ACM Class B “雷电”

大作业报告

作者：张文涛

学号：517030910425

目录

一 设计想法与思路

1.简介

1)开发环境

Manjaro Linux & notepadqq & g++(gcc 7.20)

2)思路

对于这个作业，在熟悉所提供的接口后的一个很自然的想法是面向对象的设计。因为游戏界面中的没一个移动物体都可以看作一个对象（如玩家飞机，敌机，道具，子弹），有相同点（如具有速度，位置，大小等特征），也有不同点（如玩家飞机需要在运行时接受信号等等），也需要受到管理。那我们就可以创建不同的对象并且通过一个类似于信号中转的东西在相互之间通信，以达到控制整个游戏的目的。另外，为了让设计看上去更加现代化，我使用的简洁的设计风格，从阿里的 icon 图库里下载了资源，作为贴图，达到了比较好的效果。

2.类定义详细说明

顺着刚才简介的思路，我们对整个游戏进行向下分解，于是就有了

```
class PlayerCraft;
class Bullet;
class Enemy;
class SignalRouter;
class Prop;
class UserInteract;
```

这样六个部分

PlayerCraft 用于处理各类与玩家所操控的飞机有关的信息。以及一些成员函数

Bullet 用于处理保存各类子弹有关的信息。以及一些成员函数

Enemy 用于处理保存敌方飞机的有关信息。以及一些成员函数

Prop 用于处理保存各类道具有关的信息。以及一些成员函数 ~~其实可以存在 enemy 里~~

SignalRouter 管理以上类，在其间传递信息

UserInteract 负责与用户交互，显示分数等

通过以上这些类，我们就可以让游戏运转起来。

继续向下分解， 我们发现在游戏中出现在屏幕上的对象都有共同点和不同点， 于是采取了基类和派生的方式。

构造基类 PaperObj

```
class PaperObj{
public:
        PaperObj();
        virtual ~PaperObj();
        inline void velocityChange(const PointD &v);
        inline void velocitySet(const PointD &v);
        inline void posChange(const PointD &p);
        inline void speedSet(const double s);
        virtual void move();
        BumpBox *BPB = nullptr;
        double speed;
protected:
        PointD velocity = PointD();
        PointD pos = PointD();};
```

每一个在屏幕上移动的物体都有坐标和速度， 可以对他们进行的操作是速度变化，速度设定，位置变化，和执行移动。另外由于部分对象的 move 函数有特殊需要，因此将其定义为虚函数。

然后从中派生出处理对应单个对象的类

PlaerCraft 处理玩家的飞机

EnemyCraft 处理敌方的飞机

PaperBullet 处理子弹

PaperProp 处理道具

由于我们屏幕上有众多的不同种类对象， 于是就自然地想到在对应的管理类中创建对象的数组， 加以管理。

并且为了更好地利用内存， 在本程序内所有的内存都是动态管理的。

上面的每一个类，都有类似的不相同的功能。

除了继承自基类的函数以外，都有相应的成员函数

1)PlayerCraft

保存了生命值，道具数量，受保护时间，图片资源的指针，图片的尺寸，子弹发射间隔等信息。并且由于需要接受用户键盘输入，重写了 move()函数以接受用户信息。

2)EnemyCraft

记录敌方飞机的状态， 生命值，类型，画飞机的函数等等。

3)PaperBullet

记录子弹的状态，类型

4)PaperProp

记录道具类型状态

每一个对应的管理类都包含一个所管理对象的数组（动态分配的）

每个对应的管理类都有例如 moveAll()，drawAll()，checkAndDeal 之类的全局处理函数来对所管理的对象进行操作并监察所管理对象的合法性。及时回收不合法的对象并重置，循环利用动态分配的内存。

每个管理类也有特殊的部分

Bullet 类有分配新子弹的函数，以便于其他的对象通过 SignalRouter 调用发射子弹。

并且可以通过不同的调用参数制定不同的发射策略。（直线，半圆，反半圆，追踪等等）

Enmey 类有用于产生新敌方飞机的函数，可以在一定时间内增加一定量的飞机，提高挑战性，并且最大的飞机数量是可以设定的。

Prop 类也有独有的 randomAllocate 函数， 可以随机地产生一些道具。也有从 EnemyCraft 类型产生道具的分配函数，可以做出道具掉落效果。

这样一来各部分的分工都比较明确了。最后分析权限最高的 SignalRouter 以及 UserInteract 类。

3.控制与交互方式说明

1)SignalRouter

控制以及全盘操作使用的类，在 main 函数中定义为指针 ctrl。

```
class SignalRouter{
    public:
        friend class PlayerCraft;
        friend class Enemy;
        friend class EnemyCraft;
        friend class Bullet;
        friend class Prop;
        friend class UserInteract;
        SignalRouter();
        virtual ~SignalRouter();
        void init();
        void drawAll();
        void dealAll();
        void moveAll();
        void dealBump();
        void reset();
        void bombReset();
//PointD windVelocity = PointD();//风力设计，来不及写了
        std::map<int, bool> keyboard;
        double mX = 0;
```

由于负责类之间的信息传递，几乎所有的对象类与对象管理类都是 SignalRouter 的友元类。

init() 初始化所管理的对象。

后慢三个函数执行所管理对象的对应函数

放出大炸弹的操作

```
        double mY = 0;
        bool MC = false;
        int state = GameState::INIT;
private:
        int max = 10;
        PlayerCraft *PC = nullptr;
        Enemy *EM = nullptr;
        Bullet *BLT = nullptr;
        Prop *PRP = nullptr;
        UserInteract *UI = nullptr;};
```

记录键盘映射，鼠标状态，是否被按下以及游戏状态

max 表示同屏幕可显示的最大敌方飞机数量

以及所管理对象的指针（这些对象将被在游戏启动时动态分配）

2)UserInteract

与用户交互的类，主要负责给用户提示，以及开始界面等等

```
class UserInteract{
public:
        UserInteract(SignalRouter *sig):SR(sig){}
        virtual ~UserInteract();
        void init();
        void drawHint();
        void drawWelcome(); //TODO
        void drawEnd(int res);
        void checkAndDeal();
private:
        略
```

1. 初始化时加载所有的图片资源
2. 画分数，道具数量等等给用户的提示信息
3. 开始界面
4. 结束界面（显示结果和分数）
5. 接受用户输入并给出对应响应

通过各个类之间的信息传递和配合，游戏就可以运行起来，剩下的只是一些细节问题。

4.碰撞检测

4.1 碰撞箱实现

作者本意是希望做出像雷电那样与整个机体都能够碰撞的效果，并且也确实实现了。可以看一抗碰撞箱的类定义。

```
class BumpBox{
	public:
	BumpBox(){
		BumpUnit = new CirRange[10];
	}
	void init(const int type, const PointD &p, const int &PicW = 10, const int &PicH = 10);
	void posChange(const PointD &v)
	void reset();
	void move(const PointD &v, const double sp);
	//void posSet(const PointD &p);
	void BumpDetect(BumpBox *target);
	inline int ifBumped(){
		return state;
	}
```

以上的定义是不完整的，我们删除了一些不需要的内容以更清楚地解释碰撞的问题。

首先碰撞箱具有动态分配的大小为 10 的碰撞单元，这种类型为 CirRange 的结构体中包含了一个 PointD 类型，以及一个 double 类型来存储碰撞单元中心位置以及半径，每个在通过这 10 个碰撞单元我们可以改变他们的相对关系来拟合屏幕上飞机的外形。屏幕上移　动的并且需要参与碰撞的对象都会被绑定一个碰撞箱，在对象被创建时就被动态分配，随后可以根据对象类型进行初始化，初始化的方式可以通过调用 init()，并在参数中指定　初始化的类型。在对象移动或位置被改变时，我在移动和位置改变里都调用了绑定在对象上的碰撞箱的对应函数以保证碰撞箱和对象是同步的。BumpDetect 函数用于检测碰撞，其参数是另一个对象的碰撞箱。然后还内置了一个检查状态的函数 ifBumped()来取得碰撞检测的结果。

4.2 碰撞检测实现

由上我们定义了碰撞箱，并且定义了 BumpDetect 函数，那么它的具体实现函数也是非常　　直观，容易理解的。我们利用一个双重循环，对每一个己方碰撞单元，与对方的每一个　　碰撞单元利用圆心距和半径之间的关系来判断他们是否碰撞。一旦碰撞就立刻退出循环。　如此就实现了碰撞检测。

二 功能与特点介绍

1.特点

1)游戏采用了简洁的界面和风格，和主流趋势相契合。

2)没有调用过于垃圾的 textToImage 函数，所有的提示文字都是由个人制作的图片而来，使用与播放动画类似的原理显示分数，以保证风格统一。

3)难度可自定义，可以在开始时输入一个数决定同屏幕出现敌机的最大数量。

2.功能介绍

1)拥有所有基本功能

2)上下左右方向键控制飞机移动，空格键发射子弹， b 键使用炸弹。有开始界面以及结束时的分数统计。

3)在游戏开始时的控制台中可以输入同屏幕最大容纳的飞机数量以设定难度

4)初始生命值为 99, 在分数达到一定程度时就可以触发 boss 战，战胜 boss 即为获胜，生命值减为 0，即为失败。

5)有丰富弹幕效果，有多种子弹类型（追踪，半圆，直线等）。

6)道具掉落，且有随机生成机制，玩家可以通过道具获取复活次数和炸弹。~~增加能量（来不及写了再见）~~

7)结束时可以通过剩余道具增加分数。

三 遇到的问题

1.textToImage 函数以及缩放图片产生锯齿：用不缩放的原始图片替代。

2.在 ddl 一周前几乎完工时误删整个大作业目录，并且无法找回：通宵，学习使用 git & github

3.内存不够用：完全动态分配内存。

4.加载图片无效，在助教知道下改变加载图片的位置。

5.由于对整架敌方飞机进行碰撞检测，使得游戏体验非常差，几乎无法获胜，于是将雷电式的碰撞检测更改为东方式的碰撞检测，即只对中心的一个圆点进行碰撞判定，得到较好的游戏体验。具体操作则是将碰撞箱所包含的碰撞单元数量改为 1，然后认为指定碰撞半径即可。

6.内存不足，最初在载入了过多的图片时，栈区内存不足，于是所有对象都是用了动态内存分配。

虽然我之前曾开发过规模类似的项目，但是通过这次作业，我的代码能力得到了提高。可能也是实践了一下新学会的东西，对开发有了更深的理解。

四 感想

珍爱生命

谨慎操作

五 建议

这个大作业是非常有意思的~~（重写一遍完全没有意思/再见）~~

但是竞技性不足，没有 A 班那种 AI 的竞技性。并且与游戏设计相关，设计性很强，是非常开放的，可以很简单也可以很难。

因此，我认为评分标准可以更加开放一点，给有能力的同学发挥的空间。也可以提高零基础同学的开发能力。