FreeWay 游戏实验报告

李佳鑫（211220006、211220006@smail.nju.edu.cn）

1.代码阅读和策略模型分析

1.1 策略模型分析：

策略模型执行方式：模型通过调用 act 来执行行动，在 act 中调用了 learnPolicy 函数，通过传入当前状态和一个新创建的 WinScoreHeuristic 来进行强化学习。在 learnPolicy 函数中进行了共十次迭代。在每次迭代中调用 simulate 函数进行模拟，对并对 dataset 进行更新，最终用训练该 dataset 得到学习结果。Simulate 学习方式如下：在不超过模拟深度的限制的情况下，对当前状态进行特征提取，并对当前局面用启发式函数进行分数评估，通过 policy 选取一个行动并更新，计算更新后的分数评估，将模拟的局面计算出 delta_score,并更新 factor，将特征，评分，以及行动记录下来。之后对累计的 Q 值进行计算，并将模拟结果返回。根据 policy 选取行动的规则如下，有 epsilon 概率随机选取一个行动，有 1-epsilon 的概率根据特征选取 Q 值最大的行动。

该执行方式的缺点：

m_epsilon 的不变性，在模拟初期，不确定性较大，但随着深度加深，经验的累积，从 Q 值表中能提取到较优的行动的几率则更大。

改进方法：让 m_epsilon 在初始时设置为一个较大的值，随着 Q 值表的扩充 m_epsilon 可以适当降低。

1.2 SIMULATION_DEPTH, m_gamma, m_maxPoolSize，三个变量分别用于限制模拟深度，这里的为 20，在模拟的每次深度加深中 factor 都要乘以 m_gamma，当 m_gamma 越小则模拟时层数较深的局面对特征的评分影响越小，反之则越大，m_maxPoolSize 对 m_dataset 的大小进行限制。

1.3 getAction：有 epsilon 概率随机选取一个行动，有 1-epsilon 的概率根据特征选取 Q 值最大的行动。getActionNoExplore：则直接选取 Q 值的最大行动。最大的区别是 getAction 有 explore 的过程，为了不让 agent 限于局部最优，增加随机性，让 agent 有更高的几率选取全局最优的行动。getAction 用于在模拟中寻找最优行动，getActionNoExplore 则 act 中使用获取最佳行动。两者是 Exploration(探索)和 Exploitation(利用)的关系。

2.特征提取方式修改：

当前游戏的训练方法不算有效，npc 通常只能在第一二层运动，难以到达最上层，要完成游戏就不能只呆在下层，所以要减小与目标的距离，尤其是 y 轴的距离，因为被红色方块和障碍物都是横向移动的，越过一层障碍物是很有效的进展。

在特征中添加目标的距离后，npc 就已经尽力在尝试向上层运动了

```
for(Observation o : allobj){
    Vector2d p = o.position;
    int x = (int)(p.x/28); //squre size is 20 for pacman
    int y= (int)(p.y/28);  //size is 28 for FreeWay
    map[x][y] = o.itype;
    if(o.itype == 4){
        feature[874] = obs.getAvatarPosition().x-o.position.x;
        feature[875] = obs.getAvatarPosition().y-o.position.y;
    }//与目标的相对位置
```

但是 npc 常常不能把握机会，就是在可以向上的时候不向上而向下回避障碍物，同时总是在右下角上方没有可经过道路的时候躲避，那位移到上方出口的危险又增加了，所以增加一个是否能向上移动的特征，即 npc 上方是否有墙体。

```
boolean canUp=true;
if( obs.getImmovablePositions()!=null )
    {
    for(ArrayList<Observation> l : obs.getImmovablePositions())
        {
            for(Observation o : l)
                {
                    Vector2d p = o.position;
                    if(p.y==obs.getAvatarPosition().y-28 && p.x==obs.getAvatarPosition().x)
                        canUp=false;

                }
            allobj.addAll(l);
        }

    }
```

最后是对墙体和攻击的躲避，在攻击来临时要及时躲避。

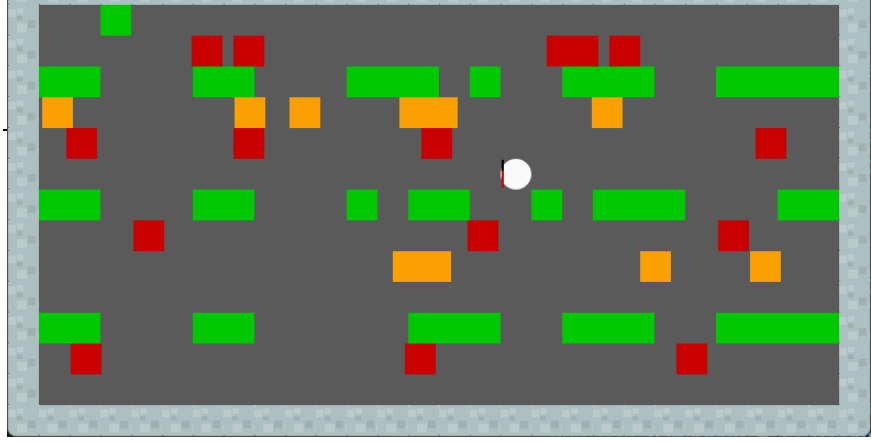
```
if( obs.getMovablePositions()!=null )
    {
        for(ArrayList<Observation> l : obs.getMovablePositions())
        {
            for(Observation o : l)
                {
                    Vector2d p = o.position;
                    if((p.x==obs.getAvatarPosition().x-28||p.x==obs.getAvatarPosition().x-56) && p.y==obs.getAvatarPosition().y)
                        needMove=true;
                }
            allobj.addAll(l);

        }
    }
```

在最后添加即可

```
Attribute att = new Attribute("GameTick" ); attInfo.addElement(att);
att = new Attribute("AvatarSpeed" ); attInfo.addElement(att);
att = new Attribute("AvatarHealthPoints" ); attInfo.addElement(att);
att = new Attribute("AvatarType" ); attInfo.addElement(att);
att = new Attribute("AvatarX" ); attInfo.addElement(att);
att = new Attribute("AvatarY" ); attInfo.addElement(att);
att = new Attribute("AvatarToGoalX" ); attInfo.addElement(att);
att = new Attribute("AvatarToGoalY" ); attInfo.addElement(att);
att = new Attribute("canUp" ); attInfo.addElement(att);
att = new Attribute("needMove" ); attInfo.addElement(att);
//action
```

添加好后可以走到第三层的入口性能相对有所优化



3. 强化学习参数修改。

修改了启发函数

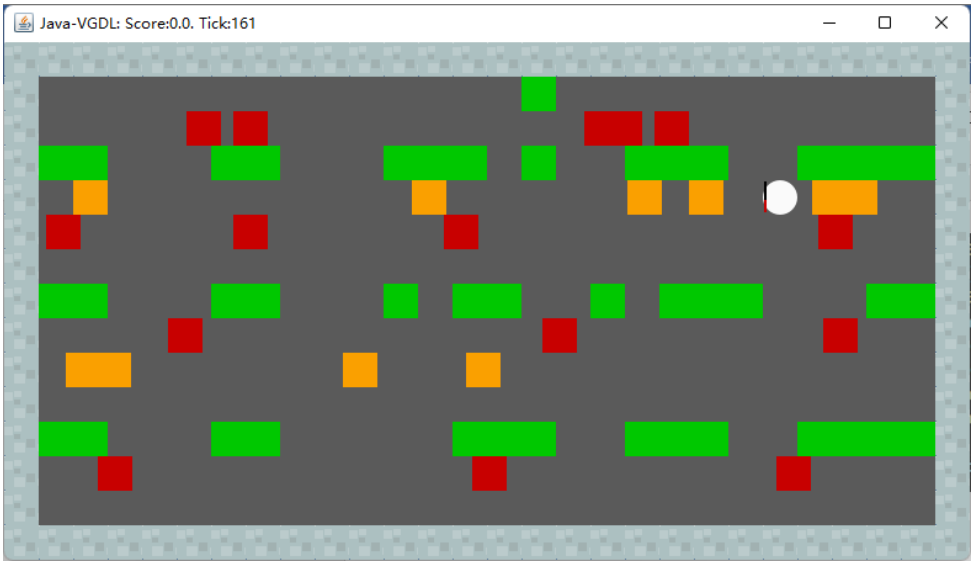
对 npc 的 y 值进行了加权，增加了生命值。

```java
public double evaluateState(StateObservation stateObs) {
    boolean gameOver = stateObs.isGameOver();
    Types.WINNER win = stateObs.getGameWinner();
    double rawScore = stateObs.getGameScore();
    rawScore+=(200-stateObs.getAvatarPosition().y);
    rawScore+=stateObs.getAvatarLimitHealthPoints();
    if(gameOver && win == Types.WINNER.PLAYER_LOSES)
        return HUGE_NEGATIVE;

    if(gameOver && win == Types.WINNER.PLAYER_WINS)
        return HUGE_POSITIVE;

    return rawScore;
}
```

虽然学习速度降低了很多，但是学习效果好了很多，甚至能走到接近最上面一层

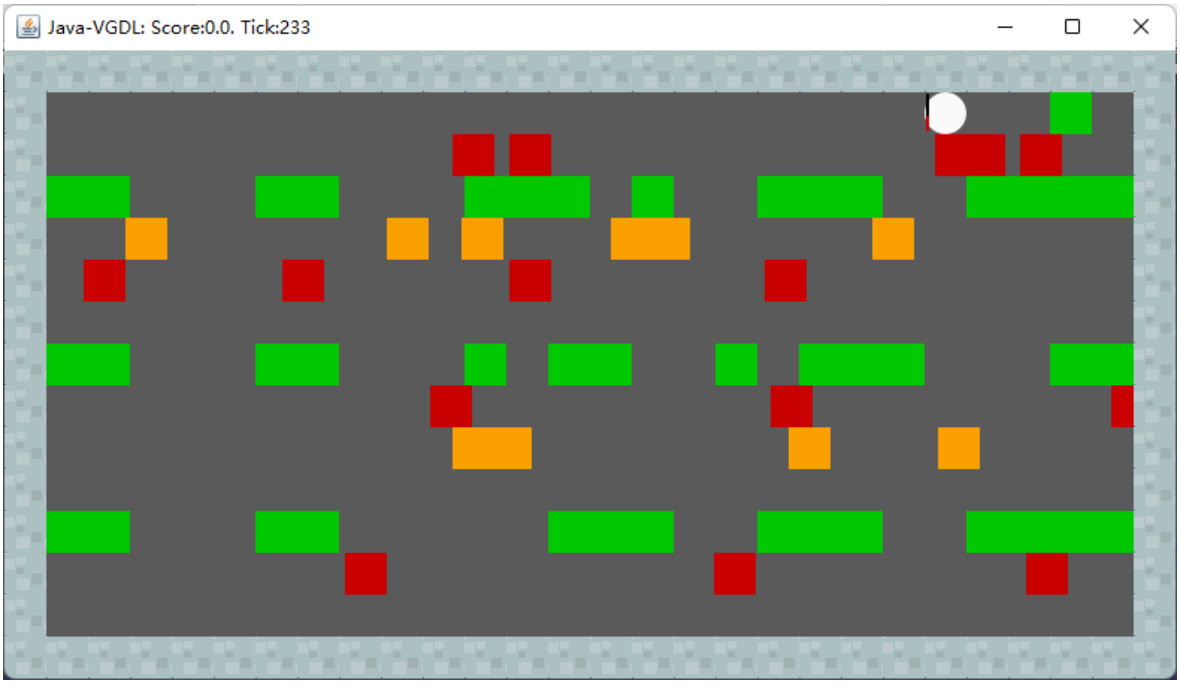


最后是对 SIMULATION_DEPTH、m_maxPoolSize、m_gamma 和 m_epsilon 的修改。

SIMULATION_DEPTH：随着深度的增加，学习效果也越来越好，越来越接近于目标，随之带来的问题就是训练的时长的增加。每一步的决策都需要 1 秒甚至更久。

m_maxPoolSize：增大了 dataset 的范围，但训练效果却没有明显的提升，同时学习时间也增加了。

m_gamma：这个值初始值为 0.99，一开始将其更改为 0.2，npc 的行为在初始经历一段学习后，进行了上下来回巡游的行为，但将其更改问 0.8 后，其学习效率和学习速度都上升了很多，甚至能完全穿过 freeway 达到最上层，该值的选取对性能的影响很大。



m_epsilon：该值选取决定了学习的随机性，当该值选取较大如 0.9 的时候，npc 呈现随机性很强的运动，当取很小 0.1 的时候 npc 更加保守，大部分时候选择在右下角躲避。该值的选取适中即可，初始给出的 0.3 是个很适宜的数组。

最终方案：将 m_gamma 修改为 0.8，m_maxPoolSize 更改为 500，SIMULATION_DEPTH 更改为 40，配合启发式函数的修改可以获得一个相对不错的效果。