QT作业报告

1. 程序功能介绍
2. 综述

本程序复刻了经典游戏黄金矿工，用持续转动的钩子瞄准并抓取物品，在限定时间内达到要求的分数即可过关。超出的分数可用于在下一关前的商店中购买道具。游戏中一共实现了6个关卡，每个关卡之间有商店。在游戏中按↓键释放钩子，按↑键使用炸药。

1. 道具介绍

（1）关卡内道具

* 炸药：一次性道具，炸掉已勾到物品。在关卡中勾上会使得炸药存量+1
* 时钟：延长关卡限制时间15秒
* 神奇药水：勾上后20秒内钩子速度加快/减慢
* 神秘包裹：随机包含金块、石头或其他道具
* 炸弹：勾中后立刻爆炸，炸掉周围物品

（2）商店售卖道具

* 炸药：炸药存量+1，价格1000
* 时钟：下一关延长关卡限制时间15秒，价格5000
* 神奇药水：下一关整关钩子速度略微加快，价格3500
* 目标减半：下一关目标分数和限制时间均减半，价格8000

1. 关卡介绍

* 小试牛刀(第1, 2关)：普通关卡，玩家可以借此熟练操作和道具
* 钻石大亨(第3关)：关卡内几乎全是钻石，积累分数的绝佳机会
* 乱石当道(第4, 5关)：动用手中的道具，清除挡道的石头，才能勾到下方的金块
* 争分夺秒(第6关)：开始只有15秒，关卡内全是时钟道具，惊险刺激

1. 项目设计细节
2. 物体

GameObject类及其子类与魔兽世界大作业中各种魔兽的实现方式类似。从一个父类GameObject开始，派生出gold，stone等各种子类。父类包含物体类型，大小，分数，钩子移动速度等信息，子类中附加特殊功能，在关卡中只要统一用父类指针的数组存储所有物体，利用多态简洁地遍历所有物体。

1. 关卡

Level类主要负责每个关卡的初始化，物品的初始随机生成与游戏中的销毁，关卡结束时的胜负判定。

由于写了各种物体生成的函数（在一定范围内生成位置、大小随机的物体，并且会判断是否重叠），因此在设计关卡时只需要提供道具种类与数量、限时、目标分数就可以方便快捷地制作新的关卡。

关卡中的计时器每1秒刷新一次，统一管理道具生效时间、动画时间，此外每1秒也会重新打印一次页面。计时结束后中止关卡，中止关卡函数运行，判断是否通关，准备进入后续界面。

此外，通过键盘控制钩子的事件写在关卡类中，将按键与钩子的状态联系。

1. 钩子（程序核心）

Hook类控制钩子的旋转、伸出、判断是否碰到物体以及收回。每50ms刷新一次钩子的状态（50ms可以理解为游戏的帧数为20帧）。

钩子的主要信息是位置坐标和角度。钩子开始在固定位置旋转（从0到π）,当按下↓键时就会伸出（保持当前角度按射线发射），在下行过程中通过两个圆是否相交的方式判定是否勾到物体，如果勾到物体或者碰到边缘就会返回。当返回到原处时，根据勾到物体的类型触发对应效果，并将物体删除。

值得注意的是，虽然将钩子显示在屏幕上时只能用整数坐标，但是内部存储钩子坐标时必须使用浮点数，否则会由于计算精度问题导致钩子下行与上行的轨迹不一致而无法回到原位。

1. 商店

Shop类负责商店初始商品的随机生成，商品的购买等操作。

开始商店会在指定范围内生成3个随机数，之后在对应位置产生对应的商品图片与描述。根据金币数量是否足够购买商品会设置对应的按钮是否可以按下（注意这个每次买完一个商品都要重新检测）。购买商品后，会有疯狂戴夫的音效和对话，金币数量会改变。

如何让购买的商品对下一关产生影响呢？目前的处理方式是在商店开始时直接产生下一关，购买物品后修改下一关的参数。当然应该也可以通过给下一关的构造函数传参的方式实现。

1. 小组分工情况

邓景文：关卡(Level)类、钩子(Hook)类的开发，物体(GameObject)类、商店(Shop)类的总体框架设计。游戏宣传视频剪辑。

赵浩男：物体(GameObject)类的开发，在关卡(Level)类中生成道具的实现，在钩子(Hook)类中钩到不同道具的效果开发，游戏图片与音效收集与ui设计。游戏实录解说的录制。

高骞：物体(GameObject)类部分道具设计，商店(Shop)类具体实现。路演PPT制作。

1. 项目总结与反思

通过本次 QT 大作业，本小组基本复刻还原出《黄金矿工》的游戏场景与玩法，并在原版游戏的基础上增加了关卡设计方面的创新，同时优化了商店界面的 UI 与呈现形式。

我们通过上手实践的方式加深了对面向对象编程的理解，并且相对熟练地运用类和对象的知识一点一点完成了我们所构想的各种功能。同时在道具消除、被炸毁以及力量药水生效时间的计时的功能开发中，我们也尝试使用了 STL 标准库中的 vector 和 deque 结构来实现。

同时，在实际开发过程中，我们也遇到并解决了一系列的问题。

首先是三人通过 Github 协作过程中，会经常遇到版本冲突问题，通过学习，我们掌握了回退版本以及 git stash 等常用 git 命令行进行版本控制，但是这对未来需要面对的团队协作来讲还远远不够，因此还需要进一步系统学习 git 的相关知识。

其次是 Windows 和 Mac 系统在共同开发 QT 项目时，会出现 Mac 不允许使用相对路径引用的问题，这就导致每一次不同系统的人在 pull 下来最新版本后，都需要再次将所有文件中出现的所有路径全部更改一遍，非常繁琐费力，经过搜索，我们目前也没有从网上找到合适的解决办法，这为我们的开发工作带来了极大的不便。

在使用 vector 和 deque 的过程中，在表达元素值自增的语句中，没有考虑到运算符优先级的问题，错误地使用了 \*it++，导致 debug 过程极其艰辛。也因此我们学会了在多种运算符一起使用过程中，可以多加几层括号来手动避免此类问题。

在 Level 类的开发过程中，我们还错误地使用了vector的erase函数，应该使用it=vector.erase(…)而非erase之后再it++。我们决定采用设置断点逐步检查的方式进行 debug，这让我们学会了打断点检查，对合作项目的 debug 更加熟练。

在添加 BGM 的过程中，我们通过搜索官方文档尝试使用过多种方法，但是全部报错失败了，后来发现我们仅仅关注了哪些库函数能够实现功能，但是忽略了版本是否支持的问题，我们使用的 QT6.5.0 版本移除了大部分有关多媒体的库函数，最终我们找到了该版本支持的库函数并成功实现了 BGM 的添加。