项目名称: 权力守卫战-Defence Of Thrones

项目概述

该项目是基于 C++和 Qt 框架开发的一款中世纪魔幻风格塔防游戏，玩家需要在地图上放置不同的防御塔，防止敌人到达目标。游戏需要有多个关卡和多种塔以及敌人类型，并提供建造, 升级塔; 攻击敌人; 选择关卡等功能。

系统架构

1. 游戏引擎与框架

- **开发语言**: C++
- **UI 框架**: Qt (Qt Widgets)
- **图形库**: QGraphicsView, QGraphicsScene 用于场景管理
- **声音库**: Qt Multimedia

2. 项目文件架构

功能需求

1 . 防御塔功能

- 玩家可以在特定位置花费金币放置防御塔。
- 玩家可以拆除已经放置的防御塔, 返还一定的金币
- 防御塔能够自动攻击进入其射程的敌人。(发射各种子弹)
- 防御塔在没有找到敌人攻击的时候, 会自动攻击地图内的障碍物
- 防御塔可以使用金币升级，提升攻击力和攻击范围。

2 . 敌人生成与路径规划

- 敌人会按照预设路径向萝卜移动。
- 敌人会从一个到两个出生点, 按一定时间间隔生成一定数量.
- 敌人有多种类型，具备不同的速度、血量和能力。(需要 boss 类型)

3 . 地图设计

- 每个关卡有独特的地图设计. 包括目标点, 可放置区域, 敌人出生点, 敌人路径和障碍物。
- 击碎地图内障碍物会获得金币, 每张地图的障碍物一经击碎就不会再生成.
- 玩家只能在特定的空位置上放置防御塔。

4 . 玩家金币与生命值

- 击败敌人或者障碍物会获得金币，用于购买和升级防御塔。
- 敌人到达目标会减少玩家的生命值，生命值为零时游戏结束。

5 . 关卡系统

- 游戏具有多个关卡，每个关卡敌人数量和强度逐渐增加。可以自由选择已解锁的关卡挑战.
- 玩家通关后进入下一关，失败可以重新挑战。
- 通过的关卡即解锁. 后续可以重复挑战.

6 . 游戏菜单与设置

- 主菜单包括开始游戏、选择关卡、设置和退出等功能。
- 设置界面允许玩家调整音效、难度等游戏参数。
- 进入游戏后允许暂停游戏和继续游戏, 暂停时会跳出暂停按键

游戏内容设计

主题

本游戏是中世纪战争主题塔防游戏, 并融入了大量魔幻元素. 玩法上借鉴保卫萝卜, 在主题上借鉴了美剧《权力的游戏》(原著小说: 冰与火之歌). 玩家将在游戏里, 使用权游中的一些防御器具, 史塔克家族的角色, 以及龙妈和她的军队, 来抵抗来自兰尼斯特家族和夜王大军的攻击.

防御塔与投掷物

- **弓箭兵 (Archer):** 射出弓箭; 特殊升级是射出魔法火箭 (带火的箭矢), 可以点燃敌人持续掉血, 对于异鬼军团单位伤害更高;
- **投石车 (StoneThrower):** 投掷伤害更高, 给敌人减速效果的石头; 特殊升级是一次投出三块石头;

- **琼恩·雪诺 (JohnSnow):** 近战平 A 面前的敌人, 同时给周围的己方单位提升伤害; 特殊升级是每四下攻击召唤一只小狼去攻击敌人 (小狼视作投掷物)
- **卓耿 (Dragon):** 喷出具有高额伤害的魔法龙焰弹, 一次可以伤害两个单位, 可以点燃敌方单位, 龙焰(弹)对于异鬼军团伤害更高; 特殊升级是投掷物转化成魔法龙焰, 在地图上成一条直线, 具有穿透地形与敌方单位的能力.

敌人

- **异鬼士兵 (DeadAlive):** 异鬼军团单位, 基础敌方单位, 给目标带来一点伤害. 出现在第一, 二关中.
- **野人 (Wilder):** 人类单位, 血量较多, 给目标带来一点伤害. 出现在第一关
- **红女巫-梅丽珊卓 (Melisandre):** 人类单位, 移速较快, 可复活自己一次, 给目标带来一点伤害. 出现在全部三关中.
- **韦塞里昂 (Vesalion):** 异鬼军团单位, 血量较多, 移速很快, 空中单位, 无法被琼恩雪诺攻击到. 它是被夜王杀死并复活的龙, 可以喷出蓝色火焰, 所以对目标可以带来五点伤害. 出现在第二关.
- **夜王(NightKing):** 异鬼军团单位, 异鬼军团的王, 第二关的 BOSS. 血量非常丰厚, 给目标带来三点伤害. 可以每八秒在自己的位置召唤一只异鬼士兵.
- **兰尼斯特士兵 (LannisterSoldier):** 人类单位, 基础单位, 带来一点伤害. 出现在第三关.
- **葛雷乔伊士兵 (GreyjoySoldier):** 人类单位, 移速较快. 出现在第三关.
- **魔山 (Mountain):** 人类单位, 血量非常非常丰厚, 给目标带来五点伤害. 出现在第三关.
- **弑君者-詹姆·兰尼斯特 (KingSlayer):** 人类单位, 血量丰厚, 移速很快, 可给目标带来三点伤害. 第三关的 BOSS.

地图, 关卡与障碍物

- **第一关:** 绝境长城 (TheWall). 守卫目标是黑城堡 (CastleBlack), 敌人出生点是一颗巨树 (GiantTree)(被占领的生命之树). 障碍物类型是树木(Tree), 巨石(Stone), 冰屋 (IceBurg). 背景图片选用权游绝境长城的图片.
- **第二关:** 临冬城 (WinterFall). 守卫目标是临冬城 (WinterFall), 敌人出生点是黑城堡 (CastleBlack). 障碍物类型是冰封湖泊(Lake), 马厩(Stable), 木屋 (Cabin). 背景图片选择临冬城的内景图.
- **第三关:** 君临城 (KingsLanding). 守卫目标是红堡 (RedKeep), 敌人出生点是贝勒大教堂 (Baelor). 障碍物类型是楼房 (Buildings)(建议设计两种不同贴图的楼房, 其他特性完全相同, 以实现外观的优美), 城墙与城门 (Walls), 钟楼 (BellBuilding). 背景图片选择君临城的图片.

艺术与音乐风格

- **艺术风格**：采用暗黑奇幻风格，建筑和单位设计应反映出《权力的游戏》的美学。
- **配乐与音效**：使用原作的音乐元素或类似风格的配乐，增加沉浸感。

类设计

1. GameController

- **继承自:** `QWidget`
- **功能:** 管理游戏资源。显示主菜单界面、显示关卡选择界面、加载关卡、暂停/恢复游戏、退出游戏、控制玩家生命与金币。
- **主要成员变量:**
  - `currentLevel`: 当前关卡
  - `hardlevel`: 游戏难度
  - `volumn`: 游戏音量等级
  - `gameBgm`: 游戏内背景音乐
  - `mainMenu`: 主菜单界面
  - `levelSelectMenu`: 关卡选择界面
  - `gameScene`: 游戏场景界面
  - `player`: 玩家
  - `map`: 游戏地图
- **主要方法:**
  - `addEnemy(Enemy*)`: 以一定的频率创建新的敌人. 调用 gameScene 的 addEnemy 加入场景中.
  - `addProjectile(...)`: 类似上面 *(注意: 在 gameController 中不需要存这些对象, 存放在 gameScene 中, 但是在 GameController 中创建)*
  - `loadMap(int level)`: 加载地图, 需要调用地图的加载关卡功能, 并且场景内绘制地图
- **槽:**
  - `startGame()`: 开始游戏
  - `endGame()`: 结束游戏
  - `exitGame()`: 退出游戏
  - `showMainMenu()`: 显示主菜单
  - `showSettingMenu()`: 显示设置页面
  - `showLevelSelectMenu()`: 显示关卡选择菜单
  - `loadLevel(int level)`: 加载指定关卡
  - `pauseGame()`: 暂停游戏
  - `resumeGame()`: 恢复游戏

- `addTower(Tower*,int towerType)`: 创建新的 tower, 花费价格, 调用 gameScene 的 addTower 加入场景中
- `updateTower(int money)`
- `deleteTower(int money)`
- `killEnemy(int money)`
- `onEnemyArrived(int damage)`
- `damageObstacle(int money)`
- `addObstacles(int x,int y,int type)`

2. MainMenu

- **继承自:** `QWidget`
- **功能:** 显示主菜单界面，提供开始游戏、选择关卡、设置和退出等功能。
- **主要成员变量:**
  - `startButton`: 开始游戏按钮
  - `levelSelectButton`: 选择关卡按钮
  - `settingsButton`: 设置按钮
  - `exitButton`: 退出按钮
- **主要方法:**
  - `show()`: 显示主菜单
  - `hide()`: 隐藏主菜单
- **信号:**
  - `startNewGame`: 开始一局新游戏
  - `openLevelMenu`: 打开菜单界面
  - `openSettingMenu`: 打开设置页面
  - `exitGame`: 退出游戏
- **槽:**
  - `onStartButtonClicked`: 用户点击了开始游戏按键
  - `onLevelSelectButtonClicked`: 类似上面
  - `onSettingButtonClicked`
  - `onExitButtonClicked`

3. LevelSelectMenu

- **继承自:** `QWidget`

- **功能:** 显示关卡选择界面，允许玩家选择不同的关卡。注意未解锁的关卡不可进入, 在 UI 上要显示锁起来, 用户点击就没有反应, 不能返回对应的 level 值.
- **主要成员变量:**
  - `levelButtons`: 关卡按钮列表
- **主要方法:**
  - `show()`: 显示关卡选择菜单
  - `hide()`: 隐藏关卡选择菜单
- **信号:**
  - `selectLevel(int level)`: 发送关卡信息给 gameController
- **槽:**
  - `onLevelButtonClicked(int level)`: 用户点击已解锁的关卡按钮

4. GameScene

- **继承自:** `QGraphicsView`
- **功能:** 显示游戏场景, 管理游戏中的所有图形元素, 处理鼠标事件。存放 towers, enemies, projectiles.
- **主要成员变量:**
  - `pauseGameButton`: 暂停按钮
  - `resumeGameBUtton`: 继续按钮
    - `scene`: `QGraphicsScene` 对象，管理游戏中的图形元素
    - `livesTextItem`: `QGraphicsTextItem` 对象, 用于显示血量文本便于更新
    - `towers`: 防御塔列表
    - `enemies`: 敌人列表
    - `projectiles`: 投射物列表 *(建议使用对象池实现)*
    - `obstacles`: 障碍物列表
    - `towerSelectMenu` **和** `towerUpdateMenu` **两个菜单, 选择里面的按钮会触发对应的信号**
- **主要方法:**
  - `addTower(Tower*)`: 添加防御塔
  - `addEnemy(Enemy* enemy)`: 添加敌人
  - `addProjectile(Projectile* projectile)`: 添加投射物
  - `addObstacles(Obstacles* obs)`
  - `updateScene()`: 更新场景
  - `pauseScene()`: 暂停游戏 (实现可以调用每一个 tower 和 enemy 的暂停函数)
  - `mousePressEvent`: 重写的鼠标事件, 用于捕捉用户鼠标点击位置. 主要用在放置新的防御塔和点击防御塔进行拆除或者升级上.

- **信号:**
  - `toPauseGame()`: 发出信号给 GameController
  - `toResumeGame()`
  - `towerSelected(int type)`: 选定了为 type 的塔
  - `updateSelected(int money)`: 选定了价格为 money 的升级
  - `toDeleteTower(int money)`
- **槽:**
  - `onPauseButtonClicked()`
  - `onResumeButtonClicked()`
  - `onTowerSelectButtonClicked(int)`
  - `onUpdateSelectButtonClicked(int)`
  - `onDeleteTowerButtonClicked`
  - `updatePlayerLives(int lives)`: 更新玩家血量的标签 *(它与 player 的血量改变信号相连接, 连接要在 GameController 中进行)*

5. SettingsMenu

- **继承自:** `QWidget`
- **功能:** 显示设置界面，允许玩家调整游戏参数。
- **主要成员变量:**
  - `volumeSlider`: 音量调节滑块
  - `difficultyComboBox`: 难度选择下拉框
- **主要方法:**
  - `show()`: 显示设置菜单
  - `hide()`: 隐藏设置菜单
  - `applySettings()`: 应用设置
- **信号:**
  - `volumeChanged(int volume)`: 发出调整音量的信号
  - `DifficultyComboBoxChanged(int index)`: 发出挑战游戏难度的信号
  - `gameBgmChanged(int index)`: 发出修改 `gameBgm` 的信号
- **槽:**
  - `onVolumeSliderChanged(int value)`: 检测用户改变音量的操作
  - `onDifficultyComboBoxChanged(int index)`: 检测用户改变难度的操作
  - `onGameBgmChanged(int index)`: 检测用户改变 bgm 的操作

6. Player

- **功能:** 定义玩家的状态。**注意这里面的东西只能在 gameController 中访问.**
- **主要成员变量:**
  - `money`: 玩家拥有的金币数
  - `lives`: 玩家剩余的生命数
- **主要方法:**
  - `spendMoney(int amount)`: 消耗金币
  - `earnMoney(int amount)`: 获得金币
  - `loseLife()`: 玩家失去生命 (顺便判断玩家是否失去所有生命)
- **信号:**
  - `lifeChanged(int lives)`: 血量改变时触发
  - `gameOver()`: 本局游戏失败, 自动重启.

7. Tower (防御塔)

- **功能:** 定义不同的防御塔类型。以它作为父类可以生成至少三种子类防御塔. 所以有些函数可以是虚函数, 是 protected.
- **主要成员变量:**
  - `projType`: 投掷物种类
  - `range`: 攻击范围
  - `attackSpeed`: 攻击速度
  - `buyCost`: 购买的花费
  - `sellPrice`: 拆除返回的价格
  - `upgradeCost_1`: 第一类升级花费 (提升攻击范围与速度)
  - `upgradeCost_2`: 第二类升级花费 (觉醒特殊能力, 比如说减速, 群伤, 一次性投出两个/对投掷物升级)
- **主要方法:**
  - `Tower(QPoint pos_)`
  - `attack(enemy* target)`: 攻击敌人
  - `upgrade(int type)`: 升级塔的属性, type 指定升级的类型
  - `sell()`: 出售塔
  - `getBuyCost()`: 返回 buyCost
  - `getSellPrice()`
  - `getUpC1`
  - `getUpC2`

8. Enemy (敌人)

- **功能:** 定义敌人的属性和行为。可以定义子类, 有不同的血量, 速度和伤害与外形.
- **主要成员变量:**
  - `health`: 生命值
  - `speed`: 移动速度
  - `damage`: 对萝卜的伤害
  - `reward`: 击败后给玩家的奖励（金币）
  - `routine`: 敌人的移动路径 (可以使用一个由 tuple 组成的数组来存放)
- **主要方法:**
  - `Enemy(QPoint pos_,vector<<int>,<int>>)` 构造函数要传入初位置与路径
    - `move()`: 移动到下一个位置
    - `takeDamage(int damage)`: 接受伤害, 检查是否死亡, 死亡则触发信号
- **信号:**
  - `isDead(int reward)`: 在被打死以后触发; 如果是到达终点, 不触发
  - `isArrived(int damage)`: 检查敌人是否到终点

9. Projectile (投掷物)

- **功能:** 定义防御塔发射的子弹或魔法攻击。
- **主要成员变量:**
  - `speed`: 移动速度
  - `damage`: 伤害
  - `target`: 目标, 它移动的逻辑可以是朝着目标走去
- **主要方法:**
  - `move()`: 更新投射物的位置
  - `hitTarget()`: 命中敌人时触发的效果

10. Map (地图)

- **功能:** 定义关卡的地图布局. 关卡与地图设计(包括敌人路线, 萝卜位置和贴图等都在这里完成)
- **主要成员变量:**
  - `grid`: 二维数组表示地图的网格（包含障碍物, 道路和可放置位置）
  - `enemyPath`: 敌人道路
- **主要方法:**
  - `loadMap(int level)`: 根据关卡载入地图
  - `drawMap(scene*)`: 在 scene 上绘制地图. 这个函数会在 gameScene upgrade 时被调用; *(注意, 在绘制时, 同时将地图内的所有障碍物通过信号传给 GameController 并加入*

scene)
    - `isPlaceAble(QPoint pos_)`: 检查某个位置是否可放置方块
    - `getPath()`: 返回敌人可以走的道路
    - `getSpawnPoints()`: 返回敌人生成的起始位置列表
- **槽:**
    - `obstaclePos(QPoint pos_,int type)`: 返回各个障碍物的位置与类型

11.Obstacle(障碍物)

- **功能:** 固定不动, 被击碎可以给予玩家金币. 不会复活. 实现上可以使用 type 变量表示不同的障碍物, 不需要定义各种子类.
- **主要成员变量:**
    - `price`: 被击碎后可以给予玩家的金币
    - `health`: 生命值
- **主要方法:**
    - `checkIfDead()`
- **信号:**
    - `isDamaged(int price)`: 障碍物被破坏时发出此信号, 给予玩家金币

类之间的关系

- **注意: 本项目的位置除了网格地图外, 均是实际位置! 地图的位置在绘制时也需要转成实际位置, 这样便于碰撞检测和对象管理. 需要求对象位置时, 直接用 pos() 和 setPos() 方法即可, 不用特地存放 pos 的信息. 所以所有的实体都得是 QGraphicsItem 的子类.**
- `GameController` 负责管理游戏的整体流程，包括显示主菜单、关卡选择菜单和游戏场景。也包括对于 `Player` 信息的获取与修改
- `MainMenu` 和 `LevelSelectMenu` 通过信号和槽与 `GameController` 交互，通知 `GameController` 用户的选择。
- `GameScene` 负责显示游戏中的所有图形元素，并与 `GameController` 交互以更新游戏状态。
- `SettingsMenu` 允许玩家调整游戏参数，并与 `GameController` 交互以应用设置。
- `Tower` 可以在 `GameScene` 的鼠标事件下完成放置, 升级与出售防御塔, 并通过信号与槽, 在 `GameController` 中对玩家金币信息进行修改. 它会通过 attack 方法生成 projectile 攻击敌人.
- `Enemy` 会按照 `Map` 给出的路线移动, 被攻击到会扣除血量. 达到终点则扣除玩家血量, 自己也消亡. 如果血量被扣除完, 则会通过信号告诉 `GameController` 给玩家增加钱, 自己消亡.

- `Projectile` 由 tower 生成, 会跟踪敌人位置攻击敌人. 发生碰撞后会扣除敌人的血量. 具体怎么实现, 可以考虑 qt 自带的碰撞判定以及信号和槽.
- `Map` 给出二维数组表示的地图, 结合关卡数字进行关卡设计; 包括路线, 空闲方块, 敌人出生点, 萝卜位置, 障碍物初始位置. 需要将障碍物初始位置传递出来, 需要将敌人的出生点与路线传递出来. 通过 `GameController`, 在构造 `Enemy` 时就通过构造函数传进去.
- `Obstacles` 是障碍物, 被破坏时会通过信号告诉 `GameController`, 增加金币.

开发计划

第 1 阶段: 核心游戏玩法开发

- 完成地图、敌人和塔的交互逻辑。
- 实现敌人路径规划与防御塔攻击功能。
- 实现基本的几个页面与功能

DDL: 10.13

第 2 阶段: UI 与关卡设计

- 设计并实现多个关卡和地图。
- 加入难度系统

DDL: 10.17

第 3 阶段: 完善与测试

- 添加音效和游戏效果。
- 美化 UI 与贴图
- 修复 bug，并进行优化。
- 丰富功能, 如加入玩家得分系统, 加入游戏得分纪录系统; 加入炮台"集火"功能

第 4 阶段: 发布与优化

- 完成最终版本，优化性能，发布游戏。
- 拍摄演示视频.

DDL: 10.21

分工

1. 毛绎然: 项目总体设计, 代码整合, 流程管理, 代码 Debug 与优化; GameController部分开发; UI 与游戏内容设计.
2. 滕燎原: MainMenu, LevelSelectMenu, SettingsMenu, Map 开发; UI 设计.
3. 陈旺: Enemy, Obstacle, Player 及其子类开发.
4. 苏敬茗: GameScene 开发; 关卡设计与难度系统设计.
5. 梁大锴 Tower, Projectile 及其子类开发; 游戏图片素材的搜寻.

注意, 在每个类与每个函数的前面 (或者函数头下方) 要写上注释, 说明此函数的功能, 几个主要变量的含义. 代码符合规范, 起名统一采用驼峰命名法, 遵循规范, tab 指定为 4 个空格.