程序一共包含 26 个类，从功能上来分可以分为六个模块，游戏基础模块，物品交互模块，主角模块，界面与界面控制模块，敌人模块以及综合控制模块。

（1） 界面与界面控制模块

本模块所有类都有基类 scene 类公有派生而来，分别有一下几个类：

1、 menu 类,即游戏主菜单,有 display()函数执行输出,有 mouse_check 函数进行鼠标选择的检测，主菜单包含了三个按键：帮助，开始游戏以及排行榜的入口。

2、 help 类，即游戏帮助页面。详细说明了游戏的操作方法，并且有返回主菜单的鼠标检测函数。

3、 record 类，是记录个人通关时长排行榜的纪录类。其构造函数可以在子目录下创建二进制文件，主要负责对新纪录的读入以及排序，以及在进入查看排行榜后读出文件内容并且输出到窗口

4、 levelselect 类，关卡选择界面，通过鼠标检测选择了哪个关卡，把选择的关卡编号传给 map 类。

5、 map 类，包含了成员 enemy_controller ,stone_controller, medicine_controlle（控制模块的类），通过 levelselect 输入的编号进行不同地图的初始化，内置有 attack 和 move 两个更新函数，以及 show()这一展示函数。

6、 stoppage 类，暂停类，在游戏暂停是展示，可以选择继续游戏或者返回 levelselect 界面

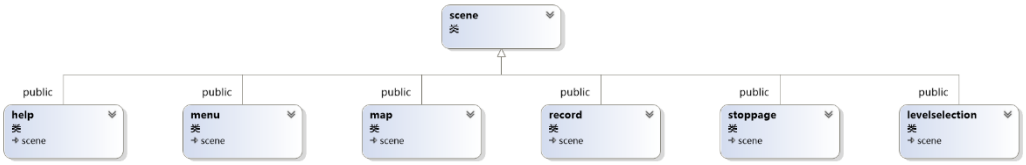

图 1 界面与界面控制模块视图

（2） 主角模块

此模块有一个 role 类

主角模块的 role 类由 unit 类公有派生而来，unit 类表示一个单元，内置了坐标，速度，血量等等数据成员以及对应的访问函数接口。

主角类包含大量的函数,主要控制了主角的行走(跳跃,二段跳,左右移动)，形态切换，不同形态下的攻击，使用理智剂（药品）后回血，以及打怪的成长值更新等等。

同时，还有主角的落地判定，撞墙判定，撞天花板判定等等特殊判定函数。

（3） 敌人模块

敌人模块先由 unit 类派生出 enemy 类，再由 enemy 类分别派生出三种小怪和 boss 类这四个类。

Enemy 类：继承了 unit 类的坐标，高度宽度，血量等等数据成员，同时自己定义了攻击 cd，帧数计数器（用于实现敌人的行走和攻击动画），索敌范围，以及每个小怪死亡后给主角的成长值等等新成员，同时还有基础模块的类成员 bullet 和 sword（子弹类和刀剑类，分别负责远程和近战），函数方面包括了近战攻击判定，远程攻击判定，子弹位置更新和伤害判定等等函数，同时还定义了 show(),move_update() ， attack_update 等四个虚函数，方便用 enemy_controller 对小怪进行动态联编。

下面是三种小怪类，由 enemy 类派生

Insect 类（此类在游戏中最终使用的不是昆虫小怪贴图，但是不影响功能实现）：昆虫小怪，索敌方式主要采用追击式，被角色唤醒后会不断靠近角色，采用的是近战攻击方式。重写了 enemy 类的四个虚函数，具有攻击更新，位移更新，展示动画的功能。

Scorseror 类：术士小怪，造成远程伤害，在固定范围内巡逻，检测到主角靠近会发射子弹。重写了 enemy 类的四个虚函数，具有攻击更新，位移更新，展示动画的功能。

Elite 类：精英小怪，造成远程伤害，攻击范围更远，血量更高，在固定范围内巡逻，检测到主角靠近会发射子弹。重写了 enemy 类的四个虚函数，具有攻击更新，位移更新，展示动画的功能。

下面是 boss 类

Blacksnake：boss 黑蛇，同样复写了虚函数。其攻击方式相比小怪更加多元，有近战，远程，二阶段苏醒时还新增了一个大招，除此之外，boss 也有落地判定，有更加丰富的追击方式。

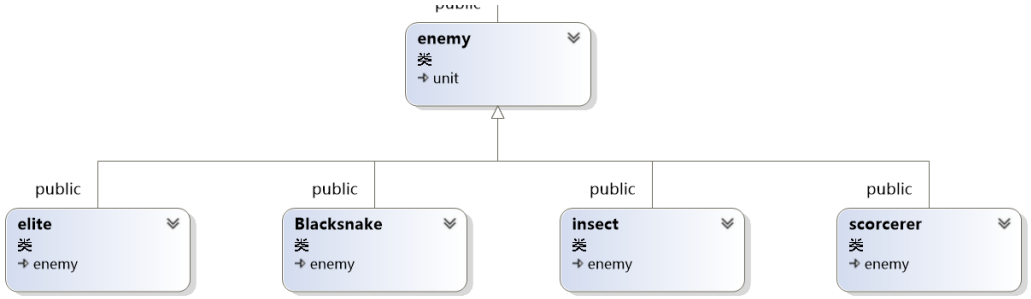

图 2 敌人模块类视图

（4） 游戏基础模块

基础模块包含了三个类，即子弹类，刀剑类和 unit（单元）类，分别作为其它类的基类或成员

Unit 类：存储了单位的坐标，高度宽度，状态 state 等基本信息，并提供了访问接口

Bullet 类：子弹类，由 unit 类派生，存储了子弹的发射状态，发射速度，负责子弹位置的更新，命中的判断。

Sword 类：刀剑类，存贮了近战伤害的伤害和范围，负责判定是否命中并返回伤害。

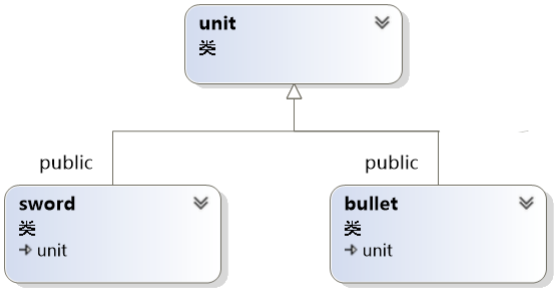

图 3 基础模块类视图

（5） 物品交互模块

首先是 object 类，由 unit 类派生而来，主要负责碰撞的判断，其函数有多个重载，通过传入 unit 类指针或者 role 类指针，可以做出碰撞判断，用于实现主角和 boss 的碰撞判断，以及所有单位是否撞墙的判断（撞墙以后速度为零或反向）

Medicine 类：由 object 类派生而来，定义，存储药品的坐标，展示的图片。

Fixture 类：固定物类，由 object 派生而来，存储地板坐标，以及状态，同时有两个虚函数 show（）函数和 change（）函数，方便动态联编。

Fixturel 类：由 fixture 类派生而来，黑色地板，不造成伤害，内置与 boss 交互函数，boss 在二阶段会随机把黑色地板改变成会造成伤害的火地板

Fixture2 类：由 fixture 类派生而来，火地板，造成伤害。

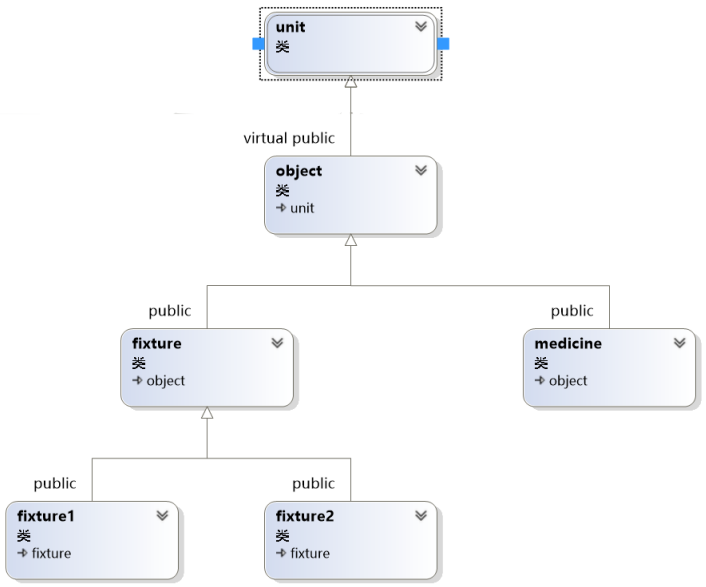

图 4 物品交互模块类视图

（6） 综合控制模块

分别负责对各个模块的总领统筹，部分类还有基类指针用于动态联编。

Game 类：游戏类，数据成员包含所有界面与界面控制模块的类，用于界面切换，游戏运行，以及对应界面下的音乐播放，同时还有一个 role 类成员作为主角。

Enemy_controller 类：内置一个 enemy 类的指针数组成员，通过指针访问 enemy 类的虚函数，实现了不同怪物在各自运行和攻击逻辑下的更新。

Stone_controller 类：内置了一个 fixture 类的指针数组成员，通过指针输出不同的地面，以及做出不同的伤害判定。

Timer 类：时间类，用于记录游戏时长用于更新排行榜，同时支持输出系统日期用于排行榜的通关日期的记录

图 5 综合控制模块类视图

三、过程论述

前置说明：

1、 游戏运行逻辑说明：设定固定时间 50ms 作为一帧，每一帧内读取角色的键盘控制，各种更新以及输出画面，如果运行时间不足 50ms 补足到 50ms，接着进入下一帧。

2、 动画更新说明：每一个小怪，boss 和角色的分别有不同的数据成员记录当前动作运行到第几帧，例如 distant_attack_frame 记录了远程伤害进行到第几帧，对应在 show()函数在每一帧调用不同的图片，以此实现了动画播放的效果。

3、 状态的设计：我为每个小怪，角色以及 boss 都设计了 state 这一变量，用于记录当前的状态，以此来调用不同的更新方式。我使用了 stand，toattacclose（远程攻击），toattckdistant（近战攻击）、walk 等状态（通过宏定义赋予各不相同的值）

4、 相对移动的设计。Map 类中有一个重要参数 xM，用来记录所有地图内成员的移动距离。为了达到人物和场景不断移动，当角色位于屏幕最左边和最右边一段距离的时候，场景移动，而当角色位于中心的时候，则改变角色坐标，以此实现场景动态变换的效果。

下面是具体过程论述

（1）界面与界面控制模块

本模块所有类都有基类 scene 类公有派生而来，分别有一下几个类：

1、 menu 类，即游戏主菜单

主要函数有析构函数，display()（展示），check（鼠标点击检测）

析构函数读取了主菜单的背景图片，并且初始化了三个 unit 类的成员 button（按钮）的坐标位置。

display 函数展示了主菜单的背景图片以及用 outtextxy 和 solidrectangle 两个 easyx 内置函数输出按钮。

check 函数根据传入的两个坐标（从 game 类中读取的鼠标点击坐标并传入 menu 类），判断按到了哪个键位，并且返回给 game 函数。

2、 help 类

主要函数有构造函数，show()（展示）函数

构造函数初始化了一个 unit 类成员 button 的位置，该 button 用于返回主界面

Show()函数用于输出帮助文字

Check 函数同样也是根据传入的坐标，判断是否按到返回界面，并把结果回传给 game 函数

3、 record 类

record 类封装了一个 record_time 结构体，包含变量 point（分数）以及 SYSTEMTIME 变量 date（windows.h 内置变量，可以存储系统时间）

主要函数有构造函数,show()函数(展示),mouse_check 函数,record_update 函数

构造函数通过先通过 ofstream 打开文件，重复利用其如果没有文件会自动创建的特性，接着用 ifstream 打开文件，如果文件为空，则写入一个结束标识符，即一个 point 为-1 的 record_time 变量。三个关卡的记录分开三个文件存储。

Mouse_check 函数，封装了对返回键的点击检测

Record_update 函数，在每次通关后从 game 函数中调用，传入地图编号，而函数通过一个 switch—case 结构打开对应关卡的文件，并且用一个 iftream 变量读出文件内容,读到结束标识符的时候结束。此时把读出的所有记录的时间(前十快的通关记录）用 STL 中重载的 sort 函数进行排序，选出新的前十快的通关记录。再用 ofstream 打开文件写入新的排名，并且打上文件结束标识符。由此实现排行榜纪录的更新。

Show 函数，用于将排行榜结果打印。同样先用 ifstream 打开文件，通过循环不断读出文件内容并利用 easyx 的函数输出到游戏窗口上。

Record 类中所有对文件的操作都是使用二进制方式打开，以此确保安全性。

4、 levelselect 类和 stoppage 类

主要函数构造函数，check 函数，构造函数用于初始化按钮位置，而 check 函数用于检测鼠标点按的按钮并且传回结果。还有 show（）函数用于输出画面。

5、 map 类，

map 类是界面与界面控制模块中最重要的类，担负着游戏地图创建，画面与玩家控制的角色的交互等等。

主要包含构造函数，init（初始化）函数，move（移动）函数，attack（攻击）函数，show（展示）函数，check（检查函数）

构造函数用于加载游戏背景图，初始化暂停按钮位置和大小，初始化计时器时间。

Init 函数用于初始化地图，函数需要传入地图编号，而 init 函数把编号分别传给成员 enemy_controller,stone_controller,以及 medicine_controller,

让他们分别生成游戏的敌人，地面以及理智剂（药品）的位置，并且初始化血量等等参数。

Move（移动的更新）类用于更新地图中所有的物品和敌人的位置，最重要的是获取玩家视角的更新。本游戏在视角移动上别处心裁，在角色到达距离左右边界一定距离时——场景移动，而角色在屏幕中间的时候则是移动角色，从而能够营造出相对移动的效果，极大的增加游戏手感和体验。Move 类需要传入 role 类的引用，通过读取 role 类中的坐标和速度（先运行 role 类的 move，基于键盘动作计算好角色状态和移动速度）判断相对位移。除此之外，map 类还作为 stone_controller 和 role 类的中间人，在 map 类中将 role 类对象的引用传入 stone_controller，以此判断角色是否触地，是否撞墙,，在此基础上对角色的状态和速度进行进一步的更新。此后，再调用数据成员 enemy_controller 中的 move()，传入 role 类对象以此对敌人的移动做出更新，

Attack（攻击更新）函数，需要传入 role 类成员的引用，通过调用 enemy_controller 中的 attack 函数，进行小怪和 boss 以及主角的攻击状态更新。

Show(画出图像)函数,分别调用成员 enemy_controller, stone_controller，以及 medicine_controller 的 show()函数，需要传入背景的位移量 xM，用于画出所有道具，敌人的图像。

Check（检测点击动作）用于判断玩家的鼠标操作，判断是否暂停

Map 类作为游戏场景，是角色模块，敌人模块和玩家交互的重要桥梁。

（2）主角模块

role 类包含了构造函数,move 函数(移动更新),creat_bullet(生成子弹)，bullet_update（子弹更新），bullet_check(子弹是否命中)，buhert（受伤扣血判断），creat_sword（生成近战伤害判定区），growing_update（成长值更新），show（画图），以及各种对 role 类数据成员的访问接口。

构造函数，读入所有图片文件，对角色血量，攻击伤害，状态，攻击冷却时间，成长值上限等做初始化。

Move（移动更新），首先利用 GetAsyncKeyState 函数获取键盘动作，接着对键盘动作做出逻辑判断(例如切换形态 cd 是否冷却,攻击时移动速度减半等等)，基于这个对 role 的状态进行更新。

子弹更新原理：role 类具有类成员 bullet 数组，在获取到创建子弹的命令时，调用 creat_bullet 函数，遍历整个 bullet 数组，寻找没有被发射的子弹，一旦找就发射并且退出遍历进程，每一帧同样要遍历所有子弹，当遇到已经被发射的子弹的时候，通过 bullet 内置的更新函数更新子弹位置。更新在 map 类的

attack 函数中完成。而 bullet_check 则在 map 被传入 stone_controller 和 enemy_contorller 后调用，分别用于判断是否碰到障碍物以及是否命中敌人并造成伤害。因为子弹在设定上到达一定距离后会消失，所以 bullet 数组不会出现“供不应求”的情况。

近战攻击原理类似。

（3）敌人模块

敌人模块先由 unit 类派生出 enemy 类，再由 enemy 类分别派生出三种小怪和 boss 类这四个类。

1、Enemy 类

包括 checkawake（判断是否唤醒），attack_close（近战索敌），attack_distant（远程索敌），getspeed（获得速度），{behurt, bullet_update, attack_update, show, move_update}(纯虚函数)

Checkwake 函数用于判断敌人是否被唤醒，其中 boss 和 insect 类小怪会用到这个判断，一开始他们的 wake（一个布尔成员，判断是否苏醒）都为 0，当主角靠近时进行判断并且改变 wake 的值，需要传入 role 类引用。

attack_close（近战索敌），attack_distant（远程索敌）分别是两种攻击索敌方式，但是总体原理类似，首先需要敌人的状态是 walk（确保在攻击时不会重复索敌）当角色和敌人都在地面，底部的 y 坐标相等，并且 x 轴上距离小于数据成员 attack_distant_range 和 attack_close_range 的时后，改变敌人的状态为 toattackclose 和 toattackdistant，进入攻击状态。

Getspeed 函数，用于更新敌人的速度，通过敌人和主角的相对位置，改变敌人的行动方向，速度，达到追击的效果。

对于五个纯虚函数：

Behurt 用于受伤判定。

bullet_update 更新子弹

attack_update 包含所有的攻击更新，enemy_controller 只会调用这个做攻击更新

show 展示图片，因为每一种敌人攻击速度，图片张数各不相同，因此需要使用虚函数重写

move_update 包含所有移动更新，enemy_controller 只会调用这个作移动更新

对于派生的三种怪物和 boss 类的构造函数将不再赘述，主要功能都是加载动画图片以及初始化数值。

下面是三种小怪类，由 enemy 类派生

2、Insect 类：

重写了 enemy 类的虚函数，主要攻击模式：近战攻击，唤醒后追击主角。

3、Scorseror 类：

重写了 enemy 类的四个虚函数，具有攻击更新，位移更新，展示动画的功能。

4、Elite 类：精英小怪，造成远程伤害，攻击范围更远，血量更高，在固定范围内巡逻，检测到主角靠近会发射子弹。重写了 enemy 类的四个虚函数，具有攻击更新，位移更新，展示动画的功能。

下面是 boss 类

5、Blacksnake：boss 黑蛇，同样复写了虚函数。其攻击方式相比小怪更加多元，有近战，远程，二阶段苏醒时还新增了一个大招，除此之外，boss 也有落地判定，有更加丰富的追击方式。

（4）基础模块

基础模块包含了三个类，即子弹类，刀剑类和 unit（单元）类，分别作为其它类的基类或成员

1、Unit 类：存储了单位的坐标，高度宽度，状态 state 等基本信息，并提供了访问接口

2、Bullet 类：

主要函数 create 和 checkhit

Create 函数用于创建新的子弹单位，将子弹的发射状态改为一发射，并且初始化伤害

Checkhit 函数，需要传入一个 unit 类对象，通过 unit 类的坐标，高和宽，构造一个碰撞框，子弹类同样有一个碰撞框，本函数即判定两个碰撞框是否重合，如果重合则子弹命中

3、Sword 类：刀剑类，存贮了近战伤害的伤害和范围，负责判定是否命中并返回伤害。，同样也有一个 checkhit 函数用于判断碰撞

（5）物品交互模块

1、Object 类

主要函数有构造函数，check_l,check_r,check_ground,check_ceiling 函数，分别用于左右上下的碰撞判定，并且各有一个重载函数，分别用于 unit 类和 role 类的判断。

构造函数：有两个重载，一个为无参数的默认构造，另外一个需要传入物体的高和宽用于初始化

Check_l 和 check_r 函数，两个水平 x 轴上的碰撞判断，其判断原理为通过物体本身和传入的类对象的两个碰撞框，如果二者重合，则判定为碰撞（实际为镶嵌），并将传入对象的速度变为 0；

Check_ground 和 check_ceiling 函数负责竖直 y 轴上的碰撞判断，其碰撞方式与 x 轴上的判断有所不同（与竖直方向有重力 g 有关），即获取传入对象的 x 轴坐标与自己的 x 坐标进行比对，如果在 x 轴有重合部分，则继续接下来的判断。在 y 轴上，获取传入对象的 y 轴坐标和 y 轴速度 vy，如果 y 轴坐标在 object 之上，加上 vy 后坐标 y 变为 object 下面，说明该 unit（或 role）对象，会穿过物体，因此直接把物理的 y 坐标减去传入对象高度赋给传入对象的 y，从而达到正好落到物体上的效果。

以上四个判断函数都有不同的重载，这是因为敌方单位处于场景本身的坐标系，需要 xM 的调整才能到达主角的坐标系，反之亦然。作为同为场景坐标系的物体和怪物不需要传入 xM，而 role 类与 object 的碰撞则需要 xM 的调整。其实此处也可以用默认参数的方法优化，但是为了代码的可读性，方便后期修改，我保留了原有重载。

2、Medicine 类：主要有构造函数，init 函数，check 函数以及 show 函数

Init 函数：根据传入的地图编号变量，用于初始化理智剂（药品道具）的位置。

Check（拾取检查函数），用于检查是否捡到道具，也是通过碰撞框重合的方法判断

Show 函数，用于画出图像

3、Fixture 类：

拥有两个纯虚函数，分别为{change，getimage}

Change 函数用于与 boss 交互时改变地板的形态

Getimage 函数用于输出时根据地板的编号获取不同的图片照片

4、Fixture1 类：重写了基类的两个虚函数，其伤害这一数据成员在构造函数中初始化为 0；

5、Fixture2 类：由 fixture 类派生而来，火地板，造成伤害。

（6）综合控制模块

1、Game 类：

游戏类，数据成员包含所有界面模块的类，用于界面切换，游戏运行，以及对应界面下的音乐播放，以及一个 role 类成员作为主角，同时还有一个重要的状态成员 state，除此之外，还有一个用于记录游戏时间的计时器

Game 类下只有一个 play()函数，用于游戏的统筹运行。

首先是 game 的运作原理，根据不同的界面，都有对应的界面编号与之对应，在一个大的 while 循环内，每个界面都有一个 while（state==number）作为对应界面的运行基础。当前 game 类的 state 为某个界面时，就不会进入其他的循环，而当 state 改变时，会马上跳出当前循环进入对应改变后的循环。

而对应进入每一个界面以后，都会播放对应的界面音乐，并且提供了实时的鼠标检查，随时传入对应界面的鼠标检查函数。

其中 map 类的过程较为关键复杂，下面详细阐述。

在进入循环后，首先重置计时器以及初始化 map 类对象，其次，先执行 role 的 move 函数，再运行 map 类对象的 move()和 attack()两种更新，需要传入 role 类对象。期间检测是否有暂停的操作，如果有则进入暂停页面（stoppage 界面），并且暂停计时器。在所有更新操作结束以后，运行角色和 map 类对象的 show 函数，最后再计算这一帧已经运行的时间，如果时间不够一帧就补齐。

当角色血量为 0 或者角色到达地图终点时跳出循环，做胜利和失败的判断。

在循环中使用了 BeginBatchDraw()和 EndBatchDraw 函数，即双缓冲输出，防止屏幕闪烁。

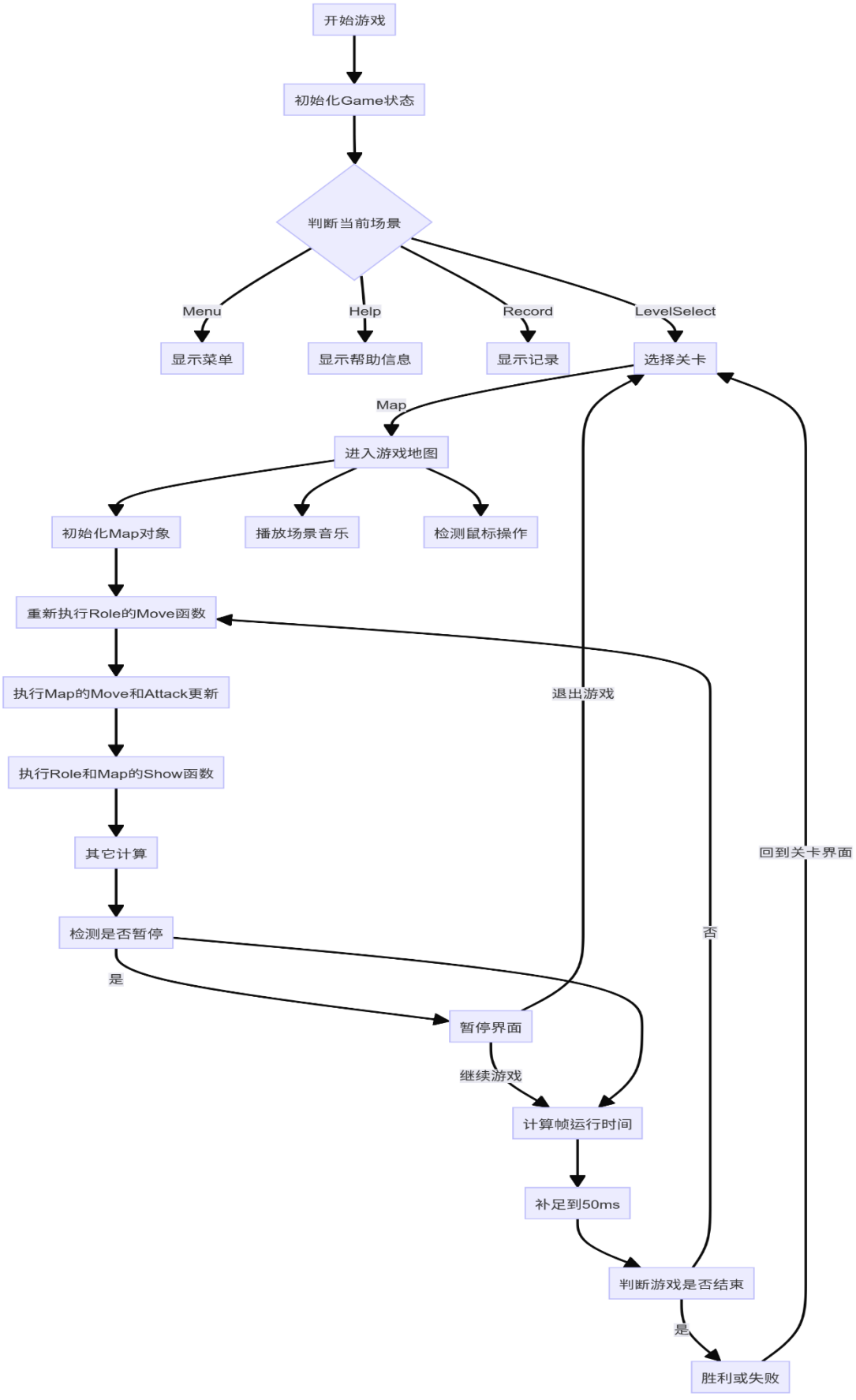

图 7 game 类 play 函数运行流程图

2、Enemy_controller 类：内置一个 enemy 类的指针数组成员，通过指针访问 enemy 类的虚函数，实现了不同怪物在各自运行和攻击逻辑下的更新。

Enemy_controller 的核心在于拥有一个 enemy 类的指针数组成员，利用动态联编的思想，可以做到对不同敌人进行快速的操作。

主要包含了 init（初始化）函数，bullet_update 函数，move_update 函数，grow_check（小怪死亡判断以及角色的对应成长），attack_check(索敌判断和更新)，show（展示），role_attack_close（角色近战攻击更新）

其中更新类以及 show()函数基本原理相同，即遍历 enemy 类的指针数组，分别调用各自的重写的纯虚函数。

Init()函数通过传入的地图 id，进行地图的初始化，即小怪位置的设定和重置。

3、Stone_controller 类：内置了一个 fixture 类的指针数组成员，通过指针输出不同的地面，以及做出不同的伤害判定。

主要内置了 check_lr（左右撞击）,check_onground（是否在地面），check_ceiling（是否接触天花板）以及 bullet_check（子弹撞墙判断），都是通过调用指向的 fixture 类中从 object 中继承来的各种检查函数，而此类做一个统筹的效果。

4、Timer 类：时间类，包括一个 double 型成员 time，用于计算累加的时间并且返回当前的时间，有一个 init 初始化函数用于将 time 归 0