逻辑设计：

船只的指令集如下：  
 1.stop，当前回合不进行任何操作

2.moveto(x,y,dr),船只移动到(x,y)并使方向增加dr

3.shoot，向指定位置发射炮弹

每一回合，对每一条船逐个执行其设定的指令，并计算出下一回合的预设指令

计算的方式如下：

1. 结算炮弹伤害，若我方死亡，设定死亡状态，指令设定stop
2. 若我方没有死亡，当我方没有自动攻击目标：
3. 攻击范围内有敌方船只，设定自动攻击目标，指令设定stop
4. 我方预设移动路径为空，设定stop
5. 我方预设路径不为空，抽取路径的第一个元素设定moveto
6. 若我方没有死亡，当我方有自动攻击目标：
7. 自动攻击目标在我方射程之外，向其所在位置寻路设定moveto
8. 自动攻击目标在我方射程之内
9. 计算两者位置向量与我方方向向量的点积
10. 若攻击目标在射击角度之内
11. 我方无攻击间隔冷却时间，设定shoot操作
12. 我方处于攻击间隔时间之中，设定stop操作
13. 攻击目标在射击角度之外，计算两向量的叉积以调整角度

对每一艘船只，设定两种攻击方式：侧舷炮击和船首炮击，其示意图如下：

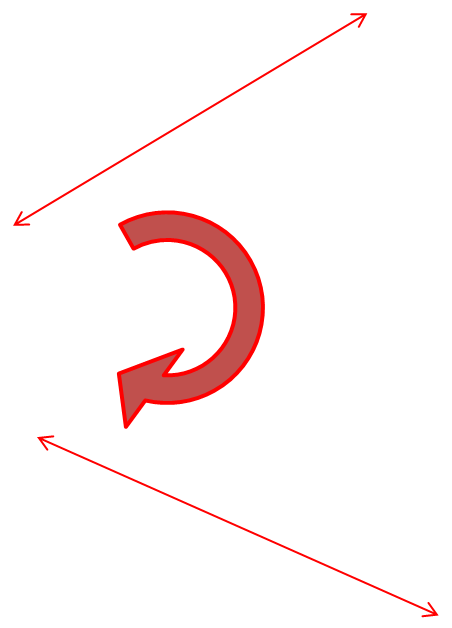
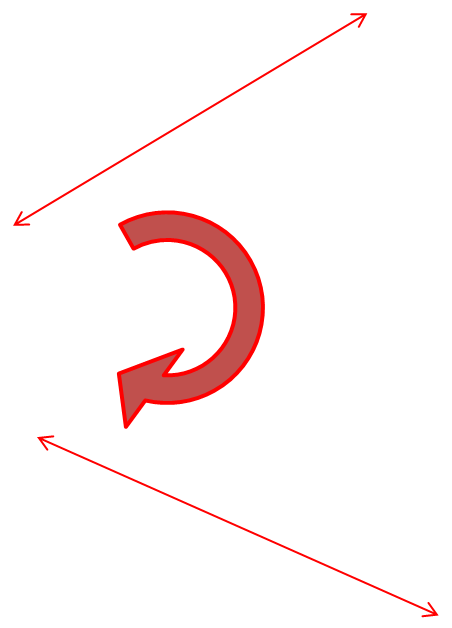


图1 侧舷炮击 图2 船首炮击

我对游戏窗体和船只所在的div层都设定了监听。

对窗体的监听中，若捕捉到左键，当前选中单位合法时显示选中图片和单位信息；若捕捉到右键，当前选中单位为我方船只则对其进行寻路，设定moveto操作并清空自动攻击目标。

对船只的监听中，若捕捉到左键，则设定当前的选中单位；若捕捉到右键，若当前选中单位不为我方单位则返回，否则播放音效。若船只为敌军船只，设定当前选中单位的自动攻击目标为此船只并阻止监听事件冒泡；若为我方船只则不处理，监听事件冒泡被窗体监听接受进行寻路。

寻路算法采用A\*算法，使用heap代替queue储存所有扩展节点并利用hash表去除重复状态。估价函数设计为：当前点与目标点的曼哈顿距离/单位的移动速度。由于逻辑设定中有大量的寻路操作，一次寻路的结果被利用到的次数较少，若每次搜出完整路径才结束将导致大量的计算资源浪费，更为严重的是大量寻路操作的堆积将使得游戏出现大量卡顿，卡死，所以每次寻路并不一定需要给出精确路径，给出大致方向即可。故我将单次寻路的搜索扩展次数设定为(每帧时间) \* 50次，目前的表现结果差强人意。

由于船只图片是img标签直接堆叠的，所以船只重叠时一方面影响美观，另一方面影响监听，我们必须防止这种重叠，因此对船只设定了碰撞体积。在寻路操作中，以其他存活船只中心点为圆心，半径30的范围内均不可通过。但由于障碍物是移动的，直接使用以前的预设路径可能导致单位碰撞，所以我们在每回合结算指令之前增加了碰撞处理：

1. 提取所有有实际位移的船只，预先设定他们是“可移动的”
2. 对可移动的船只尝试移动，若产生碰撞：
3. 设定A不移动，若B的移动仍然产生碰撞，则B不可移动
4. 设定B不移动，若A的移动仍然产生碰撞，则A不可移动
5. 对所有可移动船只进行移动操作
6. 对不可移动的船只重新寻路，目标点为预设路径的最后一个元素

这个处理机制是基于每个回合都没有位置冲撞的，这个机制对单回合的最坏影响是所单位均不移动，但这仍然保证任意两条船没有碰撞冲突

最后我们添加了一个简单的AI,每一个回合每一艘船选择最近的我方船只为攻击目标，当攻击目标位于射程之内且处于攻击间隔时，他会尝试保持当前距离移动到圆上的另一点执行攻击操作。不过由于这个选择很容易使AI卡在地图边缘，我将其和攻击目标的位置向量取反之后做相同运算，使得AI现在将不断尝试贴在我方舰队周围进行攻击。