评分

本题的评分采用线下 check 的方式。本次 check 的评分整体分为三个部分组成,其中基础功能和扩展功能为加分项,而代码规范部分为扣分项。在实现代码之前,可以仔细思考代码规范中提到的设计思路,来预先对整个游戏工程的代码做好设计。

基础功能和扩展功能中的各项不再设部分分,**如果有 bug 则视为未实现**,所以请务必测试你实现的功能。

基础功能实现 (50%)

以下的功能为加分项,功能应该正确实现没有 bug ,并且拥有合适的贴图与动画显示。

- 1. (5%) 点击向日葵种子,能成功在场地上种下向日葵并显示,植物种子有冷却时间。
- 2. (5%) 向日葵可以产生阳光,场地中也会随时间规律产生随机掉落阳光,阳光可以被收集用于种植植物。
- 3. (5%) 场地右侧会生成僵尸,僵尸会按照一定速度朝左侧移动,僵尸到达屏幕左侧后游戏失败。
- 4. (10%) 僵尸会啃咬植物造成伤害,植物血量与啃咬伤害数值设置合理,植物血量归零后消失。
- 5. (5%) 可以种植豌豆射手,豌豆射手可以按照一定攻击速度发射豌豆。
- 6. (10%) 射出的豌豆将会对第一个碰到的僵尸造成伤害,碰撞后豌豆将消失。
- 7. (10%) 成功实现铲子功能,当拿起铲子后若第一次点击到的是植物,那么将会移除该植物。

需要注意的是,以上条目中实际上包含一些隐式的要求,比如游戏应该能稳定运行、在未点击过植物种子的情况下点击草坪不应该种下植物、种下植物之后再点击草坪应该什么都不会发生、除了向日葵之外其它植物不应该生产阳光(除非你特意实现了杂交功能)、铲子被拿起并铲掉了一个植物之后就应该被自动放下等等。如果你的程序在这些方面有错误,你的得分将是 unspecified 的。所以,在实现更多植物和僵尸之前,请务必先确保这些基本的功能都是正确的。

扩展功能实现 (30%)

- 1. (10%) 以下两种功能择一实现:
- 实现坚果墙,并且实现随血量变化的贴图与动画。
- 实现铁桶僵尸,实现铁桶僵尸随血量变化的贴图与动画。
- 2. (20%) 以下两种功能择一实现:
- 实现樱桃炸弹,实现爆炸动画并且移除炸毁的僵尸。
- 实现撑杆跳僵尸,实现撑杆跳功能以及不同阶段的僵尸行为表现及其贴图动画。

代码规范(20%)

此部分为扣分项,扣满 20% 为止。

- 1. (20%) 编译没有来自于你自己的代码的 warning 。
- 2. (5%) 代码中不应手动管理内存,而是应该正确使用智能指针和标准库容器。
 - o "正确使用智能指针"还包括**正确使用 std::enable_shared_from_this 相关设施**,例如,应该只在必要时调用 shared_from_this() ,而不是粗暴地将所有 this 都替换为 shared_from_this()
- 3. (5%) 将代码按类别分在不同的文件里。按类别把文件分在不同的子目录里,并为每个子目录都创建 CMakeLists.txt ,以 add subdirectory 的方式添加进项目。

- 4. (5%) 对于相似类的类似功能(例如,各种种子的 OnClick 、各种僵尸的 Update),不应复制代码,而是应该在基类里统一实现。
- 5. (5%) 正确使用枚举 (enum) 类型,并且应该使用限定作用域的枚举类型(即 enum class),除非有特殊原因(请解释)。不应使用神秘数字 (magic numbers) 或字符串来完成本该由枚举类型完成的功能。
- 6. (5%) 正确使用 utils.hpp 中定义的 const 变量,而不是使用神秘数字。
- 7. (5%) 通常情况下,所有数据成员几乎都属于"实现细节",应当设置为 private 。注意:
 - o 一些需要被子类访问的数据成员应该设置为 protected 。但是你不能滥用 protected ,因为 protected 成员的封装性实际上和 public 是一样弱的,详见《Effective C++》条款 22 。
 - o 的确存在一些足够简单的类,比如 std::pair ,它的数据成员(first 和 second)就是它的接口,此时直接将它们设为 public 即可,<u>无需编写那些 trivial 的 getters/setters</u> 。如果你的代码中的确存在这种情况,你需要给出解释。
- 8. (5%) 不判断对象的具体类型。
 - o "是不是植物"、"是不是僵尸"这样的大致判断是被允许的,但也不应通过 dynamic_cast 或 typeid 判断,而是应该借助一组虚函数,或者用一个成员变量来表示相关的信息(但是请注 意第 6 条和第 8 条)。
 - o 通过存储和访问 ImageID (图片的编号)来判断一个对象的真实类型也是不行的。
 - 提示:"豌豆"和"爆炸"虽然长相差异巨大,但是它们的层级相同,所做的事情具有一定相似性,是否可以归为同一基类?
- 9. (5%) 正确使用 static 成员函数、 const 成员函数和虚函数。
 - o 与 this 无关的成员函数应当是 static 的。
 - o 逻辑上不修改当前对象状态的 non-static 成员函数应当是 const 的。
 - 一些函数需要是虚函数,但不应将所有函数都设为虚函数。
- 10. (5%) 合理设计各个类之间的继承关系。
 - o 比如,让双发射手继承自单发射手可能不是一个好设计,就如同让正方形继承自长方形 (Lecture 22 及《Effective C++》条款 32)。你也许可以添加一个间接的类 Shooter ,让双发 射手和单发射手都继承自它,或者用模板类,让某些有区别的部分成为/依赖于模板参数等 等。
 - 一些特殊的功能,比如"植物僵尸"之类的"杂交"功能,可能涉及多重继承。你可以找 TA 交流交流。
- 11. (5%) 尽可能使用构造函数初始值列表。
- 12. (5%) 非必要不使用显式类型转换(用来构造临时对象的表达式 Type(args...) 或 Type{args...} 可以被允许)。如若需要,不使用 C 风格的类型转换运算符 (Type)expr ,而是使用 C++ 的有名字的类型转换运算符 what_cast<Type>(expr) 。
- 13. (5%) 变量应在即将使用时才被声明,特别是不应随意使用全局变量(常量除外)。

提交

将 assets, src 以及 include (如果有的话)和 CMakeLists.txt 打包为一个 .zip 文件提交到 OJ 。

我们会在 OJ 上设置一些编译时检测,以 error 或 warning 的方式给出反馈。这些反馈仅供参考,可能不准确,最终分数仍以 check 为准。

查重

特别说明:本题将查重,**而且会与去年的代码一起查重**。和信院其它代码课类似,抄袭被发现的后果**非 常严重**,并且抄袭者与被抄袭者是同等处罚的,请不要抱有侥幸心理。

抄袭来自网络或者 AI 工具的代码也是同等处理的。像"Copilot 莫名其妙帮我补出了和别人一样的代码"这种解释我们不会接受。请合理使用 AI 工具,特别是不要直接使用 AI 生成的代码。