

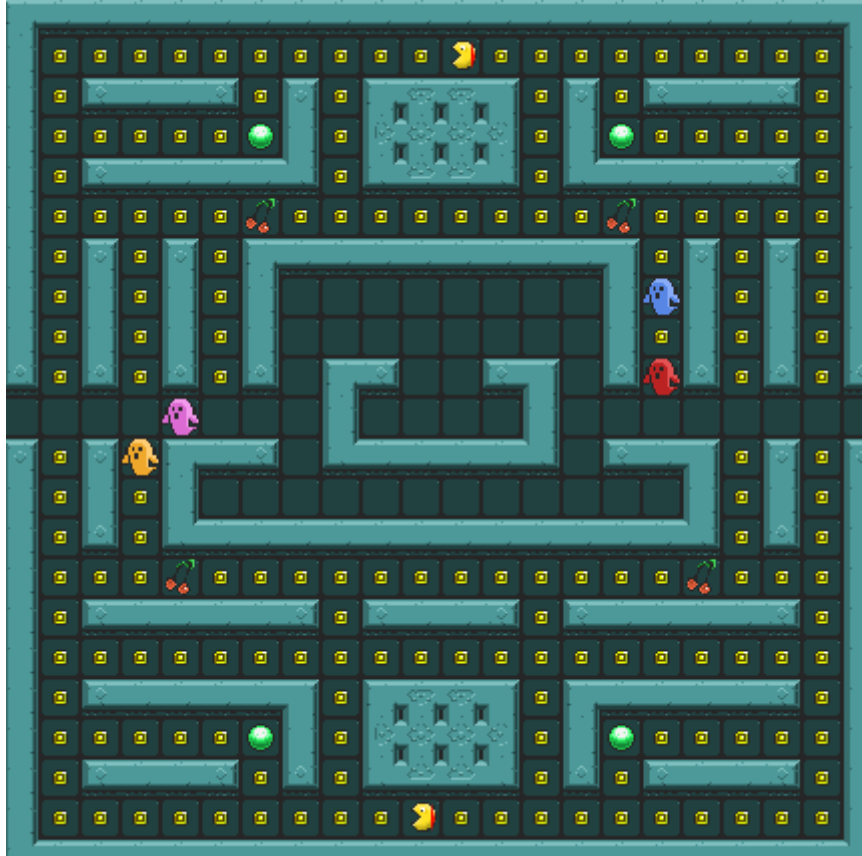
# HSEA-第一次作业

---

游戏框架基于 [gygai](#)，游戏名为 `pacoban`，是一个双人游戏。

游戏中，需要有两个agent控制两个吃豆人，要在躲避四个幽灵的前提下尽可能多的吃到金币，同时吃豆人也可以获得一些奖励的道具，获得分数的同时能够得到特殊的增益效果。

[作业框架](#)，点击即可下载。



## 关于游戏环境

---


游戏的配置文件位于 `\examples` 中，我们介绍 `\examples.pacoban.txt` 中的内容，它是对游戏环境的描述。


BasicGame key\_handler=Pulse no\_players=2 square\_size=20 obs=wall SpriteSet给出了游戏中出现过的物体的描述。


#### SpriteSet

floor > Immovable img=oryx/floor3  地板，是不可移动的。

food > Immovable

fruit > color=PINK img=newset/cherries2 

pellet > color=LIGHTYELLOW shrinkfactor=0.5 img=oryx/gold2 

power > Resource color=LIGHTGREEN shrinkfactor=0.8 img=oryx/orb2 

食物，是不可移动的。


nest > SpawnPoint img=portal portal=True invisible=True

redspawn > stype=redOk

orangespawn > stype=orangeOk

bluespawn > stype=blueOk

pinkspawn > stype=pinkOk


wall > Immovable img=oryx/wall3 autotiling=True  墙体

moving > 幽灵分为四个颜色：红色蓝色粉色和橙色。

ghost > 四个幽灵除了颜色外，其他行动相关的参数都是一致的。


red > singleton=True

redOk > RandomPathAltChaser stype1=hungryA,hungryB stype2=poweredA,poweredB cooldown=4 img=oryx/ghost3 cons=4

redSc > Fleeing stype=pacman maxDistance=500 cooldown=2 img=oryx/ghost1 


blue > singleton=True

blueOk > RandomPathAltChaser stype1=hungryA,hungryB stype2=poweredA,poweredB cooldown=4 img=oryx/ghost4 cons=4

blueSc > Fleeing stype=pacman maxDistance=500 cooldown=2 img=oryx/ghost1 


pink > singleton=True

pinkOk > RandomPathAltChaser stype1=hungryA,hungryB stype2=poweredA,poweredB cooldown=4 img=oryx/ghost5 cons=4

pinkSc > Fleeing stype=pacman maxDistance=500 cooldown=2 img=oryx/ghost1 

orange > singleton=True


orangeOk > RandomPathAltChaser stype1=hungryA,hungryB stype2=poweredA,poweredB cooldown=4 img=oryx/ghost6 cons=4

orangeSc > Fleeing stype=pacman maxDistance=500 cooldown=2 img=oryx/ghost1 

pacman > OrientedAvatar rotateInPlace=false speed=1

avatarA >

hungryA > color=YELLOW img=newset/pacman 

poweredA > color=ORANGE img=oryx/bullet1 

avatarB >

hungryB > color=YELLOW img=newset/pacman

poweredB > color=ORANGE img=oryx/bullet1

幽灵分为两个状态，Ok和Sc，Sc是白色的。

玩家可以操作的吃豆人pacman这里叫做avatar，有两种状态：饥饿状态hungry和强化状态powered。

InteractionSet InteractionSet 给出了两个物体相碰后的结果

wall pacman > bounceForward

wall wall > undoAll

wall EOS > killSprite

pacman pacman > stepBack

ghost wall > stepBack

玩家碰到墙后，墙会被向后推，但是如果墙有两层，那么玩家无法推动墙。ghost是无法通过墙的，也就是说，玩家可以使用墙体巧妙地困住ghost。

pacman EOS > wrapAround

EOS指地图边缘，当墙被推到地图边缘时会被消除；玩家可以通过地图边缘来到另一侧，而ghost不行。

ghost EOS ghost > stepBack

hungryA ghost > killSprite scoreChange=-1,0

hungryB ghost > killSprite scoreChange=0,-1

处于hungry状态的玩家碰到ghost会被消灭，玩家分数会被置为-1。

power hungryA hungryB > transformToAll stype=redOk stypeTo=redSc

power hungryA hungryB > transformToAll stype=pinkOk stypeTo=pinkSc

power hungryA hungryB > transformToAll stype=blueOk stypeTo=blueSc

power hungryA hungryB > transformToAll stype=orangeOk stypeTo=orangeSc 

power 和 处于hungry状态的玩家碰到，所有ghost被转化为Sc状态即变为白色，所有玩家变为powered状态，并持续200个timestep，之后会还原。

power hungryA hungryB > addTimer timer=200 ftype=transformToAll stype=redSc stypeTo=redOk killSecond=True

power hungryA hungryB > addTimer timer=200 ftype=transformToAll stype=pinkSc stypeTo=pinkOk killSecond=True

power hungryA hungryB > addTimer timer=200 ftype=transformToAll stype=blueSc stypeTo=blueOk killSecond=True

power hungryA hungryB > addTimer timer=200 ftype=transformToAll stype=orangeSc stypeTo=orangeOk killSecond=True

hungryA power > addTimer timer=200 ftype=transformToAll stype=poweredA stypeTo=hungryA

hungryB power > addTimer timer=200 ftype=transformToAll stype=poweredB stypeTo=hungryB

hungryA power > transformTo stype=poweredA

hungryB power > transformTo stype=poweredB

power avatarA > killSprite scoreChange=10,0

power avatarB > killSprite scoreChange=0,10

pellet avatarA > killSprite scoreChange=1,0

pellet avatarB > killSprite scoreChange=0,1

fruit avatarA > killSprite scoreChange=5,0

fruit avatarB > killSprite scoreChange=0,5

玩家碰到食物后都会获得奖励，Power、fruit、pellet奖励依次降低。

ghost poweredA > killSprite scoreChange=-40,0

ghost poweredB > killSprite scoreChange=0,40

ghost碰到强化的玩家后会被杀死，相应玩家会获得大量奖励

## 关于控制器和游戏框架

我们在程序包中附带了一些控制器，位于 `src\tracks\multiPlayer`，你可以参考这些控制器的写法去写你自己的控制器。

需要注意的是这些控制器并没有针对 `pacoban` 你需要自己根据游戏理解设计启发式函数和搜索算法。

## 控制程序

控制器位于 `src\tracks` 目录下，可用于多人游戏的控制器位于 `src\tracks\multiPlayer` 中。我们以 `sampleOneStepLookAhead` 控制程序为例，可以看到：

每个控制器都是由一个继承自 `core.player.AbstractMultiPlayer.java` 的Java类构成，这个类必须命名为Agent.java且必须实现两个方法：

```

public Agent(StateObservation stateObs, ElapsedCpuTimer elapsedTimer, playerId)

public Types.ACTIONS act(StateObservation stateObs, ElapsedCpuTimer elapsedTimer)

```

两个方法都需要两个参数 `StateObservation stateObs` 和 `ElapsedCpuTimer elapsedTimer`，其中：

**StateObservation stateObs:** 定义了 agent 当前所处的状态信息，一般通过上层程序传递下来，你可以通过在可运行的控制器上加断点来查看其所包含的内容；另外，源文件位于 `src\core\game\StateObservationMulti.java`，你可以去查看其自带的一些方法。

**ElapsedCpuTimer elapsedTimer:** 是一个类，它允许查询代理返回一个动作的剩余CPU时间。

关于 **public Types.ACTIONS act** 类，它是控制程序的核心内容，我们在此做介绍：

`sampleOneStepLookAhead` 方法是向前搜索一步，选择收益最大的动作执行。

```

public Types.ACTIONS act(StateObservationMulti stateObs, ElapsedCpuTimer elapsedTimer) {

    Types.ACTIONS bestAction = null;           初始化参数
    double maxQ = Double.NEGATIVE_INFINITY;

    //A random non-suicidal action by the opponent.
    Types.ACTIONS oppAction = getOppNotLosingAction(stateObs, id, oppID);
    SimpleStateHeuristic heuristic = new SimpleStateHeuristic(stateObs);

    for (Types.ACTIONS action : stateObs.getAvailableActions(id)) {

        StateObservationMulti stCopy = stateObs.copy();  遍历能做的所有行动，在
                                                         环境的拷贝进行模拟

        //need to provide actions for all players to advance the forward model
        Types.ACTIONS[] acts = new Types.ACTIONS[no_players];

        //set this agent's action
        acts[id] = action;
        acts[oppID] = oppAction;  要确定好自己和对手的行
                                  动，才可以向前模拟

        stCopy.advance(acts);

        double Q = heuristic.evaluateState(stCopy, id);
        Q = Utils.noise(Q, this.epsilon, this.m_rnd.nextDouble());
        if (Q > maxQ) {
            maxQ = Q;
            bestAction = action;  评估模拟后的结果，选择
                                  Q值最大的action
        }
    }

    //System.out.println("=====" + getPlayerID() + " " + maxQ + " " + bestAction + "=====");
    //System.out.println(elapsedTimer.remainingTimeMillis());
    return bestAction;
}

```

另外，框架还提供了额外的已经实现好的启发式信息，位于

`src\tracks\multiPlayer\tools\heuristics` 中，你也可以在实现的过程中作为参考。

## 测试程序

测试程序位于 `src\tracks\multiPlayer\TestMultiPlayer`，其中已有详细的注释。

```

public class TestMultiPlayer {
    public static void main(String[] args) {
        // 一些可用的控制器
        String doNothingController = "tracks.multiPlayer.simple.doNothing.Agent";
        String randomController = "tracks.multiPlayer.simple.sampleRandom.Agent";
        String oneStepController = "tracks.multiPlayer.simple.sampleOneStepLookAhead.Agent";
        String sampleMCTSController = "tracks.multiPlayer.advanced.sampleMCTS.Agent";
        String sampleRSController = "tracks.multiPlayer.advanced.sampleRS.Agent";
        String sampleRHEAController = "tracks.multiPlayer.advanced.sampleRHEA.Agent";
        String humanController = "tracks.multiPlayer.tools.human.Agent";

        // 可以在这里设置游戏中的控制器, 如果要自己玩, 可以把其中一个控制器改为humanController, 自己就可以控制相应的agent
        String controllers = oneStepController + " " + oneStepController;
        boolean visuals = true;
        String recordActionsFile = null;
        // 随机种子的设置
        int seed = new Random().nextInt(); // 你可以使用随机种子
        // int seed = 2020; // 最终运行的时候你需要固定种子
        int levelIdx = 0; // 选择游戏关卡, 从0到4一共5关
        String gameName = "pacoban";
        String game = "examples/pacoban.txt";
        String level1 = game.replace(gameName, replacement: gameName + "_lvl" + levelIdx);
        // 1. 玩一轮可视化的游戏
        // ArcadeMachine.runOneGame(game, level1, visuals, controllers, recordActionsFile, seed, 1);
        // 2. 在前N关, 玩M次:
        // int N = 0;
        // int M = 3;
        // for (int i = 0; i <= N; i++) {
        //     level1 = game.replace(gameName, gameName + "_lvl" + i);
        //     ArcadeMachine.runGames(game, new String[]{level1}, M, controllers, null);
        // }
        // 每一关会返回结果, 即四个数: 前两个数表示哪个玩家胜利了, 后两个数表示在M次运行的平均分数;
        // 我们关注的是玩家0和玩家1的总分数要高, 报告中的分数也是两位玩家的分数和。
    }
}

```

## 作业内容

作业需要你理解游戏内容, 根据自己的理解设计启发式函数和搜索算法, 并且在多个难度的游戏下取得不错的分数, 最终根据实验内容撰写实验报告。具体的任务有如下几个:

- 任务一: 理解游戏。
  - 在测试程序中提供了玩一轮可视化的游戏的接口, 你可以使用预先提供的模拟器玩, 也可以自己玩;
  - 你需要在游戏过程中逐步理解游戏, 即如何达到更高的分数, 然后以此为基础去设计搜索算法, 并在报告中谈到你的理解和你打算如何实现搜索算法。
- 任务二: 实现你自己的搜索算法。
  - 根据任务一中你对游戏的理解, 实现带启发式函数的搜索算法, 并成功运行;
  - 在报告中体现出你是如何做的, 简述你的代码实现过程。
- 任务三: 调整你的算法, 并最终在多个级别的游戏下进行测试。
  - 你需要继续调整你的搜索算法, 想办法让最终的分数尽可能高的同时使算法效率不要太低, 并在报告中体现;
  - 测试接口已经在测试程序中准备好。你需要固定种子为2020运行测试程序, 在报告中说明你在每个关卡中运行10次的平均分数(需要报告的分数为两个玩家的分数和)。
- 最后, 作为这次任务的收尾, 你可以自己和你设计出来的agent共同进行游戏, 看看结果怎么样。

在报告中, 你需要分别阐述这三个任务的内容。

注意到, 在 `src\core\competition\CompetitionParameters.java` 中存放了一些限制控制器搜索和执行动作的参数, 你在任务二的测试算法时可以将该限制取消, 即 `public static final boolean TIME_CONSTRAINED = false`; 但是在任务三中, 你需要加上这一限制进行分数评估, 并通过 `public static final int ACTION_TIME` 和 `public static final int ACTION_TIME_DISQ` 的参数设置使

你的算法可以运行。你应该在报告中体现你的参数设置，尽可能实现高效的算法。

## 作业提交要求

你需要提交一份压缩文件，以“学号\_姓名”的方式命名，如“MG1937001\_张三.zip”。文件中需要包含完整的项目代码和实验报告(模板)，在作业截止日期前发送到[xuek@lamda.nju.edu.cn](mailto:xuek@lamda.nju.edu.cn)，邮件标题命名和压缩文件一致。

## 作业评分准则

作业的评分主要参考三个任务的完成情况和报告的书写。我们会抽查代码运行结果和运行效率，**若发现结果造假和作业出现雷同的情况，会根据相关规定给予惩罚，详情请参考课程主页中“学术诚信”的相关内容。**

请同学们务必独立完成作业！