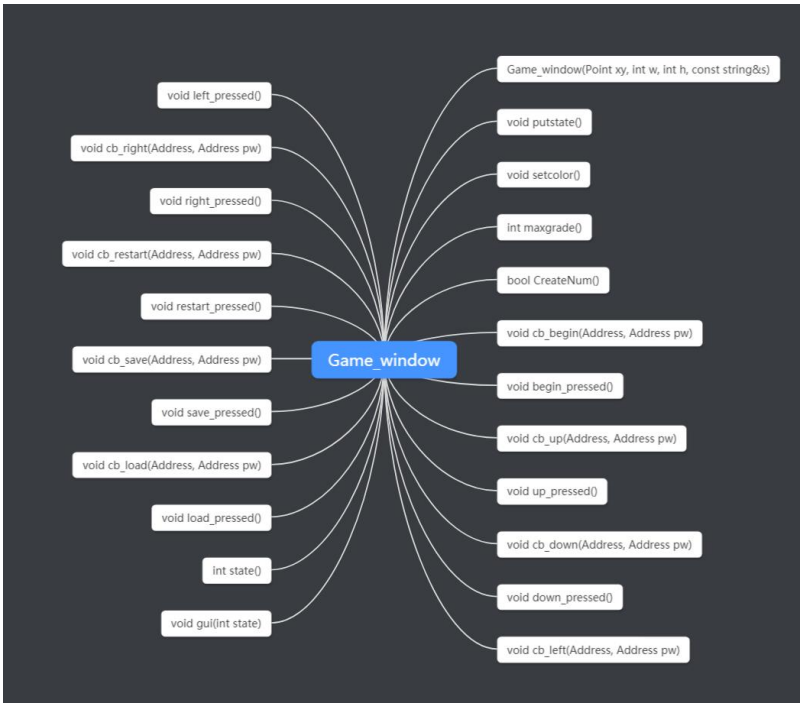
3.4 程序文件结构

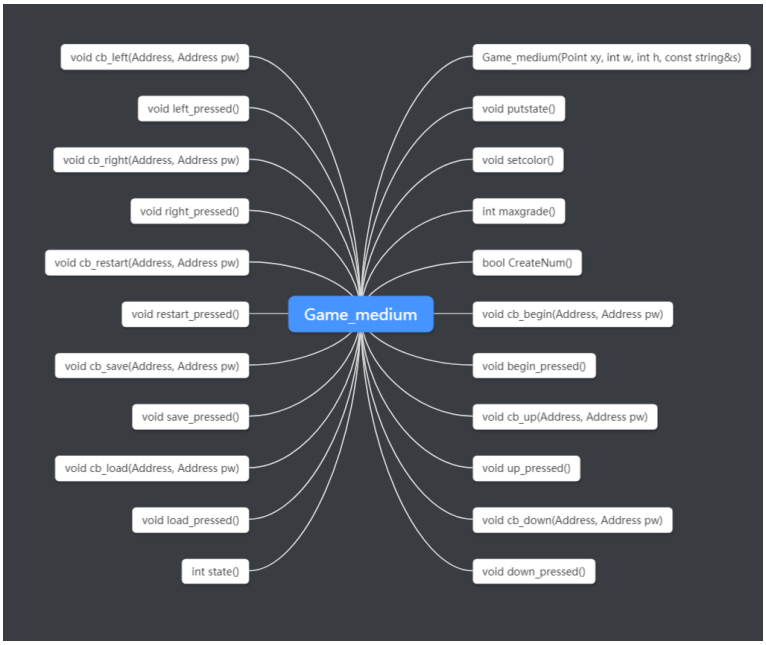
1）文件函数结构
（可通过图或文字描述程序分为几个文件（需写出文件全名，即*.h，*.cpp），每个.cpp 文件包含哪些函数定义，每个.h 包含哪些内容。）
头文件（.h）框架

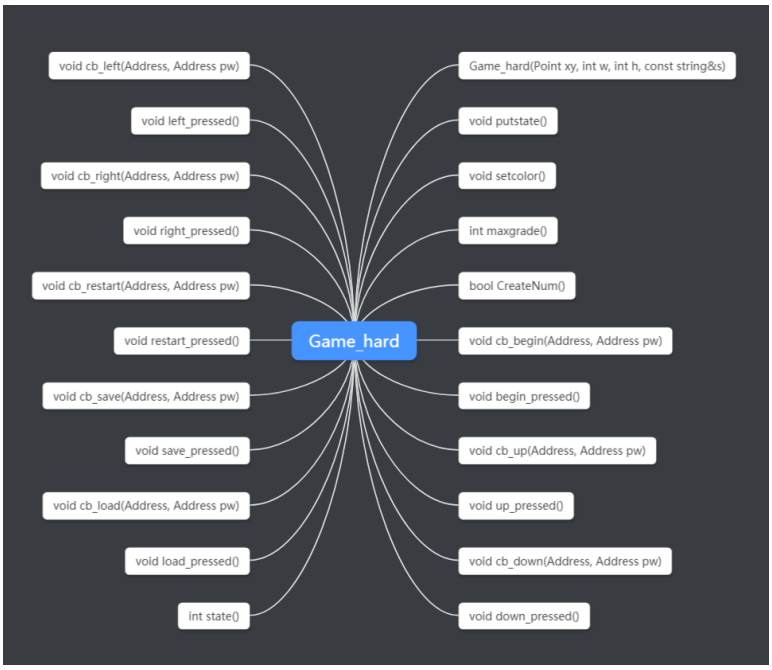

Game_2048
global.h
全局变量
Gamegraph.h
Game_window类声明
Game_medium类声明
Game_hard类声明
Applicant_window类声明
GuiWindow.h
GuiWindow类声明
Rect_with_text.h
Rect_with_text类声明
Graph.h
Color
Line_style
Font
Shape--virtual base
Line : Shape
Text : Shape
...
GUI.h
Widget
Button : Widget
Out_box : Widget
Menu : Widget
Point.h
Point类声明
Window.h
Window 类声明
Simple_window.h
Simple_window : Window

源文件（.cpp）框架


Game_2048
GuiWindow.cpp
GuiWindow(Point xy1, Point xy2, int w, int h, const string& title, const string &s)
void cb_quit(Address, Address pw)
void cb_continue(Address, Address pw)
void quit()
void continue_()
Rect_with_text.cpp
Rect_with_text(Point xy, int ww, int hh, const int s)
void Rect_with_text::draw_lines() const
int Rect_with_text::num() const
Game_2048.cpp
void callback(void*)
main()
Gamegraph.cpp
Game_window
Game_medium
Game_hard
Applicant_window
Applicant_window(Point xy, int w, int h, const string& title)
void cb_menu(Address, Address pw)
void cb_quit(Address, Address pw)
void quit()
void cb_explain(Address, Address pw)
void explain_pressed()
void cb_game1(Address, Address pw)
void cb_game2(Address, Address pw)
void cb_game3(Address, Address pw)
退出


Game_window
Game_window(Point xy, int w, int h, const string&s)
void left_pressed()
void cb_right(Address, Address pw)
void right_pressed()
void cb_restart(Address, Address pw)
void restart_pressed()
void cb_save(Address, Address pw)
void save_pressed()
void cb_load(Address, Address pw)
void load_pressed()
int state()
void gui(int state)
void putstate()
void setcolor()
int maxgrade()
bool CreateNum()
void cb_begin(Address, Address pw)
void begin_pressed()
void cb_up(Address, Address pw)
void up_pressed()
void cb_down(Address, Address pw)
void down_pressed()
void cb_left(Address, Address pw)





2）多文件构成机制

（说明分文件构成程序的实现机制）

多文件组织：

一个全局变量头文件、

一个主要集成三个级别游戏实现和游戏呈现的命名空间 Gamegraph_lib（包含.h 和.cpp）、

一个用于绘制游戏方框界面的类 Rect_with_text（包含.h 和.cpp）、

一个用于实现用户胜利或失败的 GUI 界面的类 Guiwindow(包含.h 和.cpp)、以及提供的 Graph.h、**Window.h（对该头文件和 cpp 进行了修改，典型测试中会做详细说明）**、Simple_window.h、GUI.h、Point.h、std_lib_facilities.h 以及对应的 cpp

4.运行结果

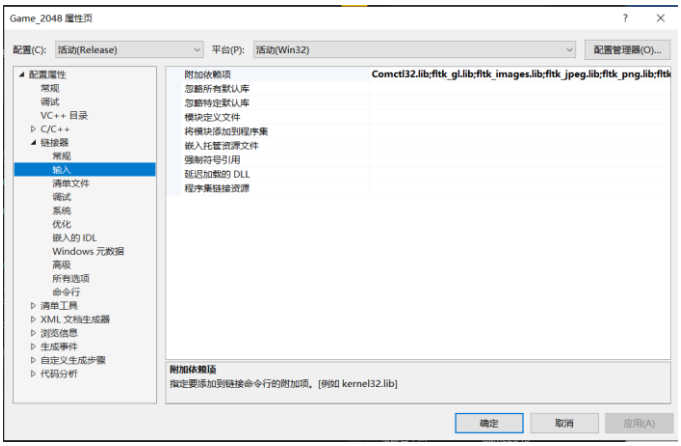
4.1 编译安装运行说明

（此部分介绍如何由提交的源代码包，进行存放、编译生成.exe 文件的过程说明，以及运行.exe 后的用户使用手册。）

4.1.1 编译安装

本项目采用 VS Release,x86 编译

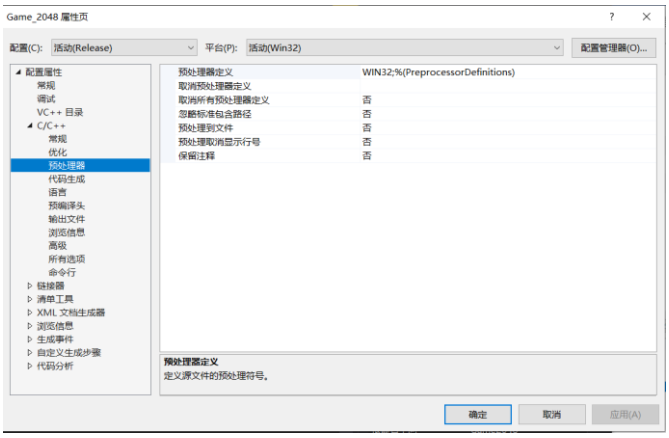
1.添加附加依赖项



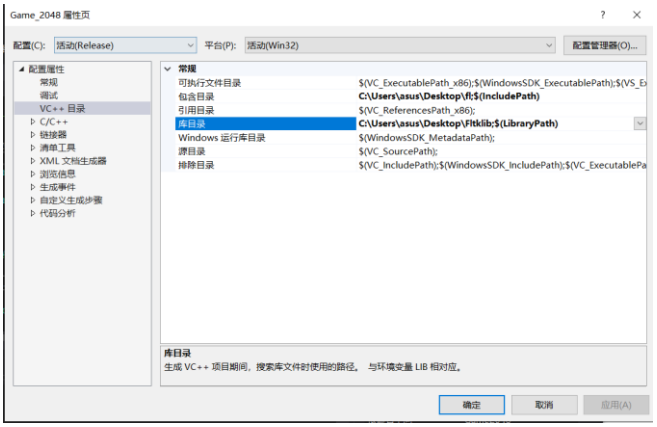
依赖项如下：

Comctl32.lib
fltk_gl.lib
fltk_images.lib
fltk_jpeg.lib
fltk_png.lib
fltk_zlib.lib
fltkforms.lib
fltkgl.lib
fltkz.lib
fltk.lib
wsock32.lib
comctl32.lib
fltkjpeg.lib
fltkimages.lib
kernel32.lib
user32.lib
gdi32.lib
winspool.lib
comdlg32.lib
advapi32.lib
shell32.lib

ole32.lib
oleaut32.lib
uuid.lib
odbc32.lib
odbccp32.lib
2. 预处理器添加 win32



3. 库目录包含 Fltklib 文件（可以不用）
包含目录无需添加（即把项目中的.h 和.cpp 都添加到项目中）



注意：不能将项目文件夹中的文件或者图片删去，否则会找不到文件或图片
4. 本项目已编译成功，项目文件夹\Game_2048\Release 下有已经编译好的 exe，可以直接打开

4.1.2 用户使用手册

Game_2048 用户使用说明

本游戏功能包括以下部分：
（1）三个级别的 2048 游戏，分别为 4*4，6*6，8*8 的 2048 游戏，对应 Game_base、Game_medium、Game_hard
（2）简单的用户使用说明界面
（3）支持用户随时重新开始
（4）支持用户随时存档读档，三个关卡不冲突
（5）用户达到 2048 会弹出胜利窗口，但是用户仍然可以继续游戏

（6）整个游戏所以按钮都支持快捷键，建议开启大写锁定体验，每个按钮的快捷键都在按钮的最后有说明，关闭每个游戏界面采用快捷键 N

打开 Game_2048.exe,点击菜单 Applicant_window

先查看 About 2048 浏览本游戏的简介，按快捷键 N 退出

根据自己的喜好进入不同等级的游戏，这里以 Base 为例进行说明：

点击 Begin 进入主菜单，游戏界面初始化，经测试，如果要让钟正常运行，需要将界面先最小化再打开(不同电脑运行情况不一),通过上下左右进行游戏，游戏上方显示当前的最大分数以及游戏所处的状态

游戏可以随时重新开始，分数会清零，状态更新为 Restart

游戏支持随时存档读档，三个等级互不冲突，注意：只有存过档才能读档，否则没有档案就读档会出现错误！本项目为避免此问题，会将档案预设，另外，存档读档也可以通过小钟下方的图标进行存档,用户可以根据自己的喜好进行操作

游戏成功或者失败后都会弹出提示窗口，成功后可以继续游戏，失败后可以按提示退出失败界面冲新开始或者退出

退出任何一个游戏界面的快捷键为 N，也可以通过图标按钮退出

退出主界面的快捷键为 Q

4.2 典型测试情况

以下是我在改进自己程序中发现的问题与解决方案

（1） 闪屏问题

2048 在改变界面时采用的是重画界面的方法，在界面切换比较快的时候会产生闪屏问题，解决闪屏问题一般用的是双缓冲，查阅官方文档发现，FLTK 有可以实现双缓冲的窗口 Fl_Double_Window，于是修改提供的库 Window.h，将其继承自

Fl_Double_Window，.cpp 中做出相应的改变，解决闪屏问题

（2） 界面问题

程序原本设定产生 2048 后即弹出胜利窗口，但是可以继续游戏，测试发现继续游戏后会不断弹出胜利窗口，于是设置全局变量 flag 判断是否成功过，从未成功完成 2048 则 flag=0,一旦在一次游戏中产生 2048,flag=1,继续游戏则不会再弹出 2048 的胜利窗口，在游戏重新开始（Restart）或者下次游戏开始（Begin）的时候将 flag 重置为 0,于是解决该问题

（3） 存档读档问题

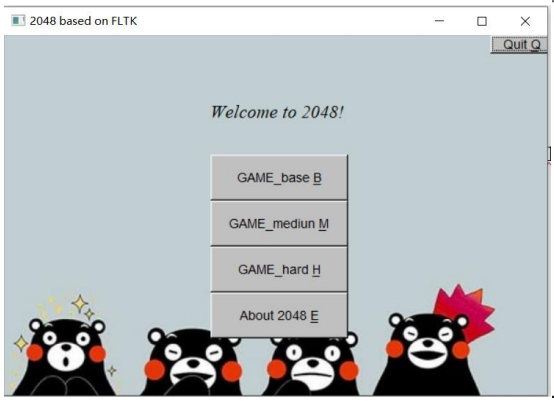
程序原本存档读档在绝对路径实现，后来经同学测试后发现问题，排查发现是没有对应的路径，于是改为相对路径存取，解决了该问题

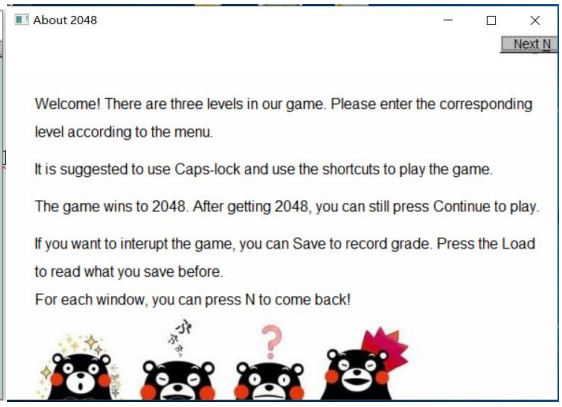
（4） 背景问题

刚开始给界面添加背景时，会出现覆盖按钮的情况，后来把 Fl_Box 和 Fl_JPEG_Image 最先初始化，就解决了该问题

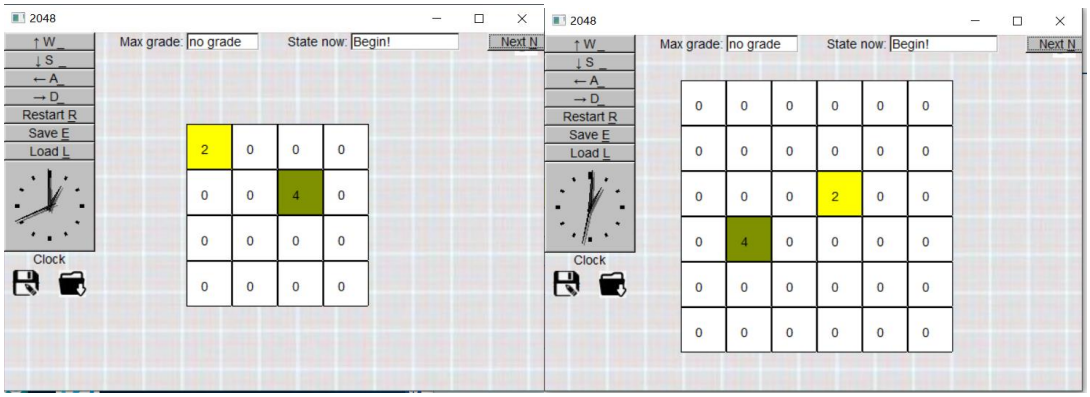
效果展示：

主界面





游戏界面（包含图标工具栏、状态工具栏、实时时钟显示、主要游戏界面、游戏功能菜单栏）存档读档既可以通过图标工具栏实现，也可以通过快键键 E、L 实现



5. 总结

5.1 程序亮点或创新之处

（1）程序将游戏实现主体（三个实现功能的类和一个链接界面的类）集合为一个命名空间，结构更加清晰

（2）程序在经典的 2048 游戏上增加了两个级别，更有益智效果

（3）程序实现的界面有简单的 GUI 说明，类似于游戏中的 HELP，功能更加完善

（4）程序支持用户随时重新开始和随时存档读档，十分方便

（5）用户能够随时看到自己的游戏状态，包括当前时间，游戏界面设计合理

（6）界面美观

5.2 应用知识点

（1）类：类的设计、类的继承与派生、静态成员函数、虚函数、输入输出流

（2）命名空间的设计

（3）模板函数的设计与使用

（4）文件的存取/存取图元信息

（5）多文件组织

（6）全局变量的组织

（7）FLTK 图形库综合应用

5.3 心得体会

本次大作业考察读库的能力，学习 FLTK 库的过程中，通过翻看官方文档以及不断链接到相关函数的定义，一步步了解 FLTK 库的使用，并发现了众多封装好的功能实现，比如说 FLTK 中有一个可以实现双缓冲的 Fl_Double_Window，非常使用，能够解决闪屏问题，并且自带的模拟时钟也比较好用

其次就是 2048 游戏算法的编写，算法总体不难实现，但是会在实现的过程中发现很多细节需要优化，比如说成功只弹出一次窗口，存档读档不能相互干扰，制作游戏还要考虑游戏过程中可能出现的情况，比如说想要重新开始等等，并且应该设计一个 GUI 界面进行一定的用户使用说明

总的来说通过大作业加深了 C++知识的应用，对类的设计、多文件组织等知识点有了更深的理解

6、参考文献和资料

（列出参考的书籍、论文、网站的信息和地址等。）

参考 FLTK 官方文档：https://www.fltk.org/doc-1.3/examples.html

FLTK 中文手册：https://wenku.baidu.com/view/4f642b260722192e4536f670.html