**گزارش پروژه چهارم**

**اشکان شکیبا (9931030)**

**بخش اول**

if ghostPosition == jailPosition:  
 if noisyDistance is None:  
 return 1.0  
 else:  
 return 0.0  
if noisyDistance is None:  
 return 0.0  
return busters.getObservationProbability(noisyDistance, manhattanDistance(pacmanPosition, ghostPosition))

در این تابع، ابتدا بررسی می‌شود که اگر روح در زندان است، در صورت None بودن noisy distance مقدار یک و در غیر این صورت مقدار صفر بازگردانی شود. اگر روح در زندان نباشد و noisy distance برابر None باشد صفر بازگردانی شده و در غیر این صورت حاصل تابع احتمال مشاهده شکارچی‌ها بر اساس فاصله منهتنی پک‌من و روح (معادل مقدار P(noisyDistance | pacmanPosition, ghostPosition) در داکیومنت پروژه) بازگردانی می‌شود.

**بخش دوم**

def observeUpdate(self, observation, gameState):  
 *"""  
 Update beliefs based on the distance observation and Pacman's position.  
  
 The observation is the noisy Manhattan distance to the ghost you are  
 tracking.  
  
 self.allPositions is a list of the possible ghost positions, including  
 the jail position. You should only consider positions that are in  
 self.allPositions.  
  
 The update model is not entirely stationary: it may depend on Pacman's  
 current position. However, this is not a problem, as Pacman's current  
 position is known.  
 """* for ghost\_position in self.allPositions:  
 self.beliefs[ghost\_position] \*= self.getObservationProb(  
 observation, gameState.getPacmanPosition(), ghost\_position, self.getJailPosition())  
  
 self.beliefs.normalize()

در این تابع با پیمایش بر روی موقعیت همه روح‌ها، احتمال با تابع احتمال مشاهده و بر اساس موقعیت پک‌من، روح مورد بررسی و زندان محاسبه شده و در فیلد گفته شده در داکیومنت پروژه، ضرب و ذخیره می‌شود.

**بخش سوم**

def elapseTime(self, gameState):  
 *"""  
 Predict beliefs in response to a time step passing from the current  
 state.  
  
 The transition model is not entirely stationary: it may depend on  
 Pacman's current position. However, this is not a problem, as Pacman's  
 current position is known.  
 """* discrete\_distribution = DiscreteDistribution()  
  
 for position in self.allPositions:  
 new\_position = self.getPositionDistribution(gameState, position)  
 for key in new\_position:  
 discrete\_distribution[key] += self.beliefs[position] \* new\_position[key]  
  
 self.beliefs = discrete\_distribution

در این تابع ابتدا یک شی از کلاس DiscreteDistribution ساخته شده و سپس با پیمایش بر روی موقعیت‌های قبلی روح و بررسی آنها با تابع getPositionDistribution، موقعیت جدید به شکل یک دیکشنری ذخیره می‌شود که با دادن هر موقعیت به آن، احتمال حضور روح در آن موقعیت در زمان t+1 بازگردانی می‌شود.

با پیمایش بر روی این دیکشنری، مقادیر شی ساخته شده بر اساس حاصل‌ضرب مقادیر پیش‌تر ذخیره شده در مقادیر جدید، تکمیل می‌شوند و در نهایت این شی جایگزین شی مربوط به مقادیر قبلی کلاس می‌شود.