

Python3 - P SimpleSurvivor Problem - f tinyHouseSearch - l tinyHouse - x 1.0 - z 2.0



(Always returns the sequence to solve tinyHouse layout)

### readCommand(argv)

PROBLEM\_CHOICES

- ① Simple Survivor
- ② Multi Survivor

(other layouts fail)

mandatory args

Flags =

- P problem
- f function
- h heuristic
- l layout
- t text graphics
- q quiet graphics
- z zoom
- r recordActions
- C catchExceptions

(one of the problem choices)  
(Search: BFS, tinyHouse, A\*)  
(nullHeuristic, manhattanHeuristic)  
(layout file: burning office, power plant, etc)  
(display steps text output)  
(no graphics, only final output)  
(float multiplier for window zoom)  
(writes actions to a file)  
(turns on exception handling during mission)

- Loads layout, identifies rescuer, and passes args to runMission.



args = {\*\*} → layout  
rescuer  
display  
record  
catchException

### runMission(\*\*args)

This calls newMission on the RescueMission instance (RescueMission)

newMission(layout, rescuer, display, False, catchExceptions)



Initializes a new RescueState (initState) that contains:

- Rescuer position
- Survivors remaining
- Terrain layout

A new Game is created (mission)

Game(agents, display, self, catchExceptions)



mission contains the following data

Game
agentCrashed : bool
agents : Agent []
display : RescueGraphics
rules : RescueMission
startingIndex : int
gameOver : bool
muteAgents : bool
catchExceptions : bool
moveHistory : Array []
totalAgentTimes : Array []
totalAgentTimeWarnings : Array []
agentTimeout : bool

We set mission's state to initState, then set rescueMission's initialState to a deep copy of initState, and set rescueMission's quiet value.

This object is now called episode.

We now call run() on episode object.

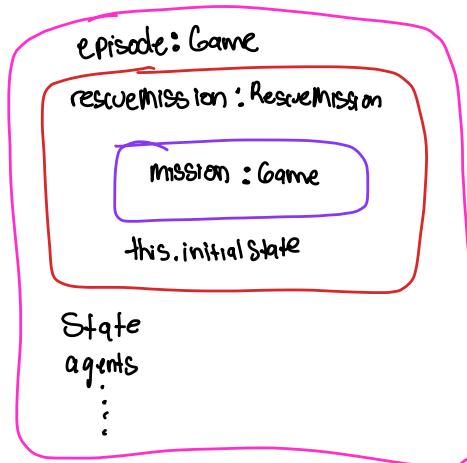
run()



① The game's display is initialized with the layout and all attributes about terrain, survivors, etc and their locations.

② The agents are initialized with initialState via registerInitialState().

③ The agent (SearchAgent) sets the problem (Search problem) by calling the init from either SimpleSearch.



Only A\* uses heuristics

This problem contains locations of walls



Using the problem, we pass it to the searchFunction() so we can get the sequence of moves to solve the given layout.

[This is where we must add the search algorithms!] (Search.py)

---

This is when we get the cost based on the terrain conditions.

In getCostOfActions, using the differentials dx and dy and the terrain type crossed over, we add up the cost of the path taken.

We then draw each step on the map in sequence.