pacman.py

PacmanRules |

PACMAN SPEED: int

applyAction(state, action) consume(position, state) getLegalActions(state)

GhostRules

GHOST SPEED: float

applyAction(state, action, ghostIndex)
canKill(pacmanPosition, ghostPosition)
checkDeath(state, agentIndex)
collide(state, ghostState, agentIndex)
decrementTimer(ghostState)
getLegalActions(state, ghostIndex)
placeGhost(state, ghostState)

initialState quiet: bool timeout: int agentCrash(game, agentIndex) getMaxStartupTime(agentIndex) getMaxTimeWarnings(agentIndex) getMaxTotalTime(agentIndex) getMoveTimeout(agentIndex) getMoveWarningTime(agentIndex) getProgress(game) lose(state, game) newGame(layout, pacmanAgent, ghostAgents, display, quiet, catchExceptions) process(state, game) win(state, game)

Main

Args: sys.argv[1:]

readCommand(sys.argv[1:]
runGames(layout, pacman, ghosts, display,
numGames, record, numTraining=0,
catchExceptions=False,timeout=30)

initialState GameState data explored : set deepCopy() generatePacmanSuccessor(action) generateSuccessor(agentIndex, action) getAndResetExplored() getCapsules() getFood() getGhostPosition(agentIndex) getGhostPositions() getGhostState(agentIndex) getGhostStates() getLegalActions(agentIndex) getLegalPacmanActions() getNumAgents() getNumFood() getPacmanPosition() getPacmanState() getScore() getWalls() hasFood(x, y) hasWall(x, y) initialize(layout, numGhostAgents) isLose() isWin()

pacmanAgents.py

GreedyAgent

evaluationFunction

getAction(state)

LeftTurnAgent

getAction(state)

game.py

Actions

TOLERANCE: float

directionToVector(direction, speed) getLegalNeighbors(position, walls) getPossibleActions(config, walls) getSuccessor(position, action) reverseDirection(action) vectorToDirection(vector)

Agent

index : int

getAction(state)

copy() getDirection() getPosition()

AgentState

numCarrying: int

numReturned: int

scaredTimer : int

configuration

isPacman

start

Configuration

direction pos

generateSuccessor(vector) getDirection() getPosition() isInteger()

FixedRandom

random: Random

agentStates: list

capsules

food layout

score: int

scoreChange : int

copyAgentStates(agentStates)

deepCopy()

initialize(layout, numGhostAgents)

GameStateData



Game

OLD STDERR : NoneType OLD STDOUT : NoneType

agentCrashed : bool agentOutput

agentTimeout : bool agents

catchExceptions : bool

display gameOver: bool moveHistory: list

muteAgents : bool numMoves : int

rules startingIndex : int

totalAgentTimeWarnings totalAgentTimes

getProgress() mute(agentIndex) run() ummute()

Grid CELLS PER INT: int

height width

asList(key) copy() count(item) deepCopy() packBits() shallowCopy()

GhostRules

GHOST SPEED: float

applyAction(state, action, ghostIndex) canKill(pacmanPosition, ghostPosition) checkDeath(state, agentIndex) collide(state, ghostState, agentIndex) decrementTimer(ghostState) getLegalActions(state, ghostIndex) placeGhost(state, ghostState)

graphicsdisplay.py

FirstPersonPacmanGraphics

capture: bool

distributionImages : NoneType

isBlue: bool lavout

previousState

showGhosts: bool

getGhostColor(ghost, ghostIndex)

getPosition(ghostState) initialize(state, isBlue) lookAhead(config. state)

agentImages: list capsules : dict capture: bool currentGhostImages : dict

currentState

PacmanGraphics

distributionImages: NoneType, list

expandedCells : list

food: list frameTime: float gridSize : float have window : int

height infoPane isBlue: bool lavout

pacmanImage: NoneType

previousState width zoom: float

animatePacman(pacman, prevPacman, image)

checkNullDisplay() clearExpandedCells() drawAgentObjects(state) drawCapsules(capsules) drawDistributions(state) drawExpandedCells(cells) drawFood(foodMatrix)

drawGhost(ghost, agentIndex) drawPacman(pacman, index) drawStaticObjects(state)

drawWalls(wallMatrix)

finish()

getDirection(agentState)

getEndpoints(direction, position) getGhostColor(ghost, ghostIndex)

getPosition(agentState)

initialize(state, isBlue) isWall(x, v, walls)

make window(width, height)

moveEyes(pos, dir, eyes)

moveGhost(ghost, ghostIndex, prevGhost, ghostImageParts)

movePacman(position, direction, image) removeCapsule(cell, capsuleImages) removeFood(cell, foodImages)

startGraphics(state)

swapImages(agentIndex, newState)

to screen(point) to screen2(point) update(newState)

updateDistributions(distributions)

InfoPane

base

fontSize: int

ghostDistanceText: list

gridSize height : int teamText textColor width

clearIcon()

clearMessage()

drawGhost()

drawPacman()

drawPane()

drawWarning()

initializeGhostDistances(distances)

setTeam(isBlue)

toScreen(pos, y)

updateGhostDistances(distances)

updateMessage(message)

updateScore(score)

keyboardAgents.py

KeyboardAgent

EAST KEY: str NORTH KEY: str SOUTH KEY: str STOP KEY: str WEST KEY: str

index: int keys: list lastMove

getAction(state) getMove(legal)

Layout

agentPositions: list

capsules : list food height

layoutText numGhosts: int totalFood

visibility walls

deepCopy()

getFurthestCorner(pacPos)

getNumGhosts() getRandomCorner()

getRandomLegalPosition()

initializeVisibilityMatrix()

isVisibleFrom(ghostPos, pacPos, pacDirection)

isWall(pos)

processLayoutChar(x, y, layoutChar) processLayoutText(layoutText)

width

STOP KEY: str

GreedyAgent

evaluationFunction

getAction(state)

KeyboardAgent2

EAST KEY: str NORTH KEY: str

SOUTH KEY: str

WEST KEY: str

getMove(legal)

LeftTurnAgent

getAction(state)

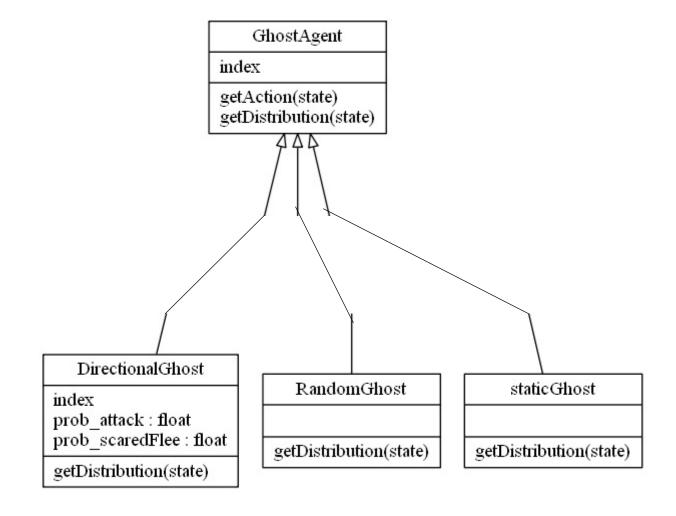
GoWestAgent2

getAction(state)

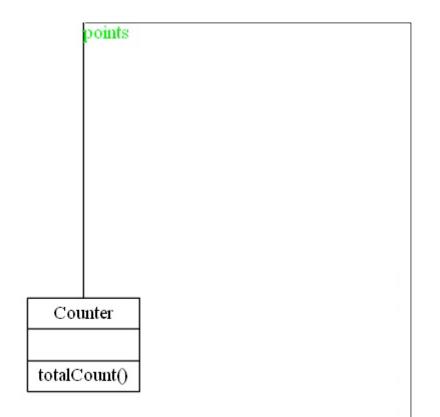
layout.py

```
Layout
agentPositions: list
capsules: list
food
height
layoutText
numGhosts: int
totalFood
visibility
walls
width
deepCopy()
getFurthestCorner(pacPos)
getNumGhosts()
getRandomCorner()
getRandomLegalPosition()
initializeVisibilityMatrix()
isVisibleFrom(ghostPos, pacPos, pacDirection)
isWall(pos)
processLayoutChar(x, y, layoutChar)
processLayoutText(layoutText)
```

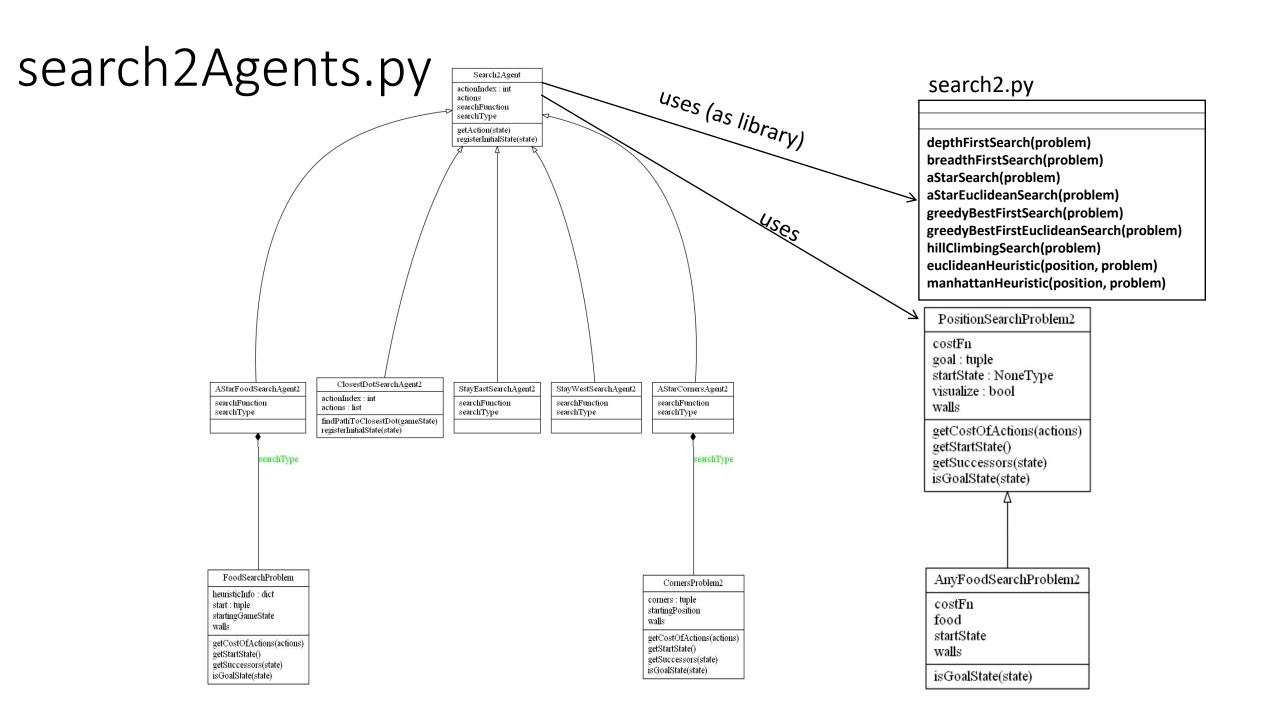
ghostAgents.py



grading.py



```
Grades
currentQuestion: NoneType
edxOutput: bool
gsOutput : bool
maxes: dict
messages : dict
mute: bool
points
prereqs : defaultdict
project
questions
sane: bool
start
addErrorHints(exceptionMap, errorInstance, questionNum)
addExceptionMessage(q, inst, traceback)
addMessage(message, raw)
addMessageToEmail(message)
addPoints(amt)
addPrereq(question, prereq)
assignFullCredit(message, raw)
assignZeroCredit()
deductPoints(amt)
fail(message, raw)
grade(gradingModule, exceptionMap, bonusPic)
produceGradeScopeOutput()
produceOutput()
```



search2.py

SearchProblem2

getCostOfActions(actions) getStartState() getSuccessors(state) isGoalState(state)

util.py

FixedRandom

random : Random

Queue

list: list

isEmpty() pop() push(item) PriorityQueue count : int heap : list

isEmpty()
pop()
push(item, priority)
update(item, priority)

PriorityQueueWithFunction

priorityFunction push(item) Stack

list : list

isEmpty() pop() push(item) TimeoutFunction

function timeout

handle_timeout(signum, frame)

TimeoutFunctionException

WritableNull

write(string)

Counter

argMax()
copy()
divideAll(divisor)
incrementAll(keys, count)
normalize()
sortedKeys()
totalCount()