64 组：三个火枪手 Qt 大作业报告

组员　　张修瑞 王政浩 肖斌

一．概述

游戏名称　　手绘：穹顶之上（极度困难的飞机大战）
环境版本　　QT CREATER 4.11.1（如图 1.1）
分类　　飞行射击

```
QT       += core gui multimedia

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++11 resources_big
```

（图 1.1）

二．函数逻辑

1. 目录

```
SOURCES += \
    bullet.cpp \
    dialog.cpp \
    end.cpp \
    enemy.cpp \
    enemybullet.cp ......
    gamecontrol.cpp \
    gamedefine.cpp \
    gameobject.cpp \
    gameobjectpool.cpp \
    guide.cpp \
    main.cpp \
    pause.cpp \
    plane.cpp \
    player.cpp \
    playerbullet.cpp \
    widget.cpp

HEADERS += \
    bullet.h \
    dialog.h \
    end.h \
    enemy.h \
    enemybullet.h \
    gamecontrol.h \
    gamedefine.h \
    gameobject.h \
    gameobjectpool.h \
    guide.h \
    pause.h \
    plane.h \
    player.h \
    playerbullet.h \
    widget.h

FORMS += \
    dialog.ui \
    end.ui \
    guide.ui \
    pause.ui \
    widget.ui

TRANSLATIONS += \
    PlaneWar_zh_CN.ts

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

RESOURCES += \
    src.qrc
```

Gamecontrol 主效用类，由 dialog 进入

Gamedefine 主定义类，包括列头文件，游戏基础参数

Gameobject 游戏物品类，分支飞机/子弹

Gameobjectpool 缓存池，减少内存占用，避免后期卡顿

Plane 飞机类，分支我机/敌机

Enemy 定义敌机（分支敌机 1、2、3）及其参数

Player 定义我机及其参数

Bullet 子弹类，分支我机/敌机子弹

Playerbullet 定义我机子弹及其参数

Enemybullet 定义敌机子弹及其参数

Main 主函数

Widget 菜单

Guide 帮助

Dialog 游戏界面

Pause 暂停 ui

End 结算 ui

2.摘选详述

（1）GameObject

```
class GameObject : public QGraphicsPixmapItem
{
public:
    enum ObjectType
    {
        OT_BulletPlayer = 0,
        OT_Enemy1 = 1
    };

    explicit GameObject(QObject *parent = nullptr);

    int GetType()
    {
        return mObjectType;
    }

    void RecycleObject(QGraphicsScene* _scene);


    ~GameObject(){}

    int mObjectType;

};
```

物品类继承像素图原件类。

在承担游戏直接原件的基类作用同时，为搭建缓存池作保障。

使用枚举确定预备缓存池类型，

其中定义缓存池回收函数以保证所有子类都能直接回收。

```
GameObject::GameObject(QObject *parent)
{
}

void GameObject::RecycleObject(QGraphicsScene* _scene)
{
    _scene->removeItem(this);
    GameObjectPool::Instance()->RecycleGameObject(this);
}
```

（2）Dialog

```
#include <QDialog>
#include <QGraphicsPixmapItem>
#include <QGraphicsView>
#include <QGraphicsScene>
#include <QList>
#include <QLabel>
#include "player.h"
#include "bullet.h"
#include "enemy.h"
#include <QSoundEffect>

namespace Ui {
class Dialog;
}

class Dialog : public QDialog
{
    Q_OBJECT

public:
    explicit Dialog(QWidget *parent = nullptr);
    ~Dialog();

    static Dialog* wind;
    static int Count;
    static int Goal;
    static int Condition;

    void keyPressEvent(QKeyEvent *event);
    void keyReleaseEvent(QKeyEvent *event);

private:
    Ui::Dialog *ui;

private slots:
};
```

在 dialog 定义静态全局变量和公用父窗口，便于控制台函数使用。

```
Dialog* Dialog::wind = nullptr;
int Dialog::Count = 0;
int Dialog::Goal = 0;
int Dialog::Condition = 0;
Dialog::Dialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Dialog)
{
    ui->setupUi(this);
    this->setWindowTitle("UPON SKY NOW!");
    this->setFixedSize(750,1000);
    wind = this;

    //加载
    GameControl::Instance()->GameInit();
}
//按键落下写入方向容器
void Dialog::keyPressEvent(QKeyEvent *event)
{
    switch (event->key())
    {
        case Qt::Key_W:
        case Qt::Key_A:
        case Qt::Key_S:
        case Qt::Key_D:
        case Qt::Key_Escape:

        GameControl::Instance()->mKeyList.append(event->key());
        break;
    }
}

//按键抬起移除容器中的方向
void Dialog::keyReleaseEvent(QKeyEvent *event)
{
    if(GameControl::Instance()->mKeyList.contains(event->key()))
    {
        GameControl::Instance()->mKeyList.removeOne(event->key());
    }
```

键盘录入存储进入链表，调用时读取链表即可，由此可以开放组合键（例如“WD”右上），减少损耗，且整洁。

（3）GameObjectPool 缓存池

```
void GameObjectPool::Init()
{
    //预生成
    for(int i = 0; i < 60; i ++)
    {
        PlayerBullet* Bullet1 = new PlayerBullet();
        Bullet1->mObjectType = 0;
        mPlayerBulletPool.append(Bullet1);

        enemy* Enemy1 = new enemy();
        Enemy1->mObjectType = 1;
        mEnemy1Pool.append(Enemy1);
    }
}
```

创造系外链表，游戏开始时 new 得到适量对应 Object，需要使用时根据类型取出，需要回收时放回链表，游戏结束时清空。

避免频繁 new、delete。

```
GameObject *GameObjectPool::GetGameObject(int _objType)
{
    switch (_objType)
    {
    case GameObject::OT_BulletPlayer:
    {PlayerBullet* Bullet = mPlayerBulletPool.first();
    mPlayerBulletPool.pop_front();
    return Bullet;}

    case GameObject::OT_Enemy1:
    {enemy* Enemy = mEnemy1Pool.first();
    mEnemy1Pool.pop_front();
    return Enemy;}
    }
}

void GameObjectPool::RecycleGameObject(GameObject *_obj)
{
    switch (_obj->GetType())
    {
    case GameObject::OT_BulletPlayer:
    {
       mPlayerBulletPool.append((PlayerBullet*)_obj);
       break;
    }
    case GameObject::OT_Enemy1:
    {
        mEnemy1Pool.append((enemy*)_obj);
        break;
    }
    }
}

void GameObjectPool::Clear()
{
    for(auto pBullet : mPlayerBulletPool)
    {
        delete pBullet;
    }
    for(auto pEnemy1 : mEnemy1Pool)
    {
        delete pEnemy1;
    }
}
```

```
class GameObjectPool : public QObject
{
    GameObjectPool(QObject *parent = nullptr);
    static GameObjectPool* instance;
public:
    static GameObjectPool* Instance()
    {
        if(instance == nullptr)
        {
            return instance = new GameObjectPool(Dialog::wind );
        }
        return instance;
    }
    void Init();
    GameObject* GetGameObject(int _objType);

    void RecycleGameObject(GameObject* _obj);

    void Clear();

    ~GameObjectPool();


protected:
    //容器
    QList<PlayerBullet*> mPlayerBulletPool;
    QList<enemy*> mEnemy1Pool;

};
```

（4）bullet 与 plane 及其子类

从 object 到具体子类的过渡类。

定义子类可以共用的功能函数或特性

减少代码冗余的弊病。

```
bullet::bullet(QPoint _pos, QPixmap _pixmap, int _type)
{
    //子弹的精参数设置
    this->setPos(_pos);
    this->setPixmap(_pixmap);

    this->mBulletType = _type;
    mSpeed = 5;

}

void bullet::BulletMove(QPoint _dir)
{
    //子弹移动定义
    this->moveBy(_dir.x()*mSpeed,_dir.y()*mSpeed);
}

void bullet::Init(QPoint _pos, QPixmap _pixmap)
{
    this->setPos(_pos);
    this->setPixmap(_pixmap);

    this->setX(this->x() - this->pixmap().width()/3);
}
```

```
enemy::enemy(QPoint _pos, QPixmap _pixmap)
{
    this->setPos(_pos);
    this->setPixmap(_pixmap);
}

void enemy::EnemyMove(QPoint _dir)
{
    this->moveBy(_dir.x() * MoveSpeed , _dir.y() * MoveSpeed);
}

void enemy::Init(QPoint _pos, QPixmap _pixmap)
{
    this->setPos(_pos);
    this->setPixmap(_pixmap);
}
```

（5）GameControl 控制台

```
//背景移动
void GameControl::BGMove() {...}
//初始化对象
void GameControl::GameInit() {...}
//加载准备界面
void GameControl::LoadStartScene() {...}
//加载游戏界面
void GameControl::LoadGameScene() {...}

//游戏开始
void GameControl::GameStart() {...}
//游戏暂停
void GameControl::mPause() {...}
//游戏结束
void GameControl::GameOver() {...}

//创建敌机
void GameControl::EnemyCreate() {...}

//我的移动原则
void GameControl::PlaneMove() {...}
//我的死亡动画
void GameControl::MeDown() {...}
//我的子弹发射
void GameControl::PlaneBulletShoot() {...}
//执行我的飞行动画
void GameControl::PlaneFlash() {...}

//Hunter子弹发射
void GameControl::Enemy2BulletShoot() {...}
//BOOMER爆炸动画
void GameControl::Enemy3Flash() {...}
//BOOMER到达末端
void GameControl::BOOM() {...}

//碰撞检测
void GameControl::Collision() {...}

//清理
void GameControl::ClearScreen() {...}
```

BGMove： 保持背景滚动。

GameInit： 游戏初始化，主要为定时器及其信号槽初始化。

LoadStartScene：“芜湖起飞”界面，缓存池生成。

LoadGameScene：初始化游戏初始原件，放置景象和原件。

GameStart：定时器开始。

mPause： 定时器暂停，根据新暂停页面的选择调整对策。

GameOver： 定时器停止，结算。

EnemyCreate： 敌机创建原则声明，开始使用缓存池。

PlaneMove： 我机移动函数，配合上文 Dialog 中的按键记忆链表。

MeDown：我机坠毁动画。

PlaneFlash：我机飞行动画。

Enemy2BulletShoot：敌机子弹发射原则。

Enemy3Flash：自毁舰爆炸动画。

BOOM：自毁舰爆炸条件判断，执行对应效果。

Collisio：检测图像碰撞，依此操作链表。

ClearScreen：方便的清屏，解构。

（6）GameDefine 游戏设定

```
#include <QGraphicsPixmapItem>
#include <QtDebug>
#include <QSoundEffect>
#include <QGraphicsView>
#include <QGraphicsScene>
#include <QList>
#include <QLabel>
#include <QToolButton>
#include <QTimer>
#include <QKeyEvent>
#include <QMessageBox>
#include "player.h"
#include "bullet.h"
#include "enemy.h"
#include "playerbullet.h"
#include "enemybullet.h"
#include "enemy1.h"
#include "enemy2.h"
#include "enemy3.h"

//游戏定义类
class GameDefine
{
    GameDefine();
public:
    static const int PlaneShootUpdateTime = 500;
    static const int Enemy2ShootUpdateTime = 600;
    static const int PlayerMoveUpdateTime = 10;
    static const int EnemyMoveUpdateTime = 10;
    static const int BulletMoveUpdateTime = 10;
    static const int BackgroundUpdateTime = 10;
    static const int EnemyCreateUpdateTime = 1000;
    static const int PlaneFlashUpdateTime = 10;
    static const int PlaneDownUpdateTime = 100;
    //static const int ItemRecycleUpdateTime = 10;

    static const int ScreenWidth = 750;
    static const int ScreenHeight = 1000;


};

#endif // GAMEDEFINE_H
```

游戏基础设定、相关引用头文件的工具类，减少效果类引用头文件篇幅，方便设定整体更改。

三．分工情况

张修瑞：基础效果函数编写、缓存池编写、后期 debug、录屏撰稿。

王政浩：对象继承链实现、工具类和工作台搭建、后期 debug。

肖斌：视图声音素材搜集提取、动作效果和特效动画编写、后期 debug。

大纲由三人共同搭建完成。

四．感言

本篇作者一直坚信人在团队中发挥作用是不可量化的，量化是通过攀比满足羡慕嫉妒本能的借口而已。正如灵感在关键时刻总是比水磨工夫要重要千万倍，人与人的思想习惯和擅长领域不可一概而论。就作者认为，三个火枪手只有一起举剑才有挑战黎塞留的勇气，只有同时立誓才有追寻独特浪漫世界的决心。