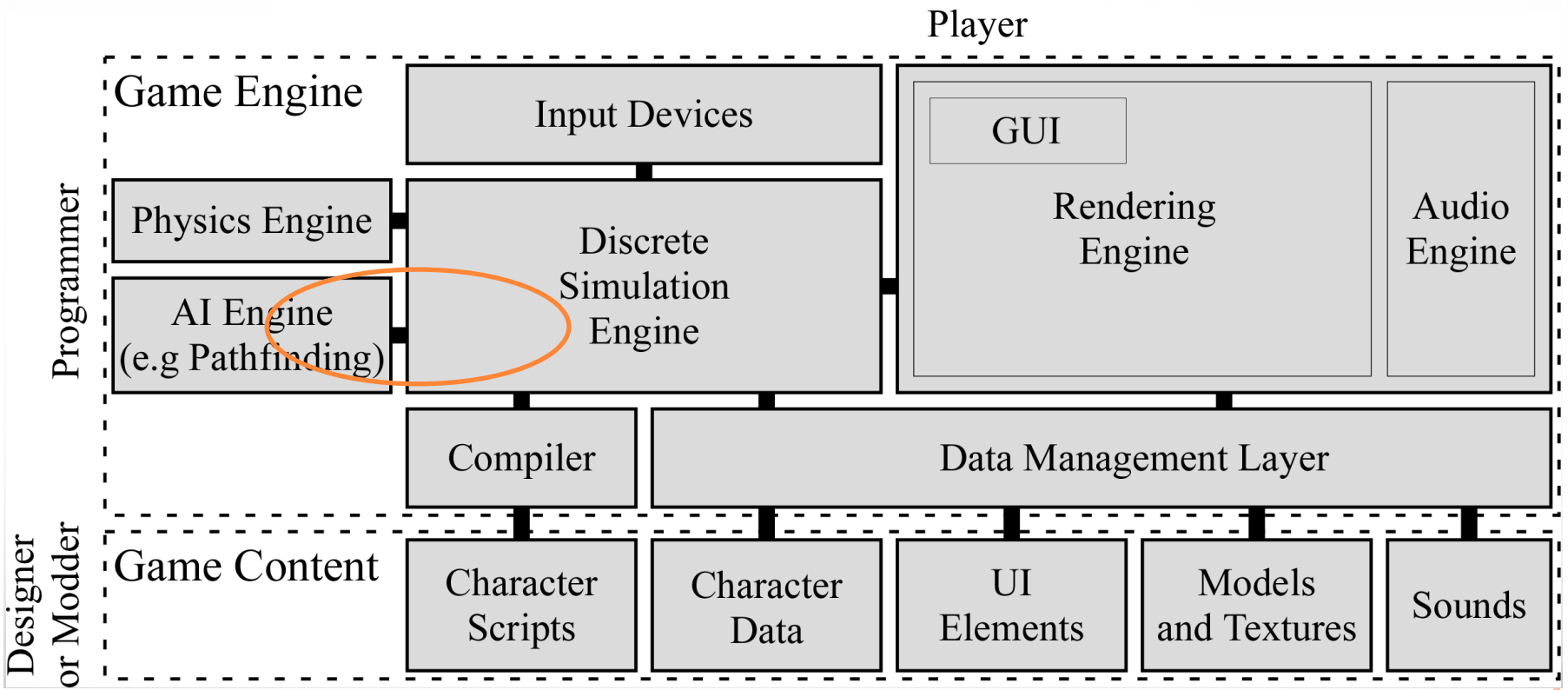
Introduction to Computer 3D Game Development

AI in Games

潘茂林，panml@mail.sysu.edu.cn
中山大学·软件学院

Ref cornell

Architecture: The Big Picture



How to control action of NPC objects?

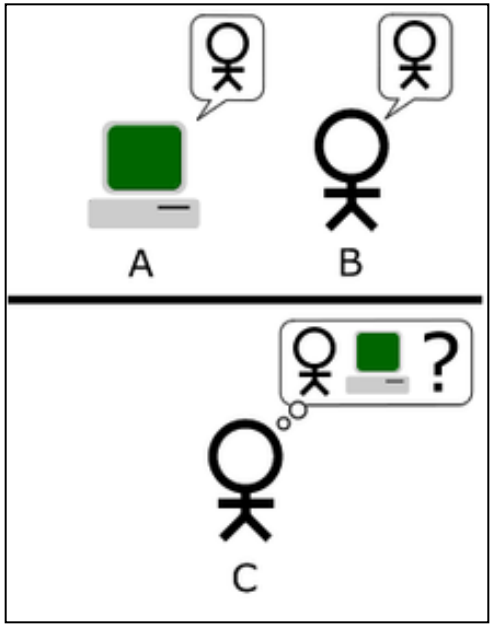
目录

- 游戏智能 AI
  - 什么是人工智能
  - 游戏智能与人工智能的区别
  - 游戏智能的应用
- 游戏智能实现常用方法
  - 感知-思考-行为模型
  - 实验一: 决策树
  - 寻路智能
  - 策略智能
  - 10 种使智能变得萌笨的方法！！！
- 面向对象的编程思考
  - Lambda 表达式

What is Artificial Intelligence?

- "the study and design of *intelligent agents*", where an intelligent agent is a system that **perceives** its environment and takes **actions** that **maximize** its chances of success

- 图灵测试

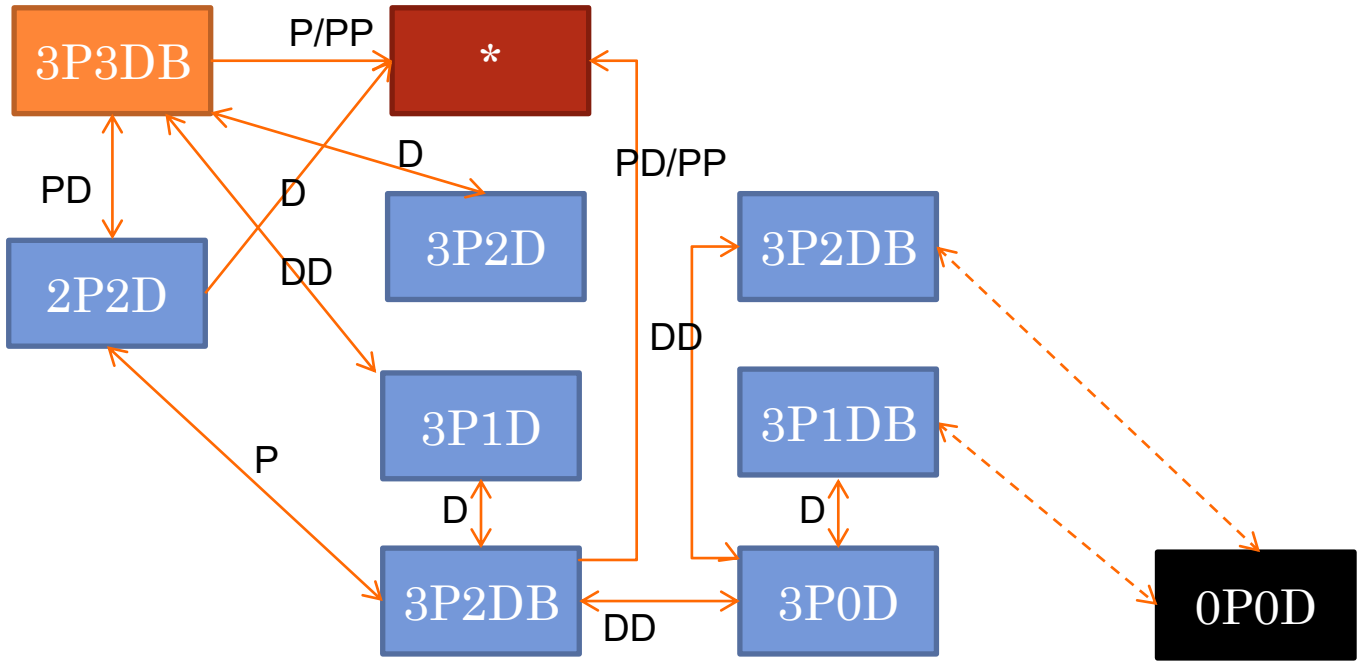


图灵试验一个标准的模式：C使用问题来判断A或B是人类还是机械

- 游戏中的智能：
  - Google Master 仅是 IT 技术进展
  - 未来 Google “入门”的“业余一级围棋手”才是有商业价值的游戏产品
  - 游戏 AI 是为不同人群服务，不是纯技术

案例研究：P&D游戏的智能

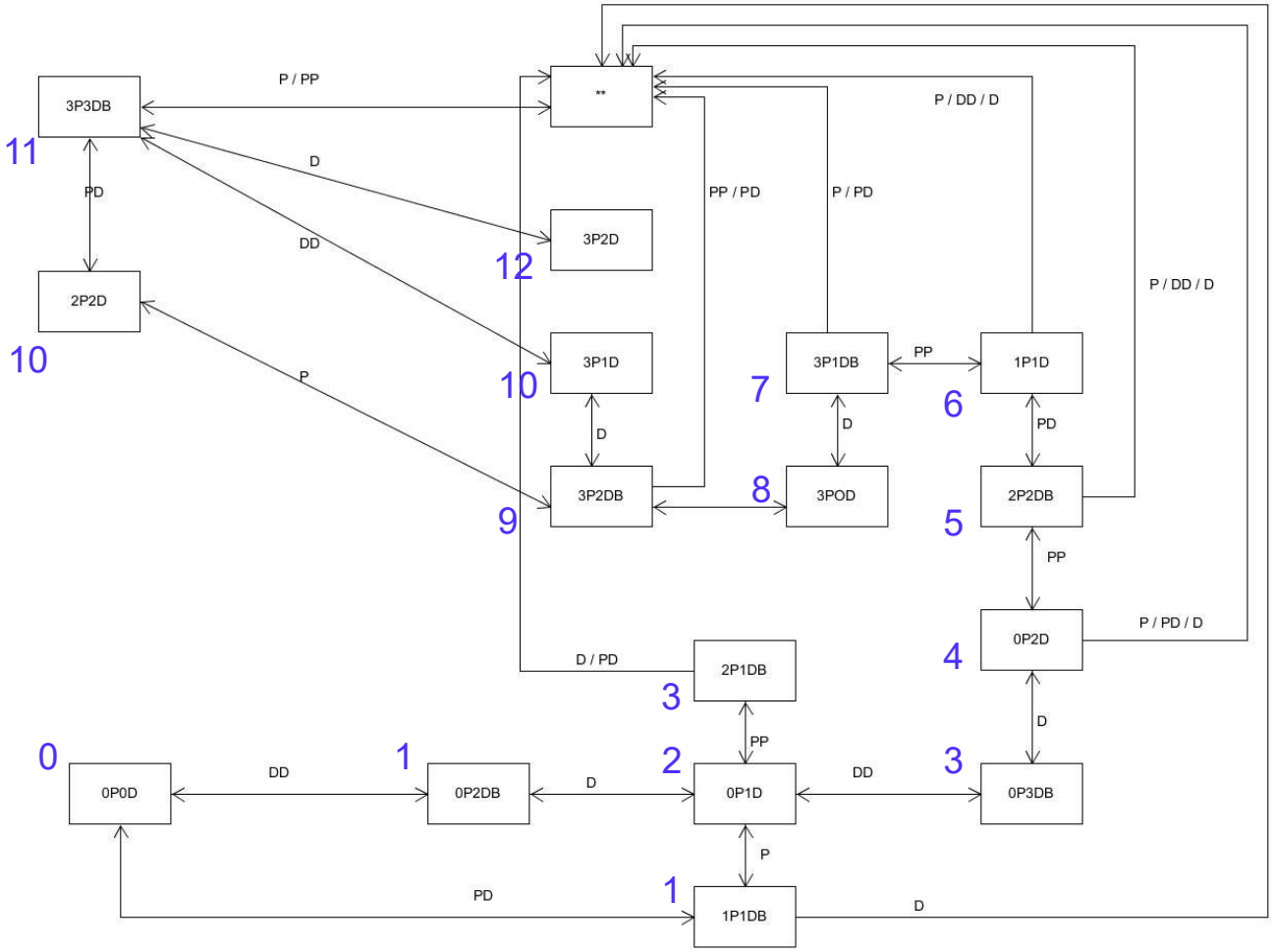
- 为了帮助小朋友玩 P&D 过河，你决定开发next功能，提示下一步最佳玩法？但怎么设计呢？



案例研究：P&D游戏的智能

- 观察状态图
  - 开始状态 [3P3DB]
  - 成功状态 [0P0D]；失败状态 *
  - 中间状态 [xPyDX], 其中 x>=y and [(3-x)>=(3-y) or (3-x)==0]
  - 可能动作：{P，D，PP，PD，DD}
- 问题求解：
  - 对于任意非成功/失败状态，找一条最短路径到达成功状态
  - 第一个动作即是问题的解
- 规约方法：
  - 最短路径

完整状态图



供图：http://blog.csdn.net/kiloveyousmile/article/details/71727667

案例研究：P&D游戏的智能

- 实现
  - 计算得到状态图
  - 如何表示？
  - 计算每个有效状态的最短路径，得到第一个动作

游戏智能
游戏 AI vs. 经典 AI

- **Classical**: Design of *intelligent agents*
  - 环境感知, 成功最大（优化问题）
  - 计算科学领域
  - 研究内容: 规划问题, 机器学习

- **Game**: Design of *rational behavior*(理性行为)
  - 不需要优化(and often will not)
  - 通常是关于人性化的“描述”
  - 属于认知科学
- NPC（非人控制角色） 能和对手开展 *meaningful choices* 。即要遇强则强，遇弱则弱；或形成挑战，而不是不可战胜。

游戏智能
实现 NPC 的 Meaningful Choices 的方法

- 收集信息的能力：
  - AI 对象能获得的有价值信息
  - 信息的完整性
  - 与 player 的信息对称
- 为了实现目标的判断动作能力：
  - AI 对象拥有哪些知识做行为判断
  - 动作或动作序列对目标实现的效果
  - 对 player 的知识应用和响应速度的挑战合理性
- 学习能力：
  - AI 对象基于历史信息的学习、优化能力
  - 与 player 学习曲线的一致性

游戏AI设计者很容易使玩家处于死地（*not fun*）！

游戏智能
Case study: NPC Shooting

- NPC Sensing information
  - Position? Velocity? Acceleration?
- NPC Acting
  - Calculation
  - Prepare gun time dt
  - Bullet speed and power
  - Hit rates
- NPC Learning
  - Player's preference discovery
  - Learning time

游戏智能
Case study: NPC Shooting

	fool-NPC	NPC1	NPC2	Smart-NPC
信息		位置	位置	位置 速度
行为 **准备时间** **子弹速度** **攻击力** **瞄准能力**	固定方向或散射 固定 固定	向用户射击 固定 慢	向用户射击 固定 快 低 给定概率高	当前位置和下一个位置间随机射击 快 快 低 低
学习				
玩家挑战	观察能力 节奏控制力	位置选择 子弹对抗 躲避节奏	位置选择 观察与反应 运动中对抗 运气	勇气 高度灵敏性 选择有效攻击武器 运气

Has game AI or not in shooting?

游戏智能
游戏中 AI 的运用

- Autonomous Characters (NPCs)
  - Mimic(模仿) the “personality” of the character
  - May be opponent or support character
- Strategic Opponents（策略对抗）
  - AI at the “player level”
  - Closest to classical AI
- Dialog（会话）
  - Intelligent commentary
  - Narrative（故事） management

游戏智能常用方法
游戏智能的类型

- Behavioral（行为） AI
  - A NPC acts with its “personality”（个性化，拟人化）
  - Sense-Think-Act cycle
- Strategic（策略） AI
  - Decision tree（决策树）
- Path findings（寻路） AI
  - Breadth-First Search
  - A* algorithm
  - Steering（巡航）

游戏智能常用方法
行为 AI: Sense-Think-Act 模型

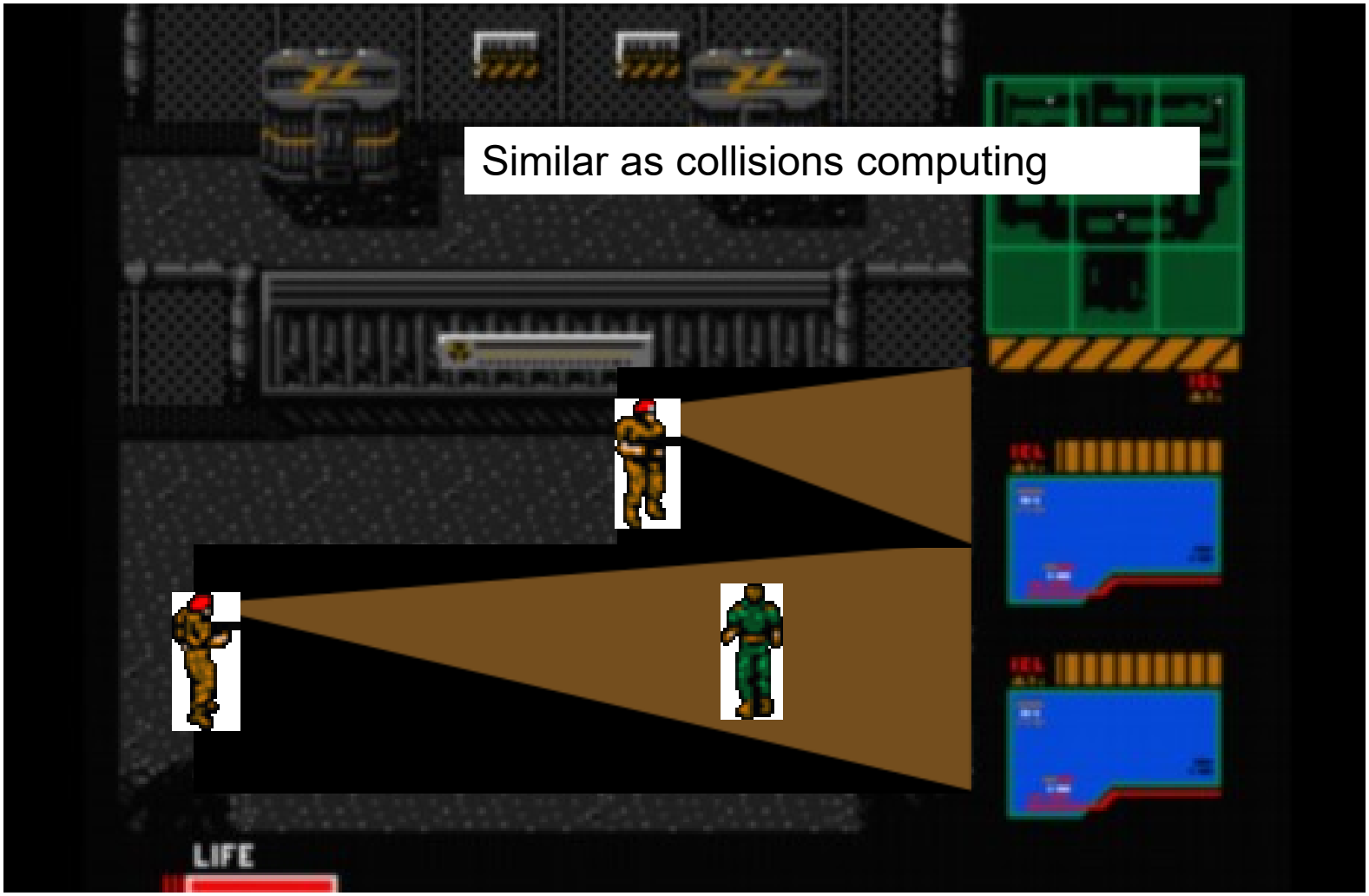
- Sense:
  - Perceive the world
  - Reading the game state
  - Example: enemy near?
- Think:
  - Choose an action
  - Often merged with sense
  - Example: fight or flee
- Act:
  - Update the state
  - Simple and fast
  - Example: reduce health

游戏智能常用方法
行为 AI: （限制）感知

- 限制 获取信息的能力
  - **Vision: 限制可视域**
    - 给出准确的位置与信息
    - 使用障碍物和范围限制
    - 使用距离使得信息获取减少或运动减慢
  - **Sound: 指向性**
    - 给出方向和距离
    - 要求玩家“追踪声源”决定行为
  - **Smell:指向性**
    - 没有方向和距离; *proximity* 仅
    - 要求玩家“追踪源”决定行为

Sensing: Line-of-Sight

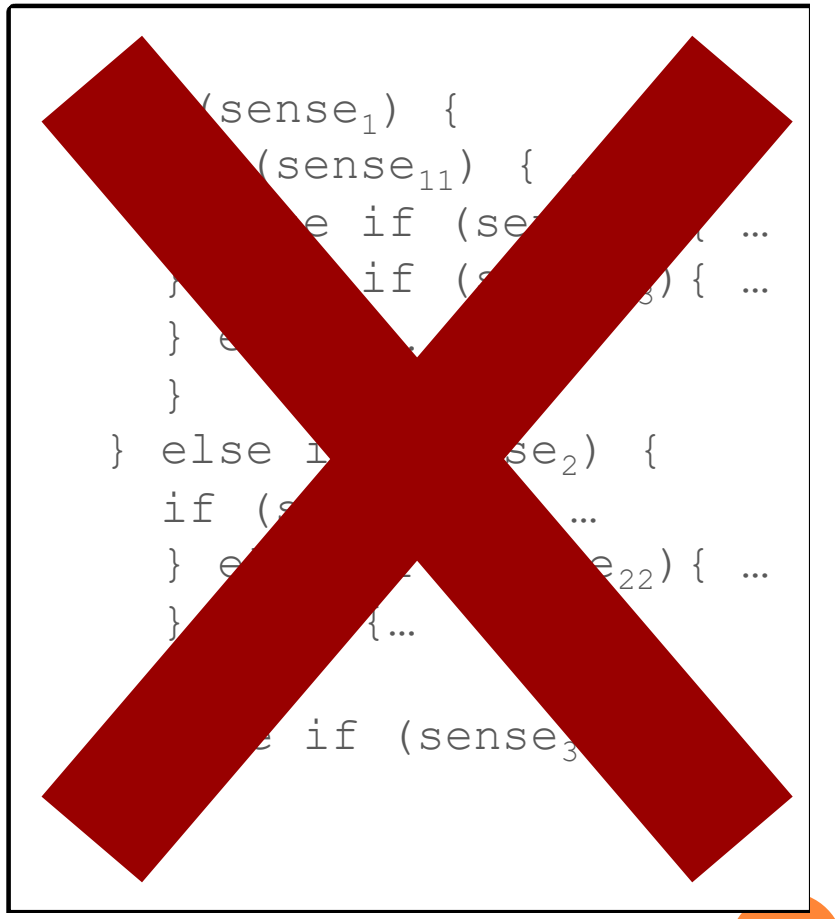
Sensing: Line-of-Sight



3D Line-of-Sight: Ray Casting

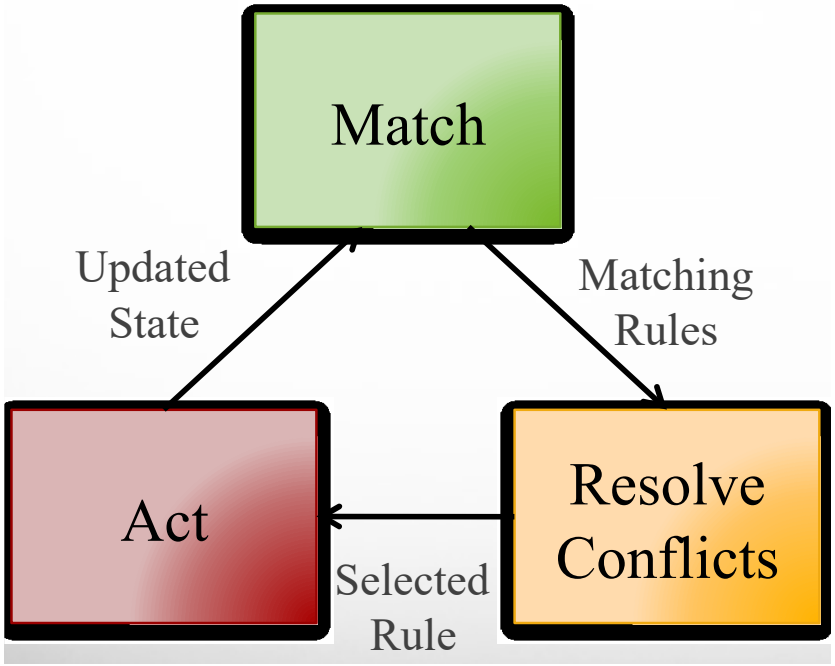
游戏智能常用方法
行为 AI: 思考

- 一堆条件语句
  - “硬” 编码
  - 难以修改
- 要抽象需求:
  - 易于可视化建模
  - 反映 “认知与思考”
- 努力分离技能
  - **Sensing:** 由程序员开发
  - **Thinking:** 由设计师开发



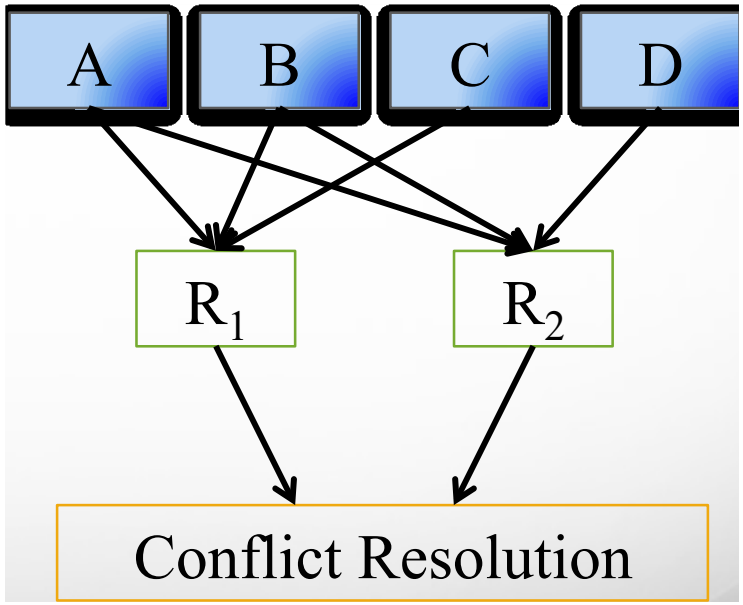
游戏智能常用方法
思考一：基于规则推理的 AI

- 三个基本步骤
  - Match（匹配）
    - 检查游戏规则“条件部分”
    - 返回所有匹配的规则
  - Resolve（冲突求解）
    - 仅能输出一个规则
    - 使用元规则选择
  - Act（行为）
    - 执行动作
    - 修改状态



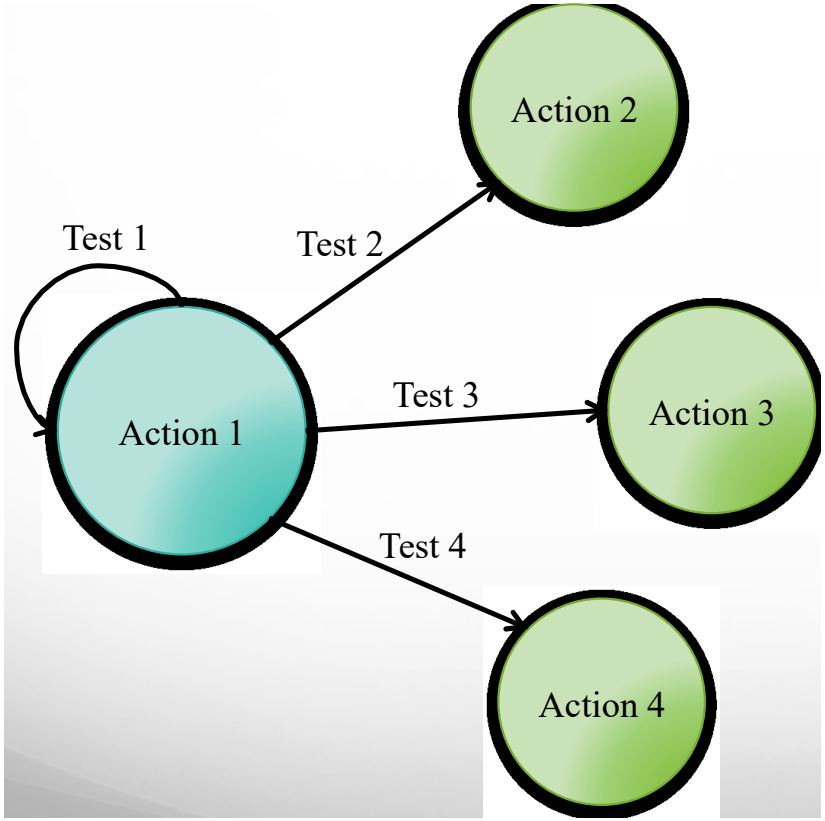
游戏智能常用方法
思考一：基于规则AI：冲突求解策略

- Fusion（融合）
  - combine with ordering
- Data statistic
  - Select by most recent used
  - Voted by most rules
  - Most unused actions
- Random
  - Select randomly
  - May “weight” probabilities



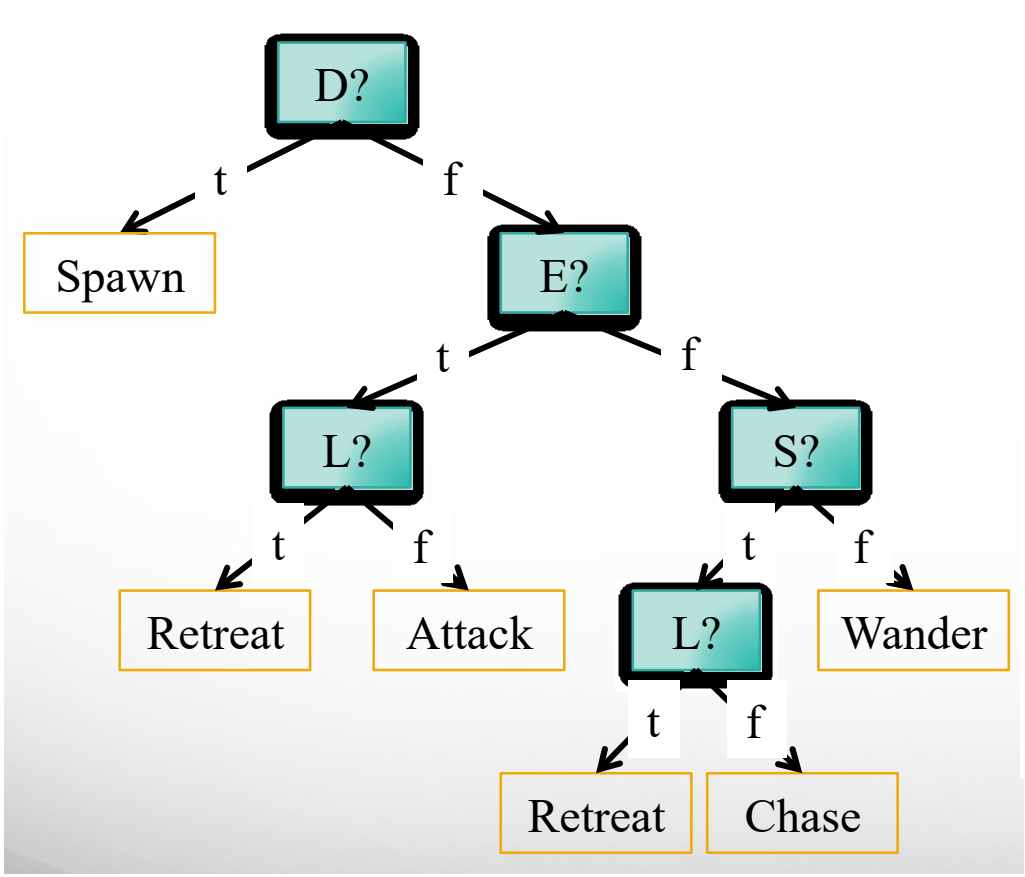
游戏智能常用方法
思考二：基于状态机AI

- Finite State Machines



游戏智能常用方法
思考三：基于决策树的AI

- Decision Trees



游戏智能常用方法
行动：基本准则

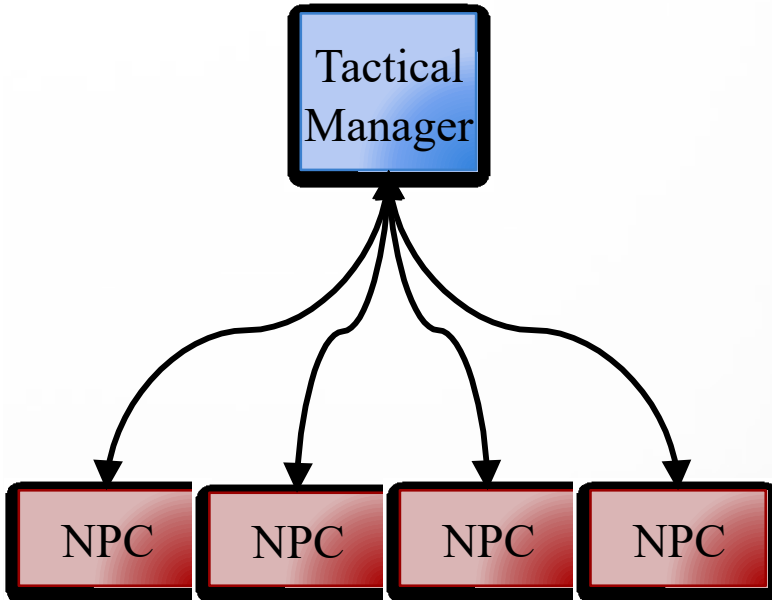
- 不能在一个短时间内完成动作

- 使用一个“子动作序列”作为动作
  - 一个动作序列就像人的行为
  - 使用数据文件（资源）驱动动作

  - 请研究“子状态”

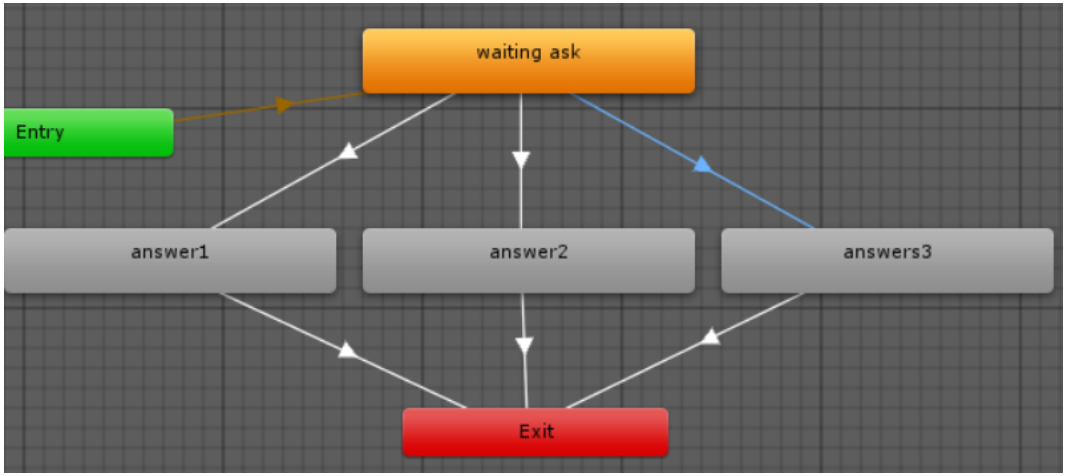
游戏智能常用方法
行动： 策略管理者

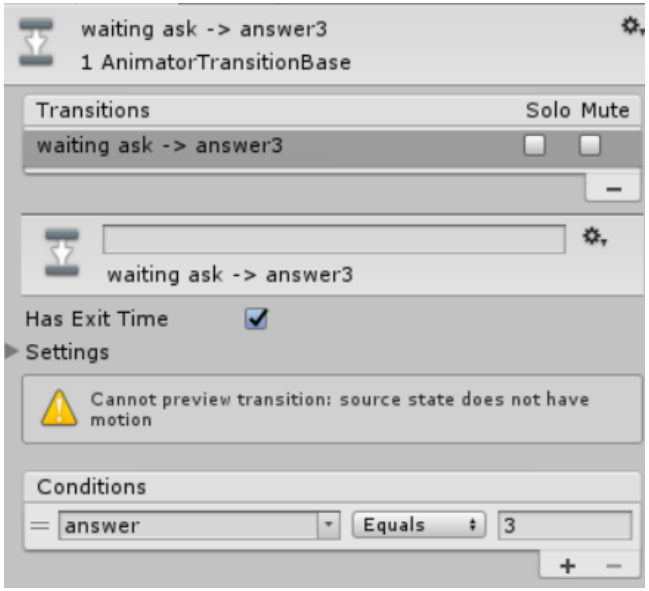
- “不可见 NPC”
  - 创建 NPC 组对象
  - 由组对象控制 NPC
- 应用场景
  - 保护游戏单元
  - 多向攻击
  - 交叉火力
  - 组队交替前进
  - 多路追击或逃跑



游戏智能常用方法
课堂实验一：决策练习

- 已知一个任务 teller 的行为状态机，如图：

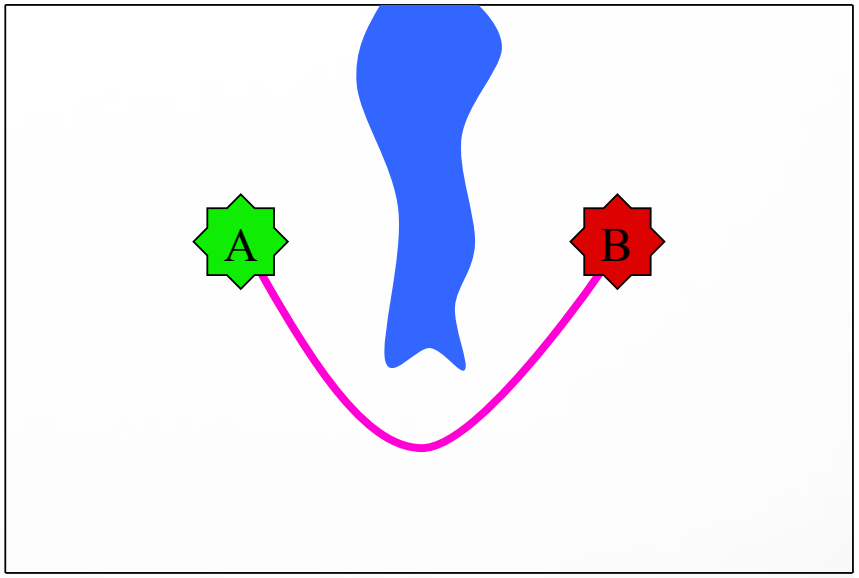




- 编写程序完成以下决策
  - answer 变量每次进入状态时为 0
  - 玩家一个任务未完成，回答的概率
    - （50%，45%，5%）
  - 玩家完成对应任务后，自动按概率分配。
    - 例如：玩家完成任务1，回答概率（0,90%,10%）

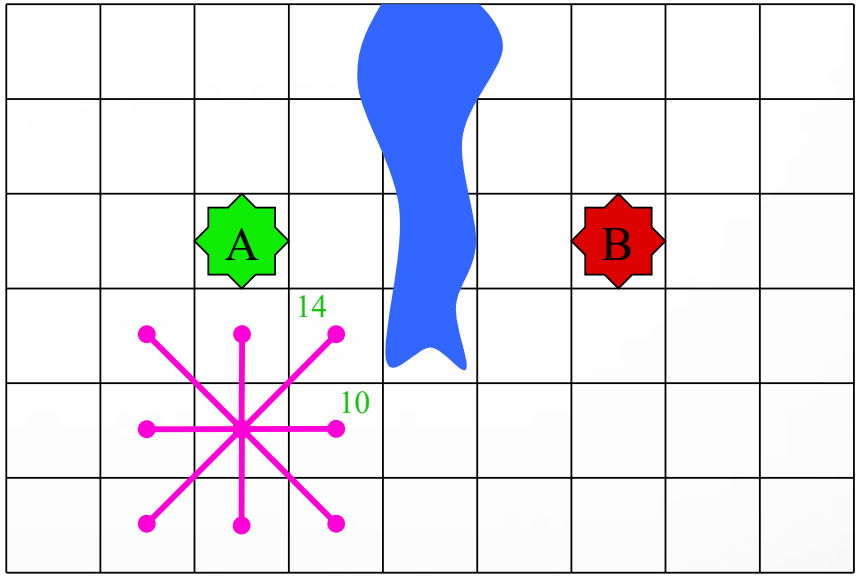
游戏智能常用方法
寻路 AI

- You are given
  - Starting location A
  - Goal location B
- Want valid path A to B
  - Avoid “impassible” terrain
  - Reasonably short path
- Algorithmic problem
  - performance



游戏智能常用方法
寻路抽象: Grid & Graph

- Break world into grid
  - Roughly size of NPCs
  - Terrain is all-or-nothing
    - Majority terrain of square
    - Terrain covering “center”
- Gives us a weighted graph
  - Nodes are grid centers
  - Each node has 8 neighbors
  - Weight = distance/terrain
- Search for shortest path!



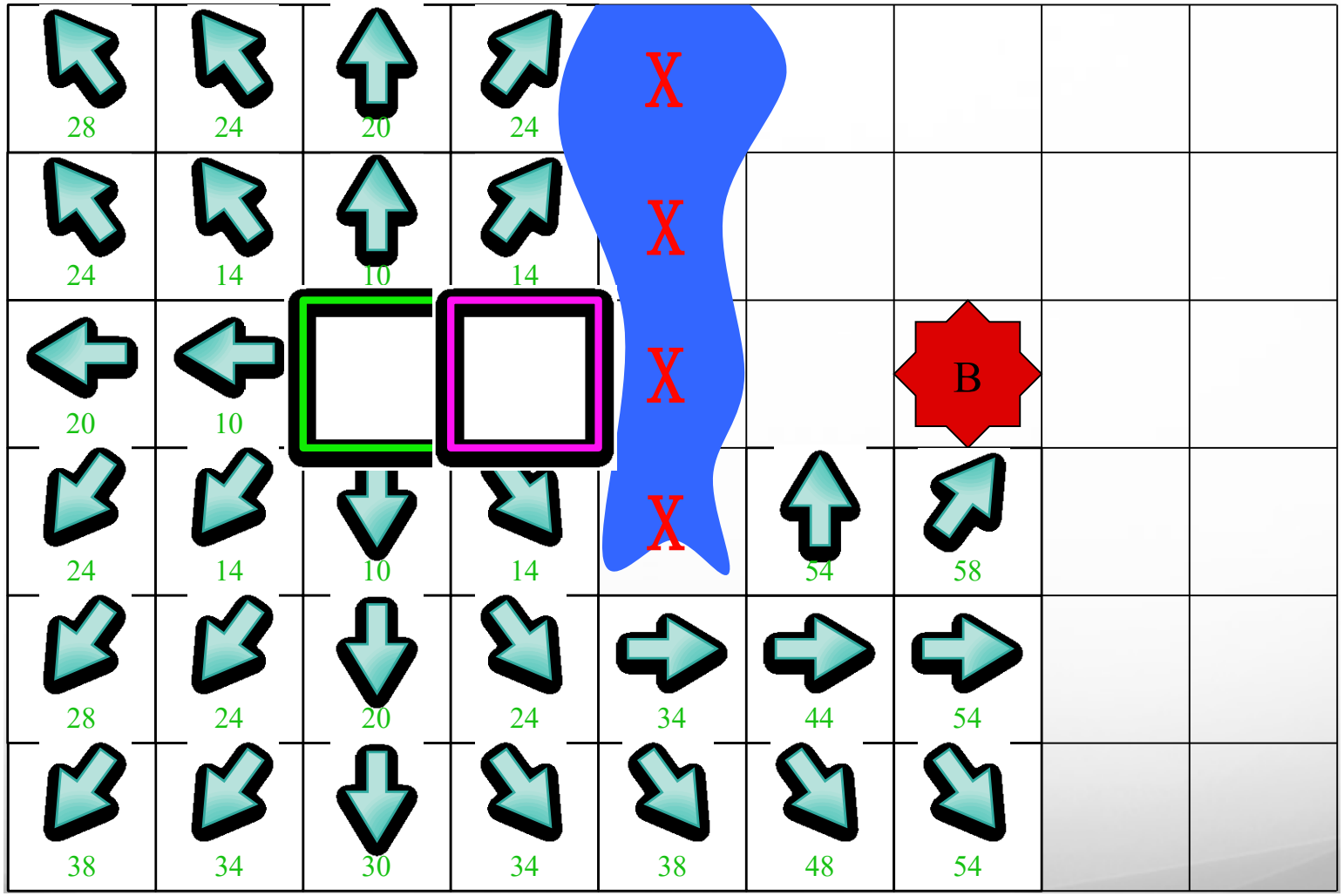
Real distance not required
- 14:10 ratio for diagonals
- Allows us to use integers

游戏智能常用方法
寻路：宽度优先检索

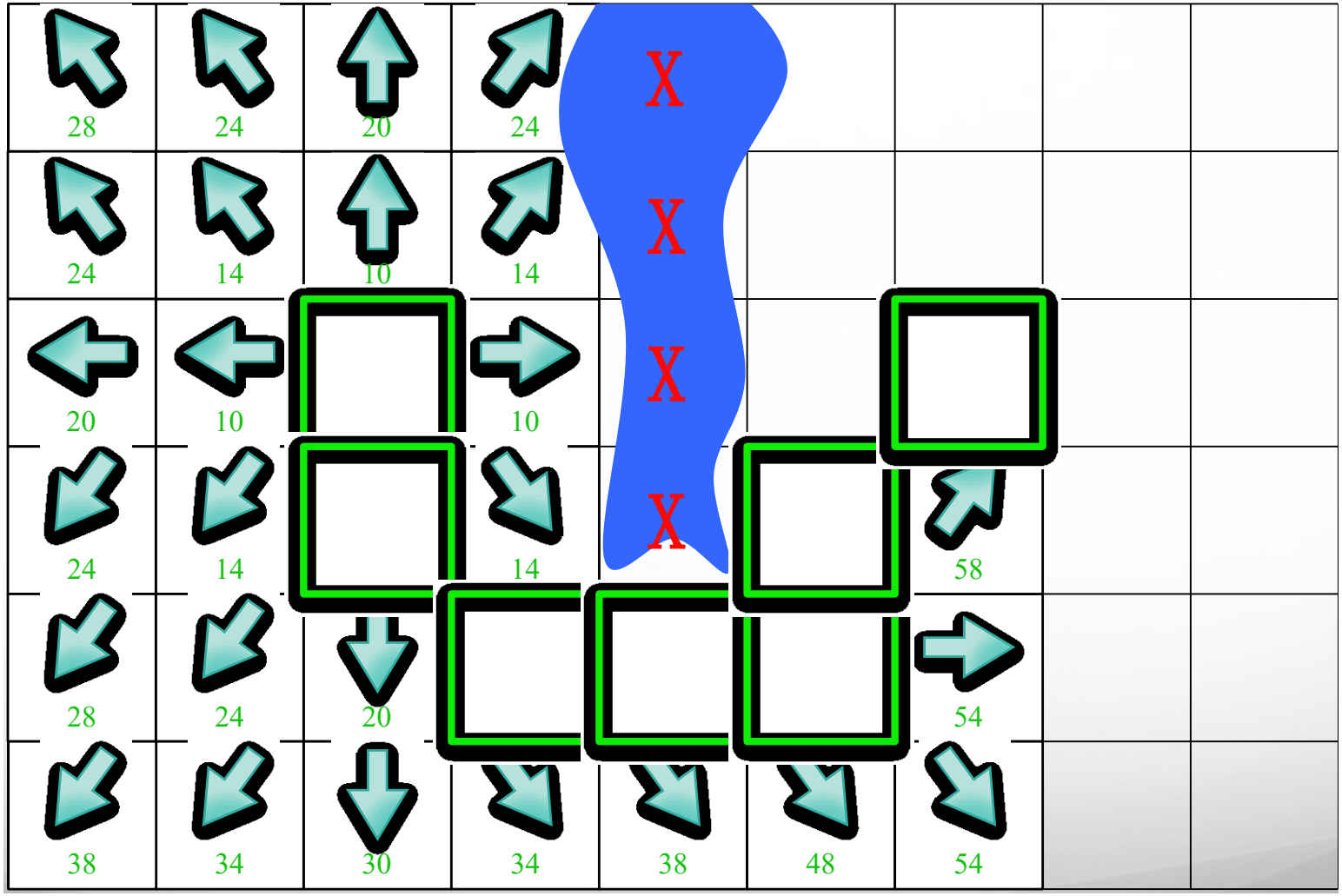
- Search maintains
  - Current node, initially **start**
  - List of nodes to visit

- Process
  - Have we reached the **goal?**
  - Add neighbors to *end* of list
  - Continue search from *first* node in list
  - List processed “first-in first-out”

游戏智能常用方法
寻路：宽度优先检索

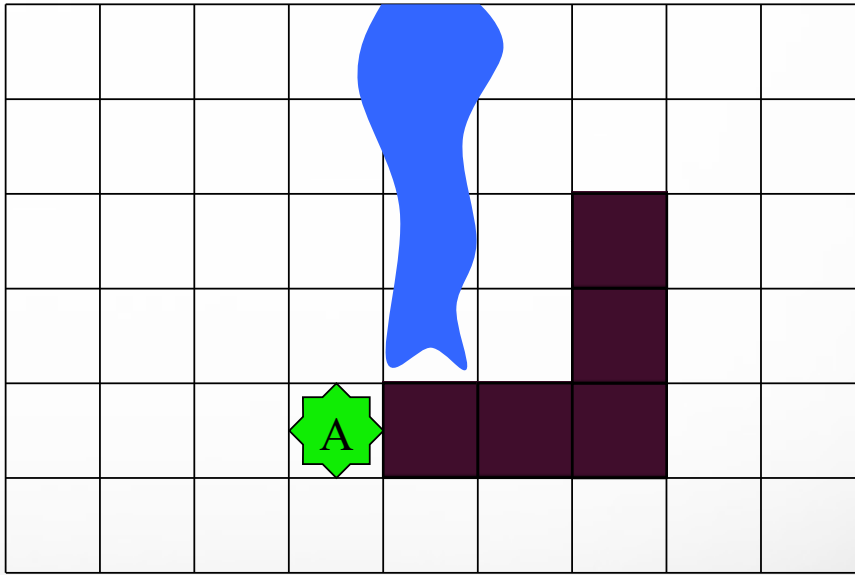


游戏智能常用方法
寻路：宽度优先检索



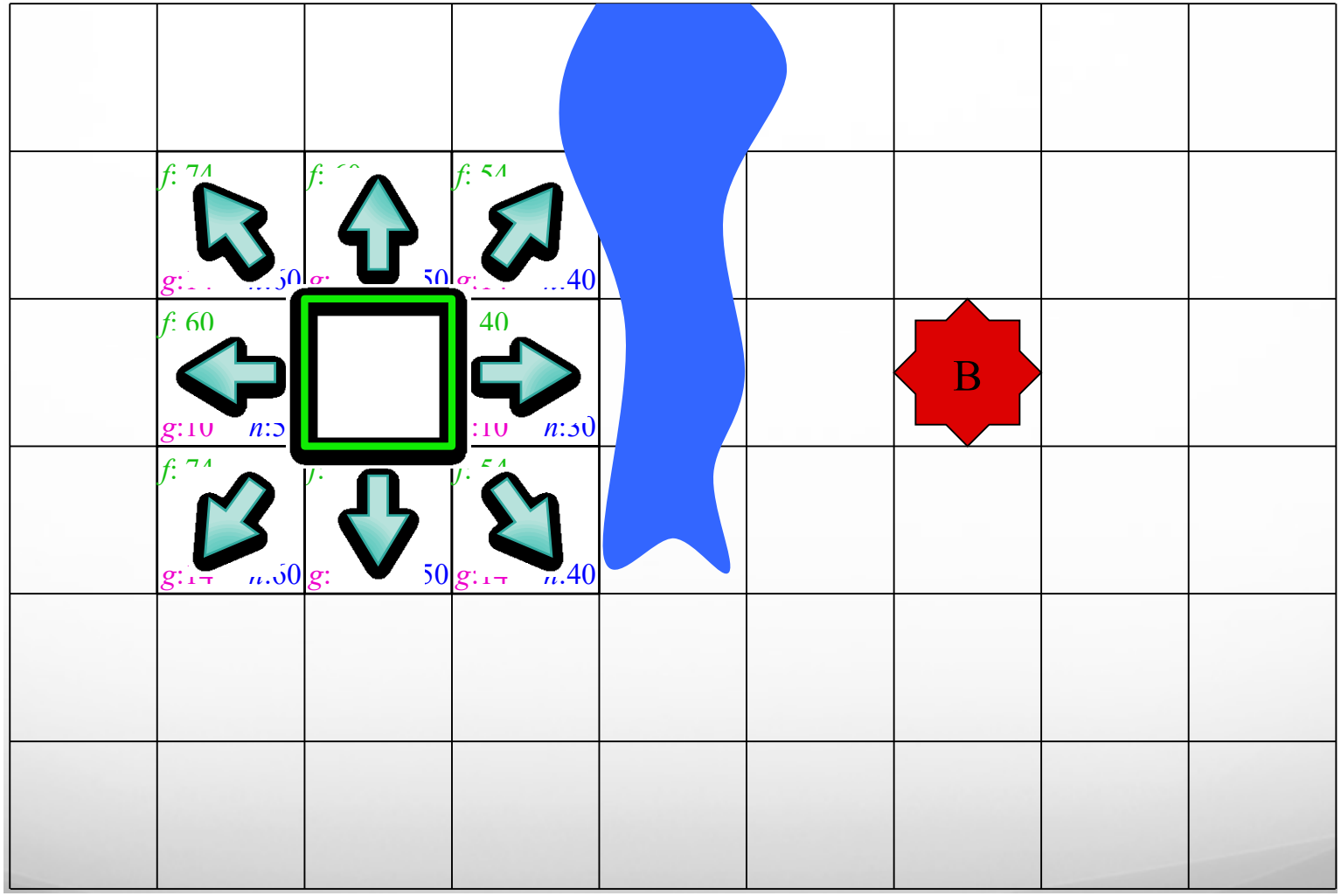
游戏智能常用方法
寻路：A* 算法

- Combine Dijkstra and Greedy
  - *g:* distance on **current path**
    - An “exact calculation”
    - Distance along graph
  - *h:* estimated distance to **goal**
    - Spatial, not graph, distance
    - Ignores all obstacles
  - Final heuristic *f*=*g*+*h*
- Many variations for *h*
  - Regular distance
  - “Manhattan Metric”

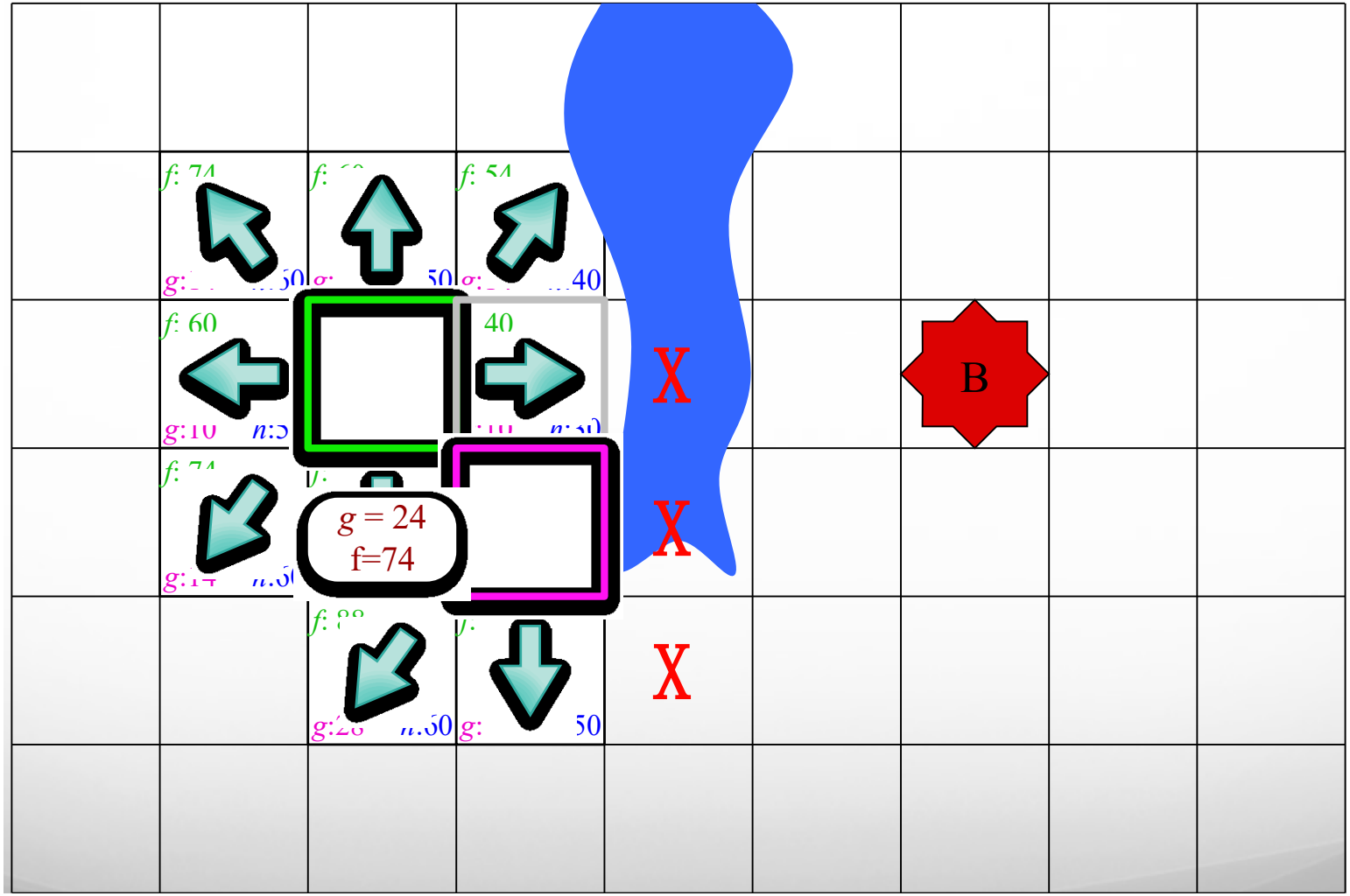


Manhattan distance = 30+20 = 50

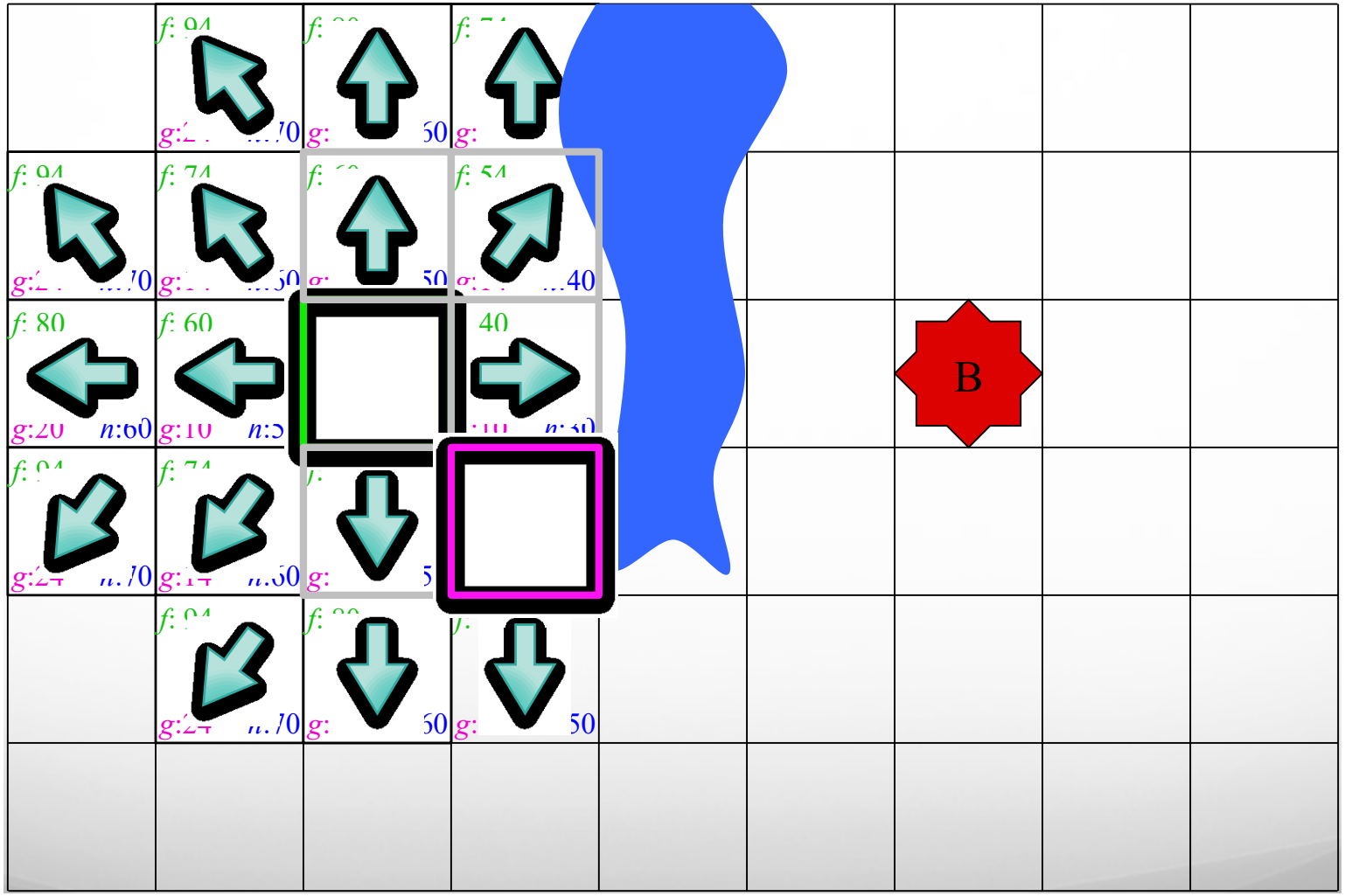
游戏智能常用方法
寻路：A* 算法



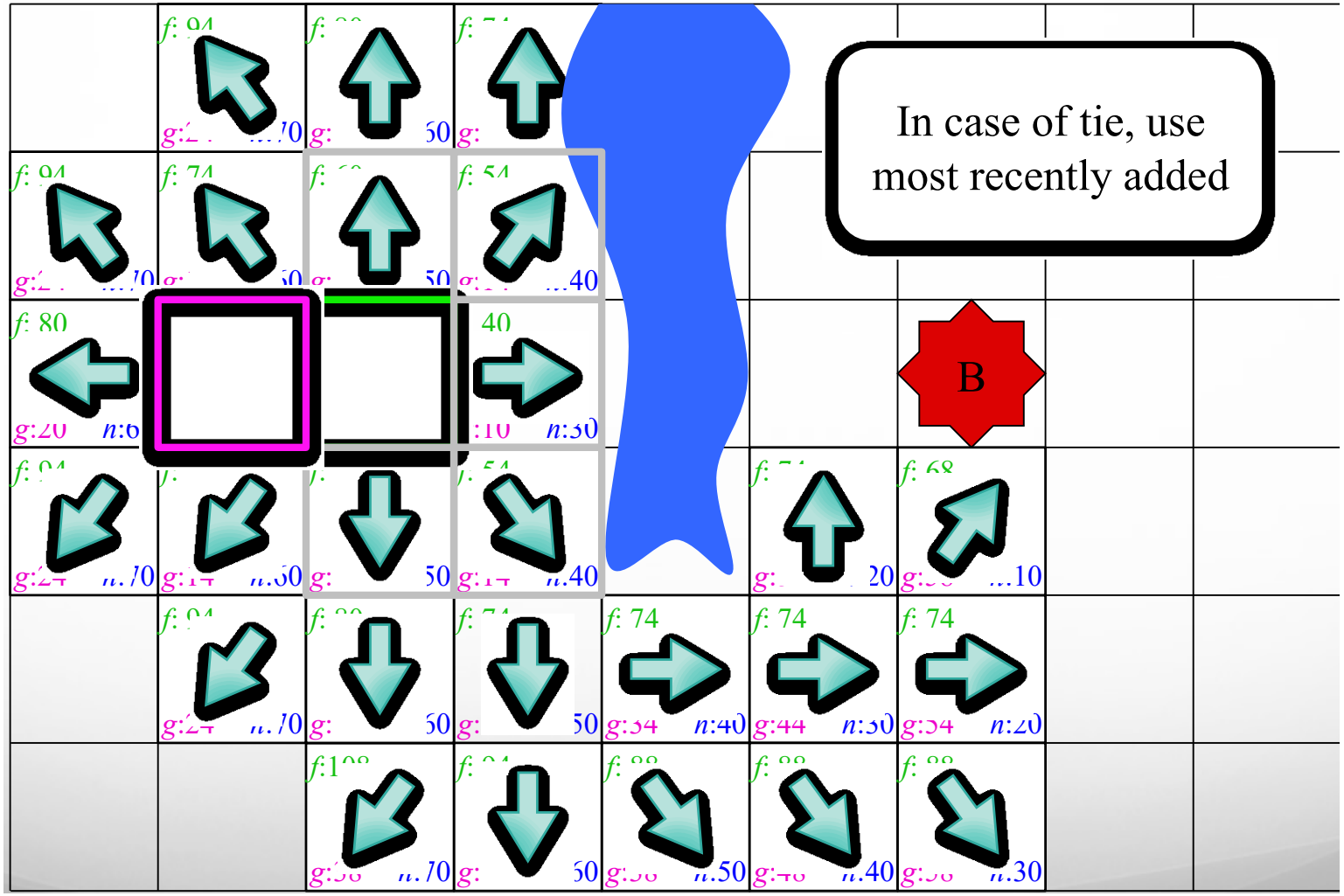
游戏智能常用方法
寻路：A* 算法



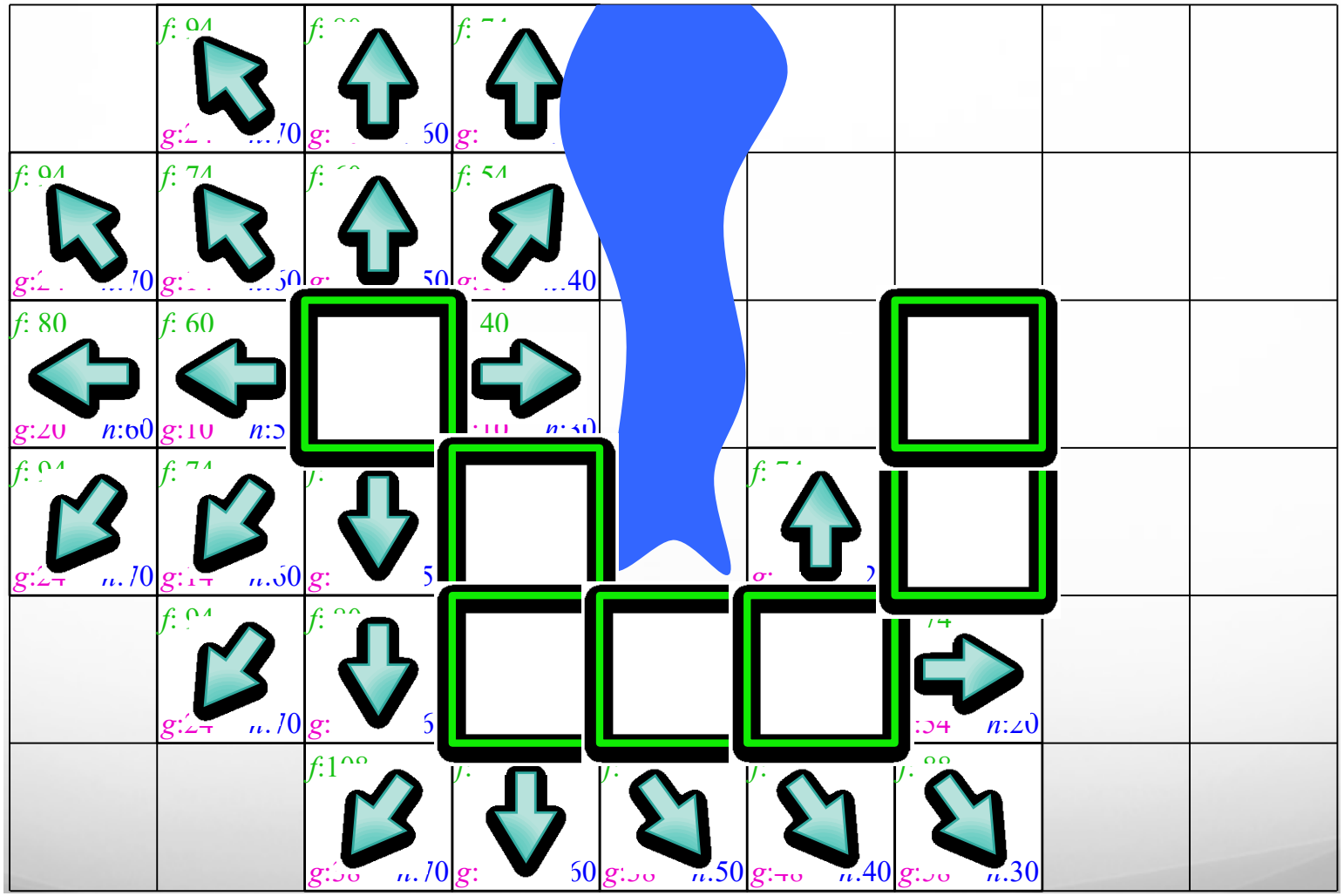
游戏智能常用方法
寻路：A* 算法



游戏智能常用方法
寻路：A* 算法

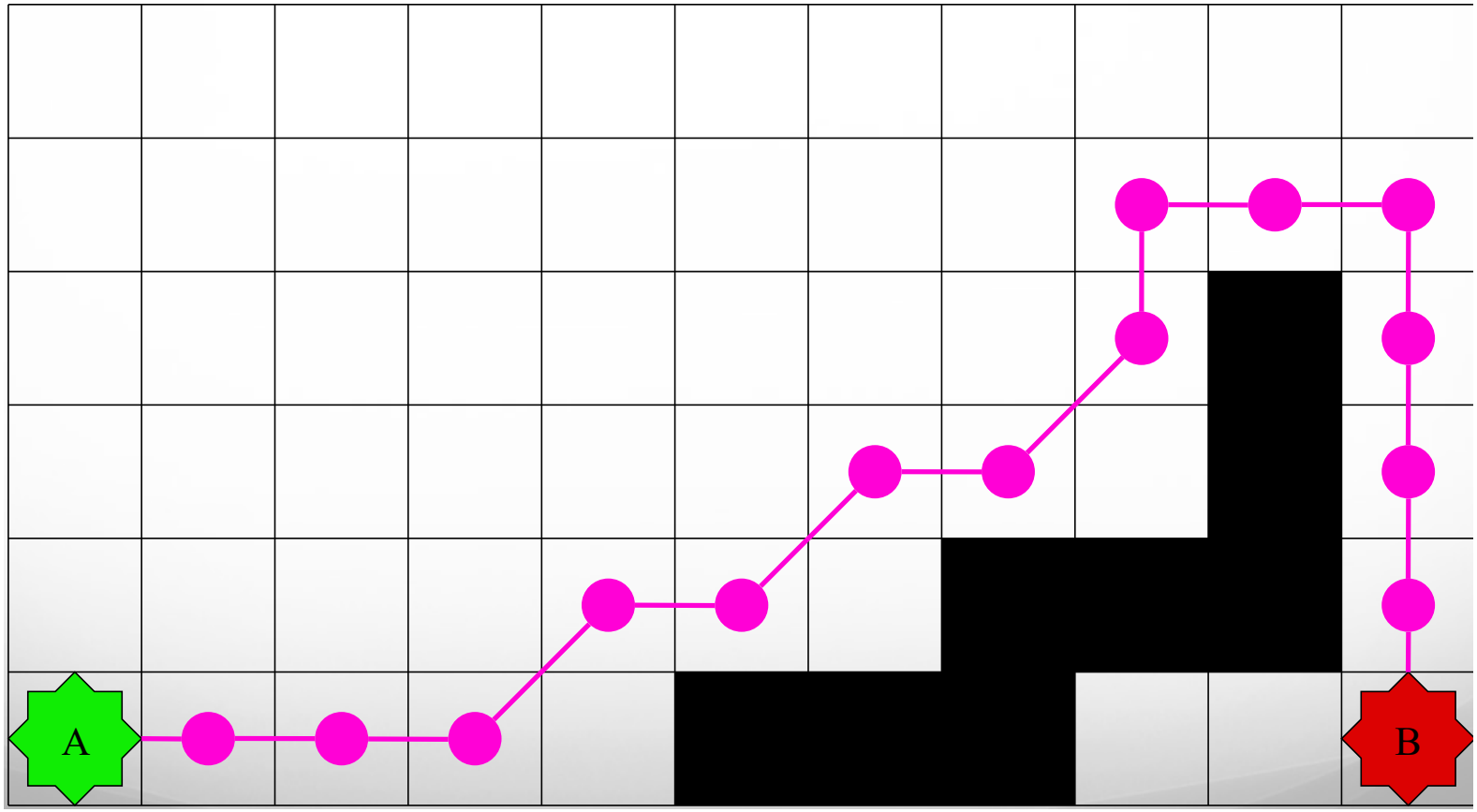


游戏智能常用方法
寻路：A* 算法

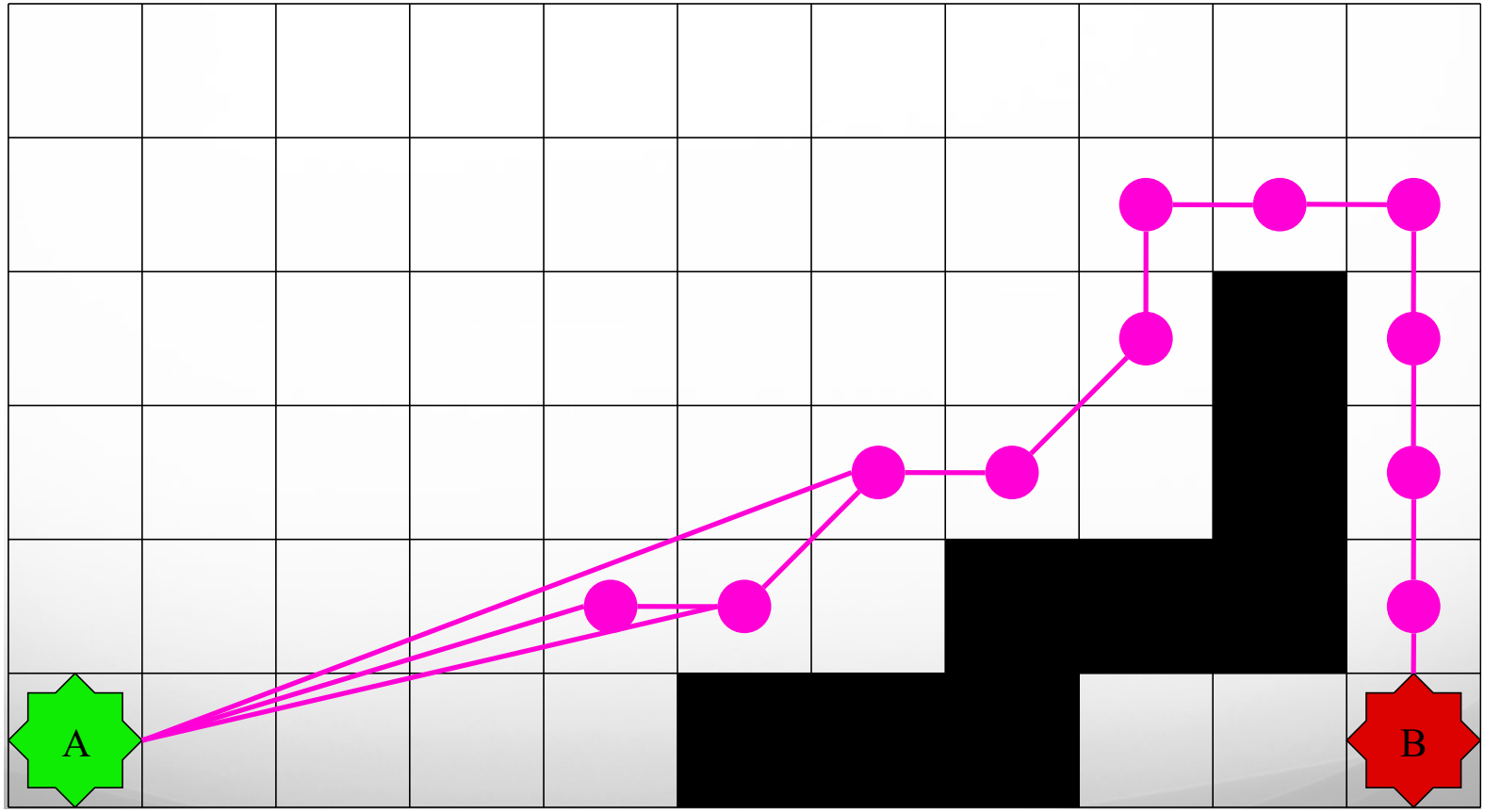


游戏智能常用方法

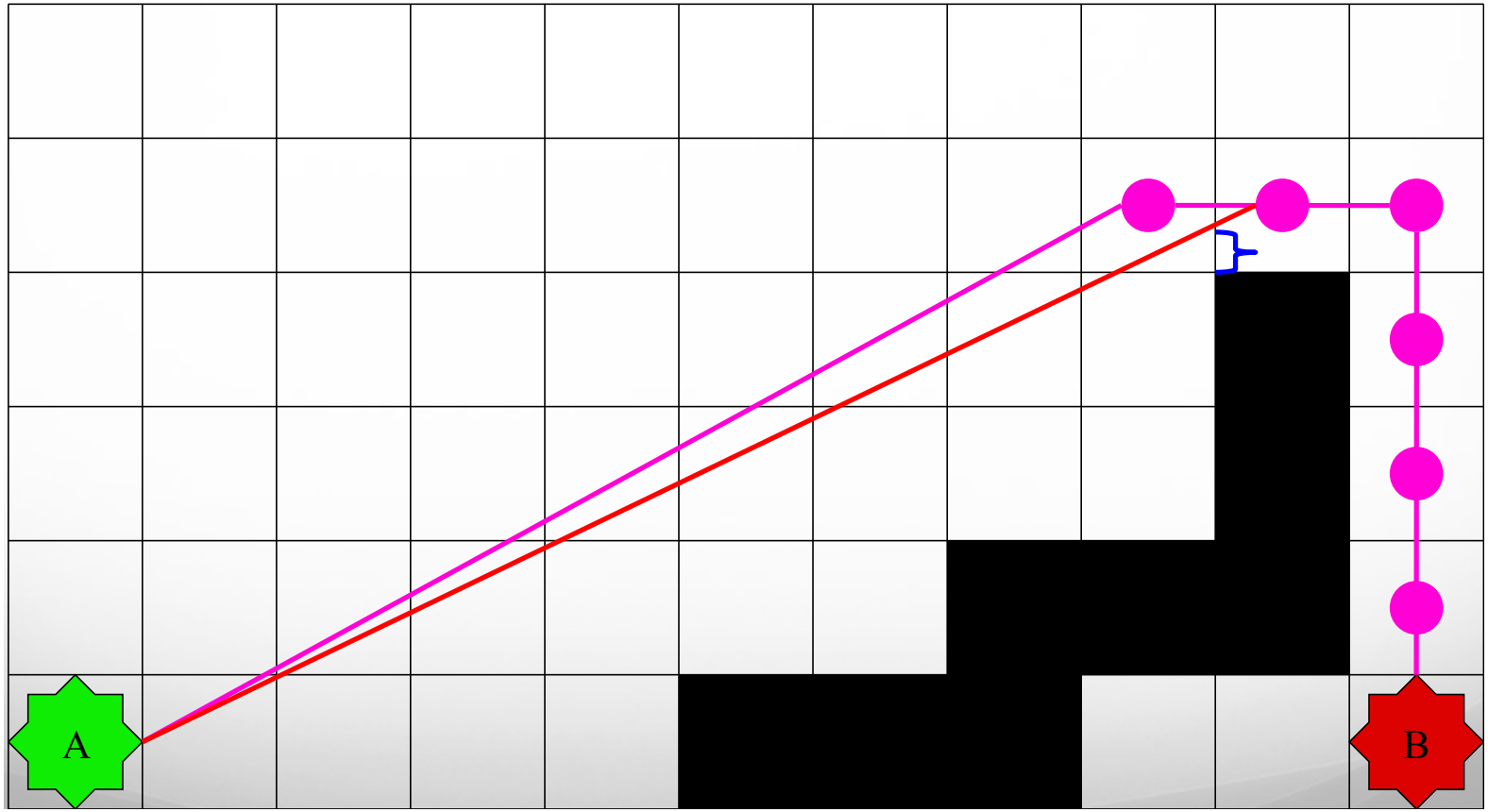
寻路：A* 算法的阶梯步进问题



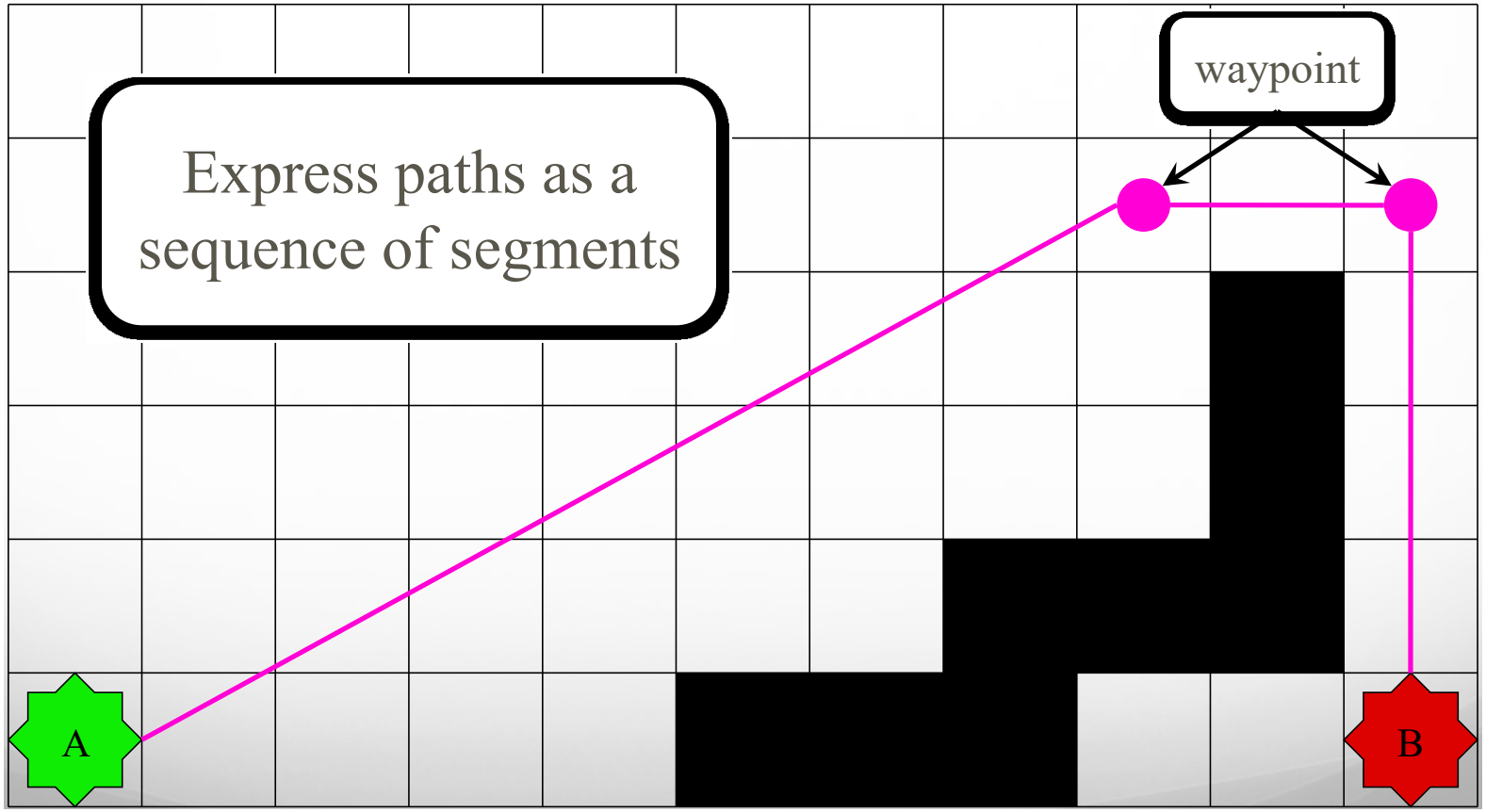
游戏智能常用方法
寻路：A* 算法 -- 路径平滑



游戏智能常用方法
寻路：A* 算法 -- 路径平滑

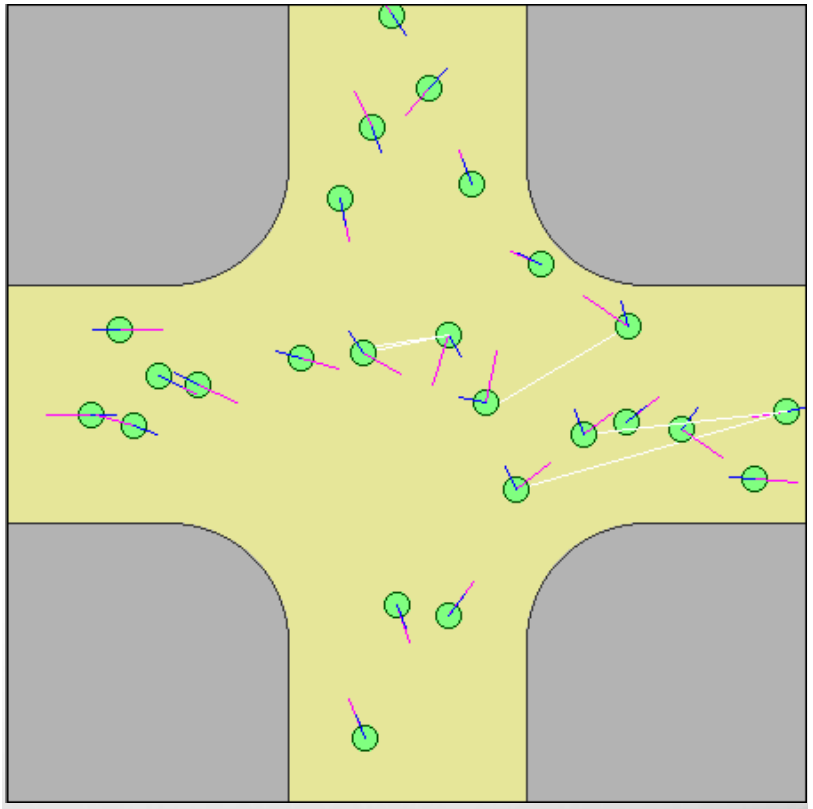


游戏智能常用方法
寻路：A* 算法 – 路径点集合

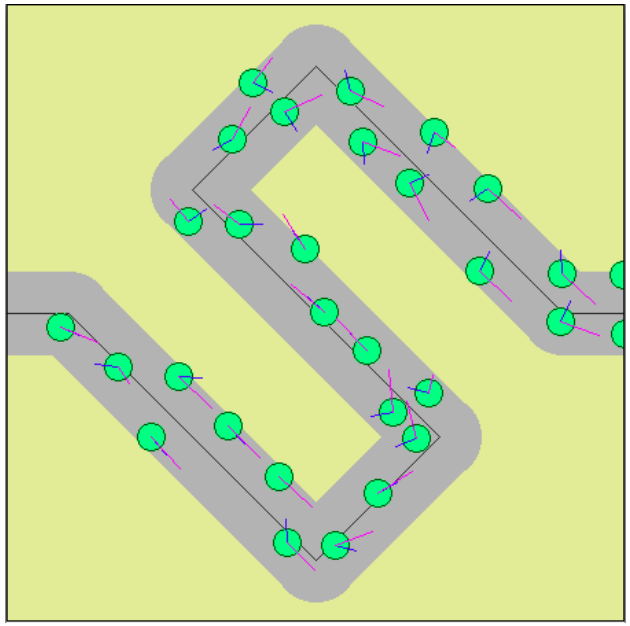


游戏智能常用方法
寻路： 力场与导航

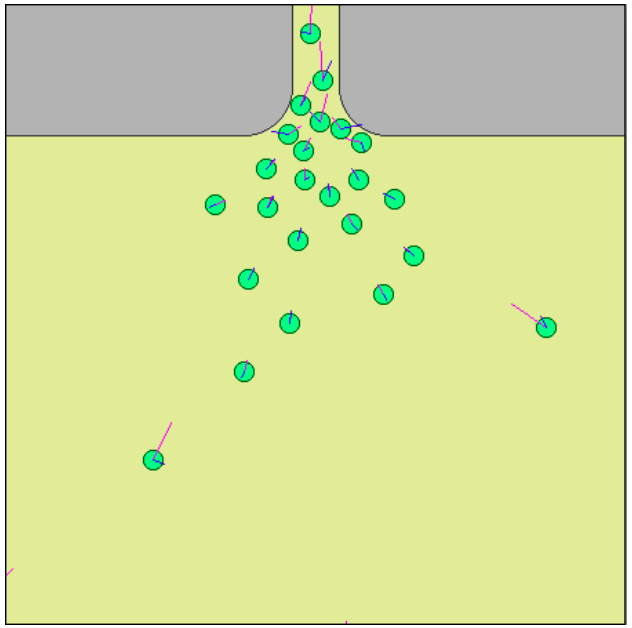
- Alternative to pathfinding
  - Uses forces to move NPCs
  - Great for small paths
- Examples
  - Artificial ***potential fields***
  - Vortex fields(涡流场)
  - Custom steering behaviors



游戏智能常用方法
寻路： 力场与导航



Use path as Force vector



Use Breadth-First from the goal to all cells, then use path as Force vector

游戏智能常用方法
策略 AI: 规划问题

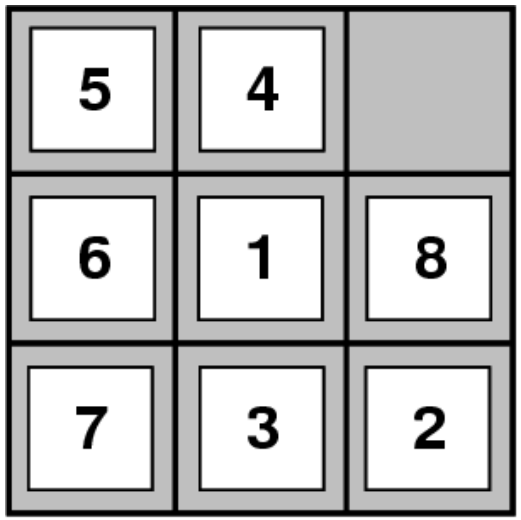
Multiple Steps: Planning

- Plan: **actions** necessary to reach a **goal**
  - Goal is a (pseudo) specific game state
  - Actions change game state (e.g. verbs)

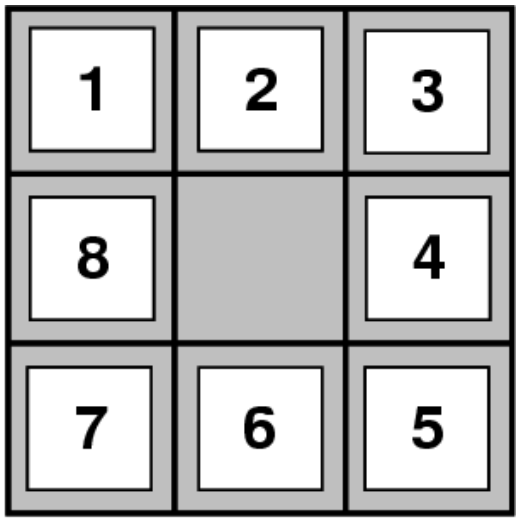
- Planning: steps to generate a plan
  - Initial **State:** state the game is currently in
  - Goal **Test:** determines if state meets goal
  - Operators: action the NPC can perform

Example: 8-Puzzle

- Given an initial configuration of 8 numbered tiles on a 3 x 3 board, move the tiles into a desired goal configuration of the tiles.



Start State



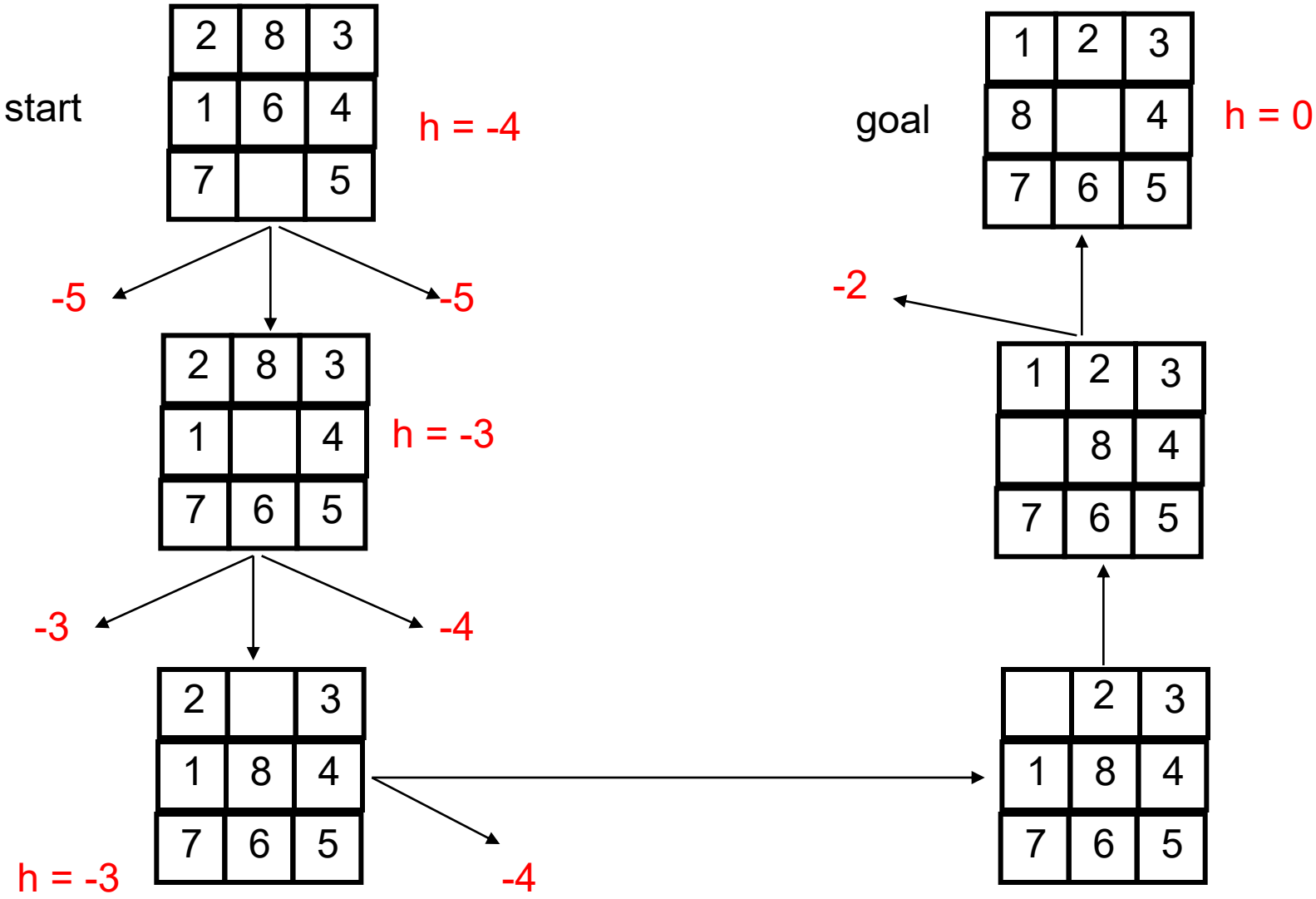
Goal State

8-Puzzle Encoding

- **State:** 3 x 3 array configuration of the tiles on the board.
- **4 Operators:** Move Blank Square Left, Right, Up or Down.
  - This is a more efficient encoding of the operators than one in which each of four possible moves for each of the 8 distinct tiles is used.
- **Initial State:** A particular configuration of the board.
- **Goal:** A particular configuration of the board.

- What does the state space look like?

Hill Climbing Example



f(n) = -(number of tiles out of place)

游戏智能须知
可玩性第一

- Complete intelligence is not fun
  - Player should have a chance of “winning”
  - Purpose is to make your game fun, not smart
  -
- Challenge: make your AI “smartly stupid”
  - AI should exhibit reasonable behavior
  - But should have flaws the player can exploit

游戏智能须知
Ten Ways to be Stupid！！！

- 1. Do not cheat
  - The AI should not be omniscient(无所不知)
  - Players will notice this
- 2. Do not kill on the first attempt
  - First “miss” gives player time to react
- 3. Have horrible aim
  - Same as it is in the movies
  - Allows abundant gunfire without being too hard
- 4. Do not shoot on first sight
  - Give player time to run for cover
- 5. Warning the player
  - Use different clues(线索): animations, sounds
  - Important when attacking from behind

游戏智能须知
TEN WAYS TO BE STUPID！！！

- 6. Attack “kung-fu” style
  - Do not all gang up at once(一伙出现)
  - NPCs should look busy (aiming, reloading)
- 7. Tell player what you are doing
  - AI state should have visual cues
- 8. Intentionally be vulnerable(故意的容易受伤)
  - Design weaknesses in the AI to exploit
  - 例如：Boss常有固定容易死的部位。探索型玩家最爱
- 9. Do not be perfect
  - NPC always makes mistakes in time
- 10. Pull back at the last minute（关键时刻，小宇宙爆发）
  - Push the player hard, and then pull back
  - 例如：最后一口血，很大概率能打死了三口血的Boss

游戏智能须知
Recommended Reading

- Dave Mark, *Behavioral Mathematics for Game*
- *AI Wisdom* Series, Charles River Media
- AIGameDev.com
- www.gdcvault.com
- 本课程没有涉及学习型 AI 算法

游戏智能常用方法
课堂实验二：寻路练习

- Unity 自带 3D 寻路组件。
  - 使用 AI 第三方角色 完成以下任务

- unity自带寻路Navmesh入门教程（一）
  - http://liweizhaolili.blog.163.com/blog/static/162307442012711613101 35/

  - 注意：该教程共三篇，在实现第三篇时请用一个策略管理者完成分路进攻。

面向对象的编程思考
LAMBDA 表达式与决策树

- Decision Table in C#
  - http://lukevoss.com/blog/post/2008/09/Decision-Table-in-C.aspx

课程小结

- 游戏智能
  - 游戏智能与人工智能的区别
  - 游戏智能的目标
  - 游戏智能的应用
- 游戏智能实现常用方法
  - 行为智能：感知-思考-行为模型
  - 寻路智能：A* 算法，场与导航
  - 策略智能：线性规划、启发式规划
  - 常见让智能萌笨的方法
- 面向对象的编程思考
  - Lambda 表达式与决策表

作业（Lab 11）

- （无）

- 可选作业
  - 从商店下载游戏：“Kawaii” Tank，构建 AI 对战游戏
  - P&D 过河游戏智能帮助实现（仅限二年级）