建议的开始流程

0. 先成功编译一次

在你开始写第一行代码之前，试着先成功编译并打开我们提供的代码框架。你应当能看到游戏的标题画面。在按回车开始游戏之后，你会看到只有背景的游戏画面。此时你看到的就是正在运行中的游戏，只是游戏里面什么也没有。

1. 背景

（为了方便大家理解，Background类的声明和实现已经为大家写好，作为example）

你要做的第一件事便是显示出“游戏画面”，也就是背景。你需要为背景创建你的第一个类，它将继承 `GameObject` 。在这时，你大可不必立刻规划好“什么成员属于 `GameObject` 基类，什么成员应当单独存储”。在你实现更多功能的过程中，这个问题会渐渐变得清晰。

创建完背景类后，你需要在 `GameWorld` 中生成一个背景。在 `Init()` 中生成一个临时变量当然是不行的，即使这个临时变量是给它动态分配内存的 `shared_ptr` ，也会因为离开 `Init()` 作用域（scope）时被销毁，然后分配的动态内存也会释放。
你应当把这个背景存储在 `GameWorld` 里。根据文档，所有 `GameObject` 都应存储在一个 `list` 里。考虑一下这个 `list` 的类型会是什么？

一切顺利的话，现在运行游戏你就能看到背景图片。如果你遇到编译错误或链接（linking）错误，看看在用到背景类的时候是否正确 `#include` 了它的文件？参考 FAQ 中的 Linking error 部分。

2. 放置玩家

接下来，你可以真正开始“写游戏”了。如果你感觉无从下手，就从先创建一个玩家开始，相信你在实现完之后就能明白游戏中任何部分的开发流程。

你需要先在 `Player.hpp` 里面**声明** `Player` 类，然后声明它的private成员变量（或许你这会还不知道需要哪些变量），但是没关系，最开始要做的工作是为 `Player` 声明public函数，比如构造函数， `void Update()` 等虚函数的重载。

然后再在 `Player.cpp` 里面完成对 `Player.hpp` 的成员函数的**实现**。

> 尽量把函数的实现放在 `.cpp` 文件里，这是因为这些函数的实现可能会牵扯到非常多的 `*.hpp` 文件，比如 `gameworld.hpp` ，而 `gameworld` 的实现又依赖于 `Player.hpp` ，因此这样会产生循环include，正确的做法是在 `*.hpp` 中只include有继承关系的头文件，在 `*.cpp` 中则可以include各种需要的头文件。因为被编译的只有 `*.cpp` 文件，生成的二进制可执行文件 `*.o` 之间是彼此链接的关系而不是include。

在正确完成 `Player` 的构造函数之后，玩家其实就可以显示在屏幕上面了，只需要在 `Gameworld` 的容器中像添加背景一样添加Player即可

如果你需要让他动起来(实则背景向后跑动），则需要正确完成 `Gameworld` 中对所有容器对象的遍历 `Update()` ，你就能看到你的玩家像是跑起来了。

3. 让玩家开枪

如果你成功地完成了上面的步骤，你应该对“ `GameObject` 需要存储什么成员”有了更多的想法。试试继续让这个游戏动起来吧：产生子弹。每次按下J或者鼠标左键时候即让 `Gameworld` 容器中加入一个子弹

4. 自由发挥

剩下的部分里，我相信你们已经不会遇到太多问题了，开始自由发挥吧！