

作业1: Bait游戏

[Print](#)
([Back to course page](#))

[Edit](#)

理解GVGAI框架

GVG-AI 框架是为了通用人工智能的研究开发的游戏框架，基于VGDL（视觉游戏描述语言），能够构成多种游戏。本次作业使用一种推箱子游戏“Bait”，下面详述。



编程语言为Java（如果你没有学习过Java语言，Java的语法跟C++非常接近，但更简单。Java的编译和运行需要安装 [JDK](#) ）
首先，下载本次作业程序包 [vggai-assignment1.zip](#)（程序包中的代码有所调整）
并在你使用的开发环境中配置好相应的工程，如果配置正确，可以允许 Test.java 启动游戏并可以通过键盘操纵。你可以通过自己玩游戏来了解游戏规则。



[Edit](#)

关于Bait游戏

Bait游戏的描述文件在 vggai-assignment1/examples/gridphysics 目录下

- bait.txt 为游戏的配置文件
- bait_lv10.txt ~ bait_lv14.txt 为游戏5个关卡的地图描述文件，根据配置文件中定义的符号来设置

下图解释了**bait.txt**文件的内容

<p>游戏基本设置</p> <p>SpriteSet这一段定义游戏中出现的物体</p> <p>这里定义了 洞(hole)，属性为不可移动 (Immovable) </p> <p>玩家操纵的精灵(avatar)，</p> <p>以及精灵有两个状态，没钥匙  和有钥匙 </p> <p>蘑菇(mushroom)，不可移动 </p> <p>钥匙(key)，可移动(Passive) </p> <p>目标(goal)，不可移动 </p> <p>盒子(box)，可移动 </p> <p>地图元素，用在地图描述文件中</p> <p>0 表示洞</p> <p>1 表示盒子</p> <p>k 表示钥匙</p> <p>g 表示目标</p> <p>m 表示蘑菇</p> <p>InteractionSet这一段定义了两个物体落在同一个位置的结果</p> <p>精灵与墙相 --> 精灵退后</p> <p>精灵与洞 --> 精灵消失</p> <p>盒子与精灵 --> 盒子向前移动</p> <p>盒子与墙、盒子与盒子、盒子与蘑菇 --> 撤销本轮动作</p> <p>盒子与洞 --> 两者都消失，分数加1</p> <p>没钥匙与钥匙 --> 状态转换为有钥匙</p> <p>钥匙与精灵 --> 钥匙消失，精灵状态转变为有钥匙</p> <p>没钥匙与目标 --> 精灵退后</p> <p>目标与钥匙 --> 目标消失，分数加5</p> <p>蘑菇与精灵 --> 蘑菇消失，分数加1</p> <p>TerminationSet定义了游戏终止的条件</p> <p>剩余0个目标时，终止，并且胜利</p> <p>剩余0个精灵时，终止，并且失败</p>	<pre>BasicGame key_handler=Pulse square_size=50 SpriteSet hole > Immovable color=DARKBLUE img=hole avatar > MovingAvatar nokey > img=alien withkey > img=alien mushroom > Immovable img=mushroom key > Passive img=key goal > Immovable img=door box > Passive img=box shrinkfactor=0.8 LevelMapping 0 > hole 1 > box k > key g > goal m > mushroom InteractionSet avatar wall > stepBack avatar hole > killSprite box avatar > bounceForward box wall box mushroom > undoAll box hole > killBoth scoreChange=1 nokey key > transformTo stype=withkey key avatar > killSprite nokey goal > stepBack goal withkey > killSprite scoreChange=5 mushroom avatar > killSprite scoreChange=1 TerminationSet SpriteCounter stype=goal limit=0 win=True SpriteCounter stype=avatar limit=0 win=False</pre>
--	--

从 **bait.txt** 文件可以看出游戏规则

- 精灵要先拿到钥匙，然后走到目标
- 如果精灵吃了蘑菇，那么额外加1分
- 精灵不能掉进洞里，否则失败
- 精灵可以推盒子吧洞填上，洞填上后就可以通过，并且每填一个洞有1分的奖励
- 只能向前推一个盒子，不能推两个盒子，也不能把盒子推到墙、蘑菇上（盒子可以推到目标上再推开）
- 箱子可以覆盖钥匙，一旦覆盖，必须推开箱子取得钥匙才能成功
- 另外， [游戏的时间为1000ticket](#)，时间结束即失败

[Edit](#)

关于控制程序

在 controllers 目录下，已经有多个样例控制程序了，我们通过最简单的样例：“随机控制” controllers/sampleRandom/Agent.java 来了解基本的控制方法

```
/**
 * Picks an action. This function is called every game step to request an
 * action from the player.
 * @param stateObs Observation of the current state.
 * @param elapsedTimer Timer when the action returned is due.
 * @return An action for the current state
 */
public Types.ACTIONS act(StateObservation stateObs, ElapsedCpuTimer elapsedTimer) {

    ArrayList<Observation>[] npcPositions = stateObs.getNPCPositions();
    ArrayList<Observation>[] fixedPositions = stateObs.getImmovablePositions();
    ArrayList<Observation>[] movingPositions = stateObs.getMovablePositions();
    ArrayList<Observation>[] resourcesPositions = stateObs.getResourcesPositions();
    ArrayList<Observation>[] portalPositions = stateObs.getPortalsPositions();
    grid = stateObs.getObservationGrid();

    /*printDebug(npcPositions,"npc");
    printDebug(fixedPositions,"fix");
    printDebug(movingPositions,"mov");
    printDebug(resourcesPositions,"res");
    printDebug(portalPositions,"por");
    System.out.println();*/

    Types.ACTIONS action = null;
    StateObservation stCopy = stateObs.copy();

    double avgTimeTaken = 0;
    double acumTimeTaken = 0;
    long remaining = elapsedTimer.remainingTimeMillis();
    int numIters = 0;

    int remainingLimit = 5;
    while(remaining > 2*avgTimeTaken && remaining > remainingLimit)
    {
        ElapsedCpuTimer elapsedTimerIteration = new ElapsedCpuTimer();
        ArrayList<Types.ACTIONS> actions = stateObs.getAvailableActions();
        int index = randomGenerator.nextInt(actions.size());
        action = actions.get(index);

        stCopy.advance(action);
        if(stCopy.isGameOver())
        {
            stCopy = stateObs.copy();
        }

        numIters++;
        acumTimeTaken += (elapsedTimerIteration.elapsedMillis()) ;
        //System.out.println(elapsedTimerIteration.elapsedMillis() + " --> " + acumTimeTaken + " (" + remaining + ")");
        avgTimeTaken = acumTimeTaken/numIters;
        remaining = elapsedTimer.remainingTimeMillis();
    }

    return action;
}
```

游戏程序会不断调用控制器的act函数
stateObs对象表示当前游戏局面，
elapsedTimer是计算一次动作可用的时间

可以从stateObs中读取当前局面上的物体
(在这个控制器中并未使用)

stCopy是局面对象的拷贝，用于模拟执行动作后的结果，通过在stCopy中执行
动作，可以“预见”一系列动作带来的结果，并根据这个结果来选择当前最优
的一步动作

循环直到快要超时

计时

获得当前可以执行的动作集合

随机选取一个动作

在stCopy中执行该动作，stCopy中的局面会发生改变（在这个控制器总并没有用处）

如果stCopy中游戏结束，
则重新开始

返回选择的动作

从上图可以看出

- act函数是控制程序的关键，需要在给定的时间内返回一个动作
- stCopy是用来理解游戏变化的“仿真器”，当我们进行搜索时，需要在stCopy上“模拟”动作的执行，从而能够看到状态发生了什么变化

[Edit](#)

作业内容

程序包中 Assignment1.java 已经准备好运行本次作业的4个任务

[Edit](#)

任务 1

针对第一个关卡，实现深度优先搜索 controllers.DepthFirst.java:

在游戏一开始就使用深度优先搜索找到成功的路径通关，记录下路径，并在之后每一步按照路径执行动作。注意在搜索时避免回路，可使用**StateObservation**类的**equalPosition**方法判断状态是否相等。

由于搜索到通关路径所需时间较长，通过CompetitionParameters.ACTION_TIME来设置足够的时间来允许完成搜索。

[Edit](#)

任务 2

在任务 1 的基础上，实现深度受限的深度优先搜索 controllers.LimitedDepthFirst.java:

修改为每一步进行一次深度搜索，但这时不需要一定搜索到通关，而是搜索到一定的深度，再设计一个启发式函数判断局面好坏。

同时CompetitionParameters.ACTION_TIME设置较小，需要在设置时间内完成一次决策。

针对Bait这个特定的游戏，我们可以利用目标的位置和钥匙的位置构造启发式函数：

```
ArrayList[] fixedPositions = stateObs.getImmovablePositions();
ArrayList[] movingPositions = stateObs.getMovablePositions();
Vector2d goalpos = fixedPositions[1].get(0).position //目标的坐标
Vector2d keypos = movingPositions[0].get(0).position //钥匙的坐标
```

该游戏中坐标从 (0,0) 开始，50为一个单元，在第一关中目标的坐标为 (50,50)，钥匙的坐标为 (100,200)。

[Edit](#)

任务 3

在任务2的基础上，将深度优先搜索换成A*算法 controllers.Astar.java:
并尝试在第二关、第三关中使用A*算法

[Edit](#)

任务 4

阅读 controllers.sampleMCTS.Agent.java 控制程序，并介绍其算法。

[Edit](#)

作业报告

本次作业需要提交报告和代码。对于以上4个任务，报告需分别详细介绍对代码的实现。 使用[这个文档模版\(点击下载\)](#) 撰写实验报告。

[Edit](#)

作业提交

将 [作业报告](#) 存储为PDF文件，用学号命名，例如151221001.pdf，并与的源码打包为 学号命名的.zip文件，例如 151221001.zip

上传到 ftp://lamda.nju.edu.cn/AI/assignment1/ (用户名: ai18, 密码: ai18)
(注意：该ftp不能替换文件，上传一次后，如果需要修改，请在文件名后加上版本号再上传，例如151221001-1.zip)

注意： **作业严禁抄袭！**