坦克动荡魔改版作业报告

gw说的对不　队

张子沐　任致远　张科宇

本项目是对经典小游戏“坦克动荡”的魔改与加强，不仅基本复刻了坦克动荡的游戏设定和规则，而且对原游戏的细节做出了升级和加强。

一、小组成员分工情况

（一）张科宇同学的负责内容

1. 游戏初始界面的布局。包括游戏背景音乐的打开/关闭按钮，帮助、设置、进入游戏、退出游戏等按钮的鼠标悬停动画及音效，程序的标题、图标等设计。

2. 游戏玩法说明界面。包括当玩家未确认时，“Q”键和“M”键的动画，以及当玩家按下“Q”和“M”后图案颜色的改变。

3. 游戏界面中的音效。包括坦克发射子弹的音效、子弹和墙壁碰撞从而反弹的音效、坦克被子弹击中后爆炸的音效。

（二）张子沐、任致远同学的负责内容

负责游戏主体部分的实现。包括：

1. 每局游戏开始时，地图中墙壁单元块的布局，坦克初始时位置和姿态的随机摆放，以及游戏沙漠背景的加载。
2. 用键盘控制两个坦克进行前进、后退、左转、右转、发射炮弹。
3. 坦克和墙壁摩擦行进时的物理仿真效果。
4. 每辆坦克一次性最多可以发射5颗子弹、每颗子弹在地图上最多存在10秒钟的游戏设定。
5. 子弹与墙壁进行碰撞后的反弹效果。
6. 子弹与坦克碰撞后，坦克的爆炸动画。
7. 一局游戏结束后，对胜利玩家进行计分。

二、程序功能介绍

（一）游戏初始界面

游戏初始界面包含三个主按钮。

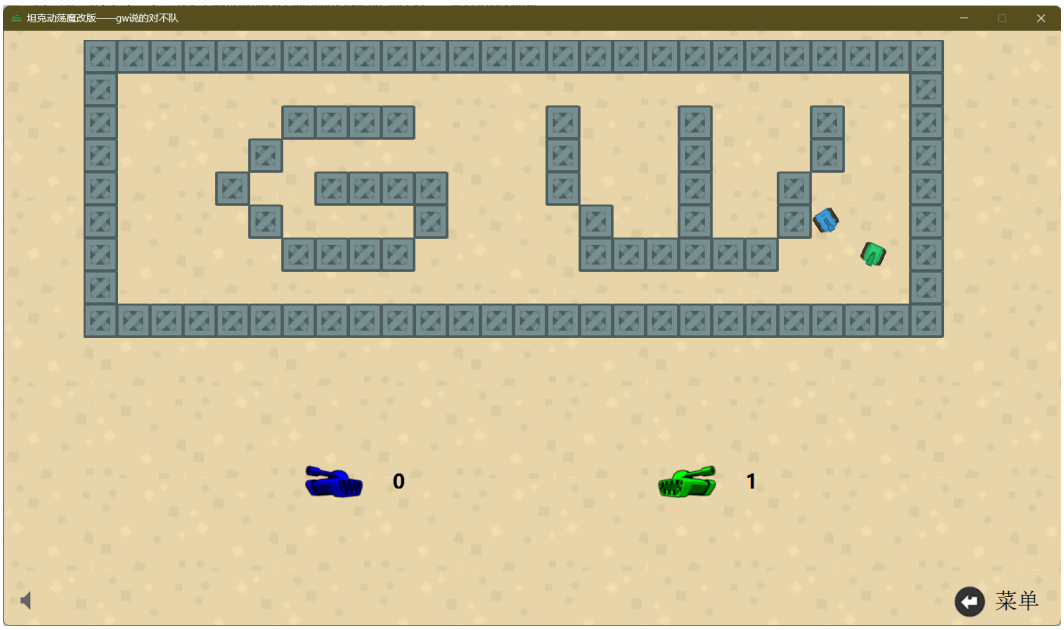
单击帮助按钮会弹出有关游戏操作方式说明的对话框。

单击设置按钮会进入到设置界面。玩家在该页面可以调节游戏背景音乐的音量大小和游戏音效的音量大小。

单击双人对战按钮，玩家会进入到准备阶段的界面。该界面提示了两位玩家的游戏操作方式：两位玩家分别使用“ESDF”键和方向键控制坦克的移动，同时跳动的“Q”和“M”提示了玩家发射子弹的方式。当两位玩家均准备好之后，分别按下Q和M键，灰色的键盘图片会变为彩色，代表玩家已做好准备。

（二）游戏界面

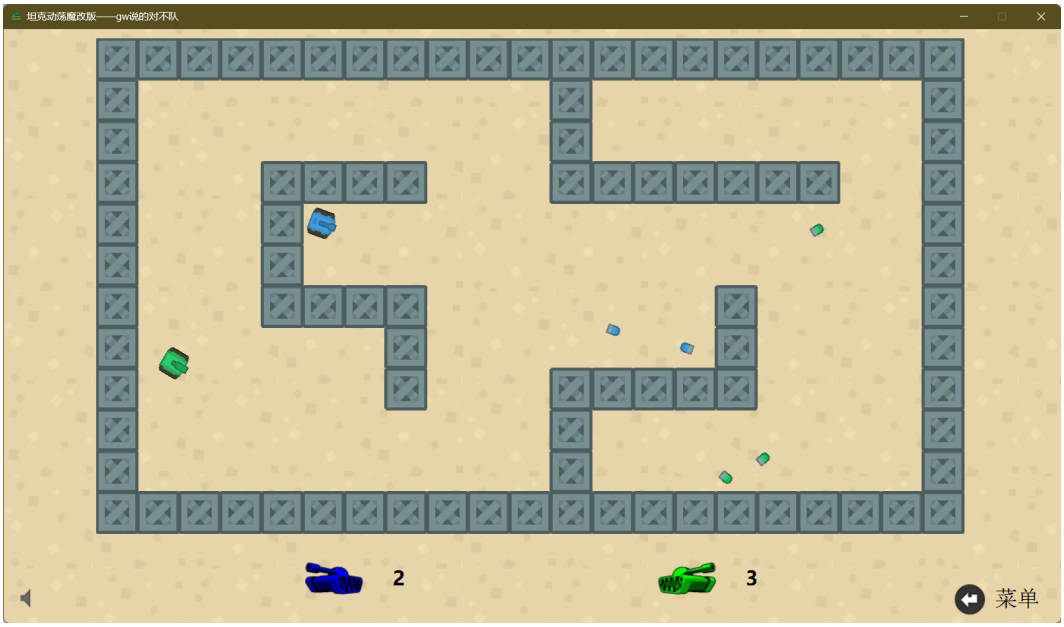
为了增加游戏的趣味性和不确定性，每局游戏开始之前，游戏的地图都会进行刷新；坦克的初始位置和朝向则会随机确定。



玩家可以通过键盘操作控制坦克的前进、后退、转向以及发射子弹，绿方坦克发射绿色子弹，蓝方坦克发射蓝色子弹。坦克和子弹的运动速度会随着地图的大小进行等比例的调整。

当坦克和墙壁发生接触后，坦克的运动速度会发生减速，减速的程度会根据坦克运动方向和墙壁的夹角进行变化。当坦克运动方向垂直于墙壁时，坦克的运动速度减为0；反之，当坦克运动方向和墙壁的夹角很小时，坦克的速度几乎不发生变化，从而给这款游戏带来了真实的物理仿真体验。

子弹与墙壁碰撞后会发生类似于光线的"镜面反射"，因此一方坦克发射的子弹也是极为可能打中自己的，这为游戏增添了极大的乐趣。一方坦克一次性只能够发射五枚子弹，子弹在飞行10秒钟后会消失，这时坦克又可以重新发射子弹。所以，玩家在操控坦克时，要合理控制每次发射子弹的颗数，防止自己陷入到短时间没子弹的被动局面之中。



当子弹与坦克碰撞后，坦克会触发爆炸动画。当一方坦克死亡之后，本局比赛还会进行一小段时间，如果在这段时间内，另一辆坦克也不幸爆炸，则本局计为平局，双方均不得分；反之，若有一方坦克活了下来，则该坦克获得本局游戏的胜利，分数加一。

游戏下方记录了两位玩家的总得分。

坦克发射、子弹与墙壁的碰撞、坦克爆炸均会触发相应的音效。玩家可以通过窗口下方的声音按钮控制游戏声音的打开关闭，单击菜单按钮可以返回主界面。

三、项目各模块与类设计细节

本节重点介绍游戏主体部分的设计细节。通过合理运用 `QGraphicsItem` 、 `QGraphicsScene` 、 `QGraphicsView` 等工具，对游戏场景和各个游戏元素进行统一管理。

（一） `GameScene` 类

`GameScene` 是游戏的场景控制类，继承于Qt中 `QGraphicsScene` 类。

在 `GameScene` 的构造函数中，通过C++的文件操作读入地图文件，根据地图文件的信息绘制墙壁单元格，并根据地图信息等比例调整墙壁单元格的大小。除此之外，构造函数还将两辆坦克添加到场景中，并完成了坦克初始位置、姿势的随机确定。

`GameScene` 还有两个多态成员函数 `keyPressEvent` 与 `keyReleaseEvent` ，完成了程序与键盘操作的交互。通过不断地监测键盘按键的状态，达到实时改变两个坦克运动状态的目的。

（二） `GameItem` 类

`GameItem` 类继承于Qt中的 `QGraphicsItem` 类，是本游戏中所有游戏元素的父类，完成了对于游戏的元素的基本操作，包括素材图片的加载，将元素对象添加到游戏场景中等基本操作，体现了高级程序设计中的多态继承思想。

（三） `Tank` 类

`Tank` 类继承于 `GameItem` 类，同时又是 `Bluetank` 类和 `Redtank` 类的父类。

在 `Tank` 类中的 `advance` 函数是控制坦克移动的重点函数。该函数是 `QGraphicsItem` 类的虚函数，当帧率定时器发出信号时， `advance` 作为一个槽函数接收到该信号，并完成对于游戏的刷新。在 `Tank` 类中， `advance` 函数根据已经记录下来的键盘按键状态，确定坦克的运动方向并根据运动方形重新绘制该帧的坦克图片，这样就完成了坦克运动动画的实现。

在 `advance` 函数中，通过 `QGraphicsItem` 类的 `collidingItems` 方法来监测坦克的碰撞事件。如果坦克和墙壁发生的接触，则根据坦克和墙壁的连线、坦克运动方向等信息确定出坦克与墙壁的夹角，用夹角的余弦值乘摩擦系数即可确定坦克减少的速度大小；如果坦克和子弹发生碰撞，则会通过Qt的信号与槽机制发出死亡信号， `QGraphicScene` 中的槽函数接收到信号之后就会触发爆炸动画函数。

（四）`Bullet`类

`Bullet`类继承于`GameItem`类，同时又是`Bluebullet`和`Redbullet`类的父类。

与`Tank`类类似，`Bullet`类的`advance`方法实现了子弹的运动，通过`collidingItems`方法检测与墙壁是否发生碰撞，如果碰撞，则根据角度实现“镜面反射“。

在具体的实现过程中，由于当坦克发射子弹的一瞬间，坦克距离子弹会非常近，容易触发坦克和子弹的碰撞事件。为了修复这个漏洞，在子弹被发射的一瞬间，设置一个较短的计时器，在该计时器计时期间忽略和坦克的碰撞事件，计时结束之后再开始检测是否与坦克发生碰撞，从而有效的规避了游戏Bug。

（五）其他模块与类

`MyConstants.h`头文件保存了本项目所需要的所有常量，可以被所有的cpp文件访问。

`set`、`Widget`等类完成了其他游戏窗口的构建，包括各种按钮的功能实现、鼠标悬停时的按钮动画、游戏背景音乐和各种音效的播放与关闭等。

四、项目总结与反思

（一）界面设计

相比经典的坦克动荡游戏，我们虽然采用了更为全新的游戏素材，但是整体做出来的效果不够自然简约，仍然还有较大的进步空间。

（二）小组合作

由于对Github的运用不够熟练，缺乏对版本控制工具的学习，因此小组经常采用U盘进行源代码的移动，合作效率较为低下。

（三）代码架构

小组成员还没有接受过工程编写的相关训练，缺乏写中小型项目的相关经验，因此对于头文件的引用、不同类的定义位置等等基础问题较为生疏，也造成了许多不必要的麻烦。

最后，非常感谢程序设计实习这门课程给我们提供了这样一个接触工程项目编写的机会。俗话说，“一回生，二回熟”，这次“零的突破”带给了我们许多在课堂之上学不到的知识和经验，希望在未来的学习道路中，我们能够接触到更多真实项目工程的编写机会，增长代码能力，打下坚实的程序编写基础。