

设计文档

设计任务

编写一个“飞机大战”式游戏。

任务分析

为了编写这样的游戏，需要考虑以下几点：

1. 使用图形库还是引擎作为主框架？若使用图形库，选择哪一个？引擎同理。
2. 如何控制程序的主流程？
3. 如何持续渲染画面？如何控制帧数？
4. 如何处理外部输入（键盘、鼠标）？
5. 如何统一管理设置？
6. 如何设计各种实体（飞机、子弹）的抽象层次？
7. 如何设计统一接口对所有实体进行管理？
8. 如何设计可以提高项目的扩展性，使得以后可以非常方便地添加新要素，如 Buff、Bomb、Boss？

设计思路与流程

1. 主框架的选择：

由于课程原先使用的是图形库 MFC，故本项目也采用图形库从零搭建游戏，而不是使用引擎。因某些原因受同学推荐，这里使用一个比较面向新手的基础简易图形库，[Easy Graphic Engine, EGE](#)。

项目采用 Visual Studio 2017 构建。

2. 编写主流程：

根据官方的[俄罗斯方块 demo](#)以及曾经使用过的另一个图形库 [SDL](#) 的经验，这里采用的流程模式为：

- 功能抽象：所有主流程相关功能及游戏相关变量抽象为一个 [World](#) 类。详见[文档](#)。
- 前期准备：进行一些准备动作，如读取设置、材质，初始化各种实体，创建一个窗口等。
- 进入 Main Loop：程序通过一个 Loop 来处理几乎整个运行流程。由于 EGE 库比较简易，没有提供良好的事件机制，这里采用几个函数来顺序处理流程：

```
for (fps ui_fps; world.is_running( ); delay_fps(world.fps( )))  
{  
    world.update( );  
    world.updateCollision( );  
    world.updateState( );  
    world.render( );  
}
```

其中，几个函数的作用为：更新移动状态、更新碰撞状态、更新存在状态，渲染一帧画面。具体介绍详见[文档](#)。

流程通过 is_running()方法来控制是否结束。

- 清除世界，为下一次初始化做准备。

主流程采用了一个 goto 结构，使得游戏可以很容易地在结束后重新开始。

3. 渲染画面及控制 FPS：

对于 Main Loop 来说，每一次循环即处理一帧画面，而这一帧画面通过 world 的 render()函数绘制。

通过在 for 循环的末尾循环体中调用 delay_fps()方法，控制 Main Loop 的执行速度，从而达到控制 FPS 的目的。

	<p>4. 处理外部输入： 为了使得外部输入的处理作为一个独立的组件，将其专门抽象为一个类，即 InputController。详见文档。</p> <p>5. 设计参数设置： 最初，程序使用宏来控制设置，所有宏都放在一个头文件中。这样设计的缺陷很多，如只要修改一个数据，几乎整个项目都要重新编译一次。 现在，程序通过/settings 文件夹中的 JSON 文件来控制游戏设置。详见 Settings 名空间 文档。</p> <p>6. 抽象层次、统一接口设计： 对于所有实体，设计了一个 Entity 基类，提供了足够必要的信息来描述一个实体。派生出 Plane 及 Bullet 基类，来表达各种不同的实体。 对于所有事件，设计了一个 Action 基类，通过回调函数来实现事件的分层及连锁调用。 继承关系见类关系图。</p> <p>7. 扩展性： 在目前的设计下，实际上大部分新东西都可以很简单地扩展实现，如： Boss: 设计新的 Plane 与 Bullet，通过 Action 设计各种特殊的子弹行为（如射出一段距离后子弹爆炸，形成子子弹四散开来） Bomb: 可以设计为 Entity，也可以设计为 Action，通过一个圆环扩散，发生碰撞时销毁对应 Entity。 Buff: 设计为一个新的 Entity 派生类，当碰撞时发生一个 Action，或是修改飞机的设置数据。</p>
功能介绍	<p>1. 进行飞机对战游戏： 玩家操控一个迷路的凤凰战机（by 星际二），与泰伦帝国的人族追军进行战斗，突破这片巡逻区。 玩家飞机拥有 100 的血量，通过 W-A-S-D 按键操控方向，Space 射击。 敌人有两种：</p> <ol style="list-style-type: none">1. 新兵敌机：特点为血少，子弹伤害低，路径单一，会飞离屏幕；但子弹速度快，射击间隔短。玩家飞机默认子弹可以一发击毁。2. 自机狙敌机：特点是血厚，不会飞出屏幕，子弹追踪玩家飞机，伤害高；但子弹速度较慢，射击间隔长。玩家飞机默认子弹需要三发击毁。 <p>玩家飞机在击中乃至击毁敌机后会获得积分。随着积分越来越高，难度等级会逐渐提升。 玩家飞机生命值归零后，游戏结束。结束界面会显示玩家的分数，以及根据难度等级给予玩家相应的称号。</p> <p>2. 难度分级： 随着难度等级的升高，敌机将开始表现出不同的特性。具体为：</p> <ul style="list-style-type: none">● 等级 1：无特殊表现。● 等级 2：新兵敌机子弹的速度将在一个区间变动，初始为 0.8~1.2 倍原速度。随着难度越来越高，区间将变得越来越宽。 <p>这样做会使得新兵飞机构造出一个子弹网，使得玩家飞机难以通行。</p>

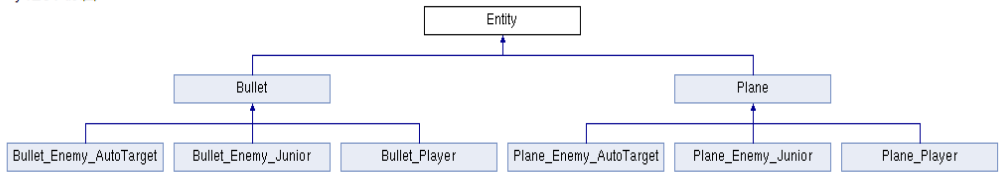
	<ul style="list-style-type: none">● 等级 3: <i>自机狙敌机</i>每次射击时, x 轴速度会相对于默认速度按一定倍数变大。难度越高, 倍数越大。● 等级 4: 暂无特殊表现。● 等级 5: <i>自机狙敌机</i>若受到伤害且未坠毁, 则立即进行一次复仇射击。该次射击不影响之前的射击冷却。 <p>此外, 还有一些每个等级都会固定更新的特性, 具体为:</p> <ul style="list-style-type: none">● 敌机新一波刷新冷却时间降低● 敌机刷新数量变多。新兵敌机与自机狙敌机数量错开增加。 <p>难度分级与积分的关系见相关文档介绍。</p> <p>3. UI 设置:</p> <p>UI 将显示当前生命值、分数及当前难度等级。同时, 由一个指针来追踪当前玩家击中的敌人; 若指针不为空, UI 将在一定时间内显示被追踪敌人的血量。</p>																														
文件与对象描述	<p>文档地址: https://vigilans-yea.github.io/Lost-Phoenix/html/</p> <p>文件夹结构:</p> <ul style="list-style-type: none">/├─ /assets: 材质资源文件夹, 保存了各类贴图。未来可以加入声音等资源。├─ /settings: 设置文件夹。由 JSON 文件保存。<ul style="list-style-type: none">├─ GeneralSettings.json: 全局设置。├─ PlaneSettings.json: 所有飞机设置。└─ TextureSettings.json: 所有材质设置。├─ /include: 代码头文件库。├─ /src: 代码实现源文件及内部类文件。├─ Lost Phoenix.rc: Visual Studio 资源文件。主要用来处理 exe 文件图标。└─ Lost Phoenix.vcxproj: Visual Studio 项目文件。 <p>代码文件描述:</p> <table><tr><td>▼ Lost Phoenix</td><td></td></tr><tr><td>▼ include</td><td></td></tr><tr><td>Actions.h</td><td>定义Action基类以及其所有派生类。</td></tr><tr><td>Enemy_AutoTarget.h</td><td>定义自机狙敌机的飞机类及子弹类。</td></tr><tr><td>Enemy_Cruiser.h</td><td>BOSS类, 暂时未实现。</td></tr><tr><td>Enemy_Junior.h</td><td>定义新兵敌机的飞机类及子弹类。</td></tr><tr><td>Entity.h</td><td>定义所有实体的基类, 以及一个阵营的枚举。</td></tr><tr><td>InputController.h</td><td>定义输入控制器类。</td></tr><tr><td>Plane - Bullet.h</td><td>定义飞机及子弹的基类, 以及两个"dealDamage" (子弹->飞机, 自机->敌机) 全局函数。</td></tr><tr><td>Player.h</td><td>定义玩家的飞机及子弹类。</td></tr><tr><td>Resources.h</td><td>资源类, 几乎会被所有头文件包含。定义了材质类及设置名空间, 及若干重要全局函数。</td></tr><tr><td>Vector2D.hpp</td><td>定义数字二维向量模板类, 方便数据的组织及计算, 提高可读性, 以及提供极强的扩展性。</td></tr><tr><td>World.h</td><td>定义程序的核心类World类, 以及有一个world实例的extern声明。</td></tr><tr><td>▼ src</td><td></td></tr><tr><td>ResourcesLoader.h</td><td>定义材质加载类。非公共类, 仅会被Resources.cpp包含, 隐藏于实现中。</td></tr></table> <p>外部使用库: misakamm/xege (GUI 库)</p>	▼ Lost Phoenix		▼ include		Actions.h	定义Action基类以及其所有派生类。	Enemy_AutoTarget.h	定义自机狙敌机的飞机类及子弹类。	Enemy_Cruiser.h	BOSS类, 暂时未实现。	Enemy_Junior.h	定义新兵敌机的飞机类及子弹类。	Entity.h	定义所有实体的基类, 以及一个阵营的枚举。	InputController.h	定义输入控制器类。	Plane - Bullet.h	定义飞机及子弹的基类, 以及两个"dealDamage" (子弹->飞机, 自机->敌机) 全局函数。	Player.h	定义玩家的飞机及子弹类。	Resources.h	资源类, 几乎会被所有头文件包含。定义了材质类及设置名空间, 及若干重要全局函数。	Vector2D.hpp	定义数字二维向量模板类, 方便数据的组织及计算, 提高可读性, 以及提供极强的扩展性。	World.h	定义程序的核心类World类, 以及有一个world实例的extern声明。	▼ src		ResourcesLoader.h	定义材质加载类。非公共类, 仅会被Resources.cpp包含, 隐藏于实现中。
▼ Lost Phoenix																															
▼ include																															
Actions.h	定义Action基类以及其所有派生类。																														
Enemy_AutoTarget.h	定义自机狙敌机的飞机类及子弹类。																														
Enemy_Cruiser.h	BOSS类, 暂时未实现。																														
Enemy_Junior.h	定义新兵敌机的飞机类及子弹类。																														
Entity.h	定义所有实体的基类, 以及一个阵营的枚举。																														
InputController.h	定义输入控制器类。																														
Plane - Bullet.h	定义飞机及子弹的基类, 以及两个"dealDamage" (子弹->飞机, 自机->敌机) 全局函数。																														
Player.h	定义玩家的飞机及子弹类。																														
Resources.h	资源类, 几乎会被所有头文件包含。定义了材质类及设置名空间, 及若干重要全局函数。																														
Vector2D.hpp	定义数字二维向量模板类, 方便数据的组织及计算, 提高可读性, 以及提供极强的扩展性。																														
World.h	定义程序的核心类World类, 以及有一个world实例的extern声明。																														
▼ src																															
ResourcesLoader.h	定义材质加载类。非公共类, 仅会被Resources.cpp包含, 隐藏于实现中。																														

[nlohmann/json](#) （标准库风格的 C++ JSON 库）

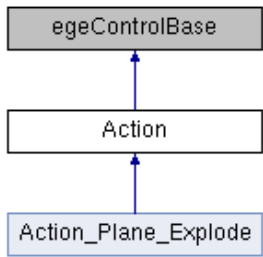
类/名空间关系图：

Entity:

类 Entity 继承关系图:



Action:

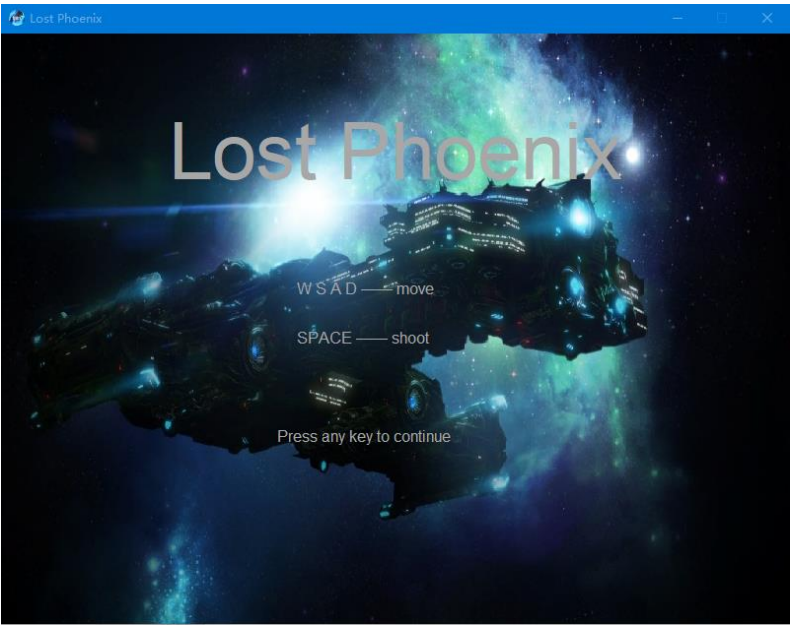


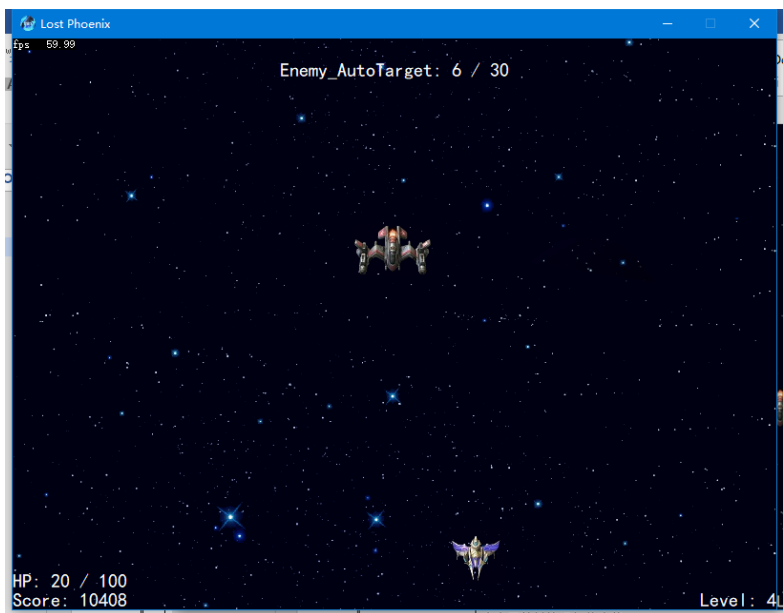
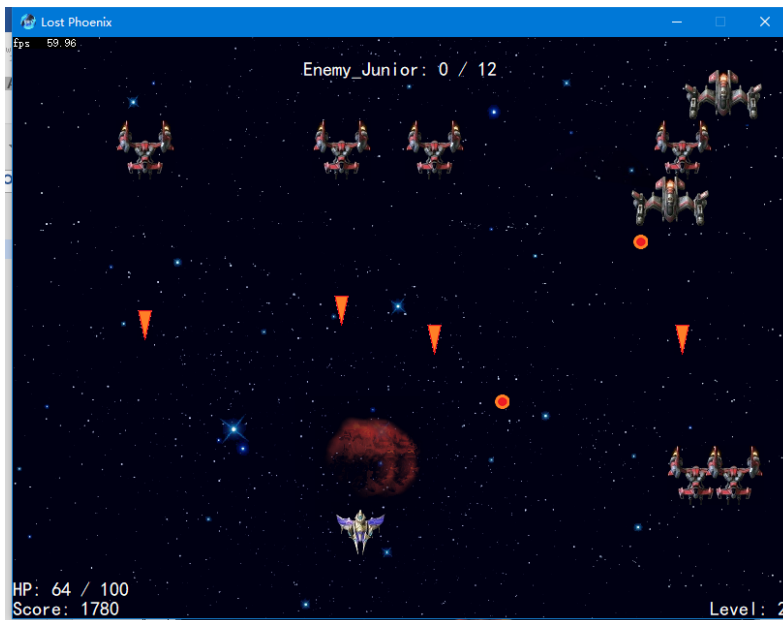
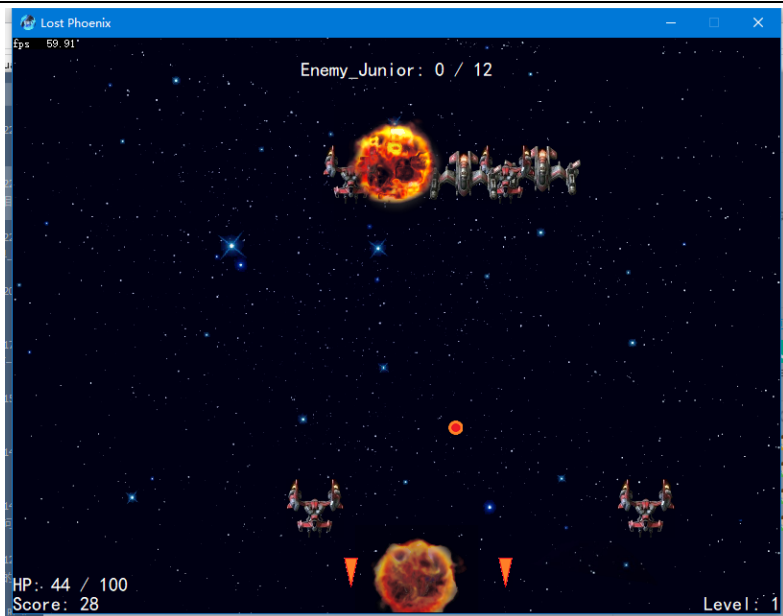
Settings 名空间描述：

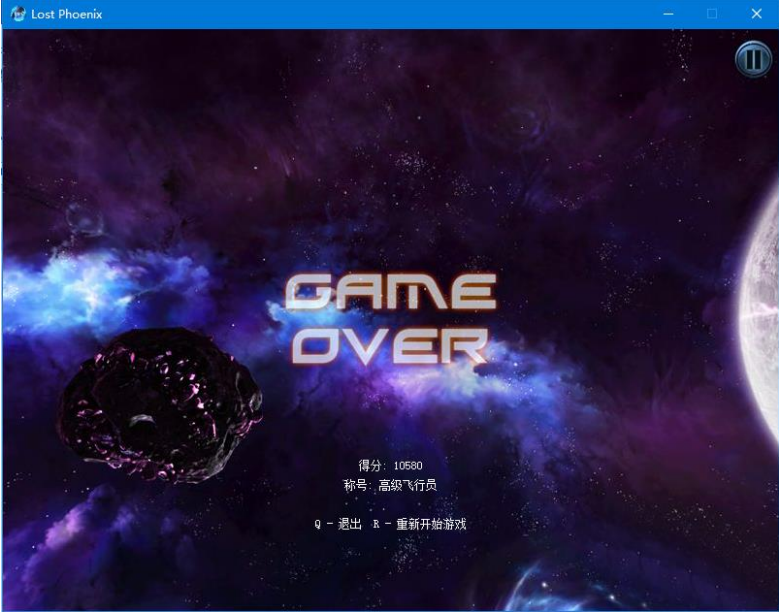
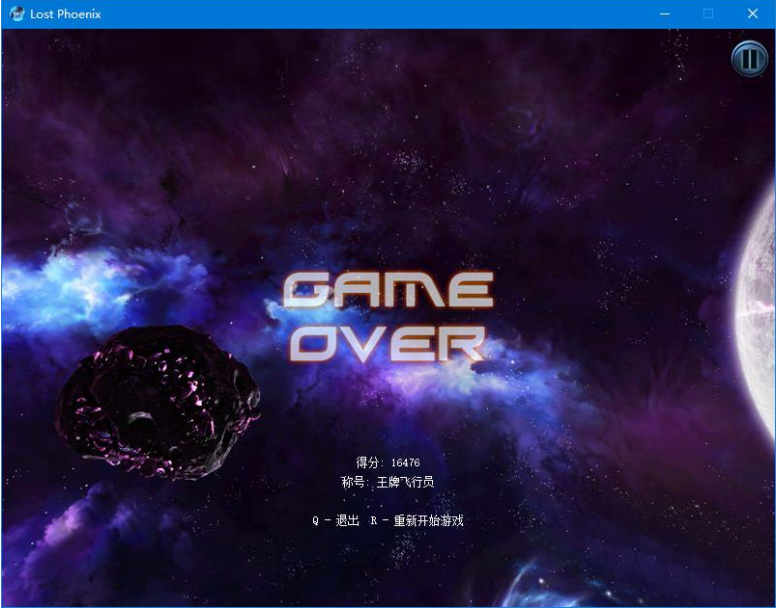
▼ N Settings	封装了游戏全局与所有实体设置的名空间。所有获取默认设置的方法采用 <i>Lazy Load</i> （延时加载）机制。推荐参考详细描述来获取完整信息。
☑ AnimeTextures	所有动画、 Action 相关材质设置。
☑ BgTextures	所有游戏背景材质设置。
☑ Bullet	子弹的设置基类。
☑ General	游戏全局设置，主要包含UI与相关时间设置。
☑ Plane	飞机的设置基类。
☑ TextureInfo	材质的设置源，可以用来构造一个完整的材质。

本游戏项目暂未提供单元测试及集合测试，所有测试通过直接游戏来完成，故在此处贴出游戏屏拷：

测试报告





	<div></div> <div></div>
总结	<p>未来可以继续添加的功能：</p> <p>参考 GitHub Project:</p> <p>https://github.com/Vigilans-Yea/Lost-Phoenix/projects</p> <p>以及添加排行榜（文件读写）、菜单中键位及其他设置及地图控制功能。</p>