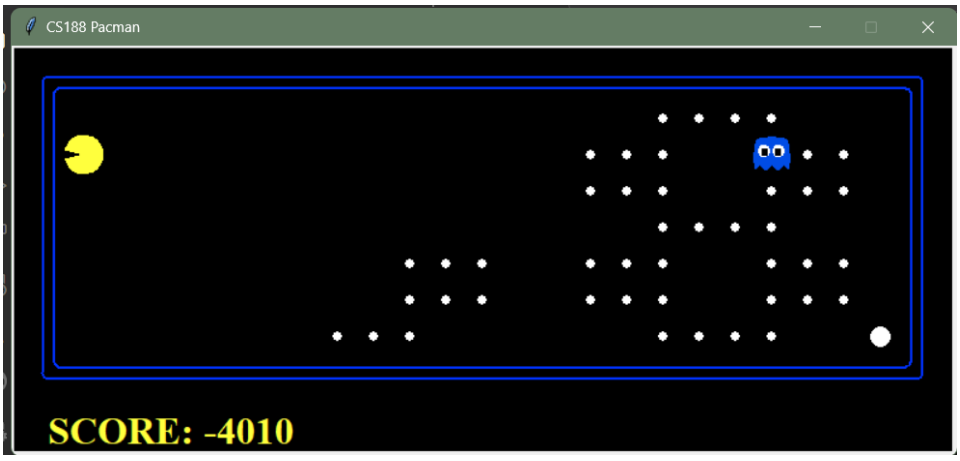
BÁO CÁO MÔN CS106

BT4 - Evaluation functions for Minimax/AlphaBeta/Expectimax



Giảng viên hướng dẫn:

Lương Ngọc Hoàng

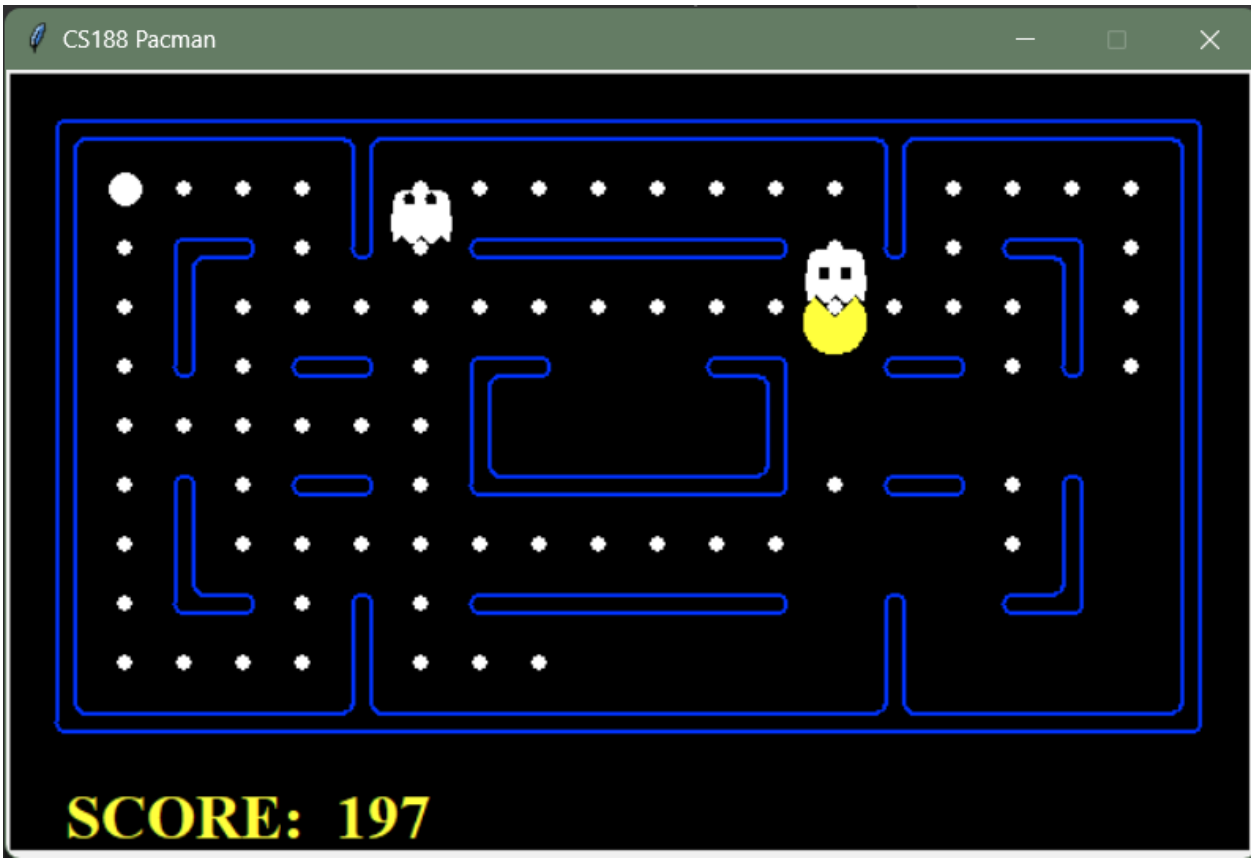
Sinh viên thực hiện:

Nguyễn Gia Bảo - 22520109

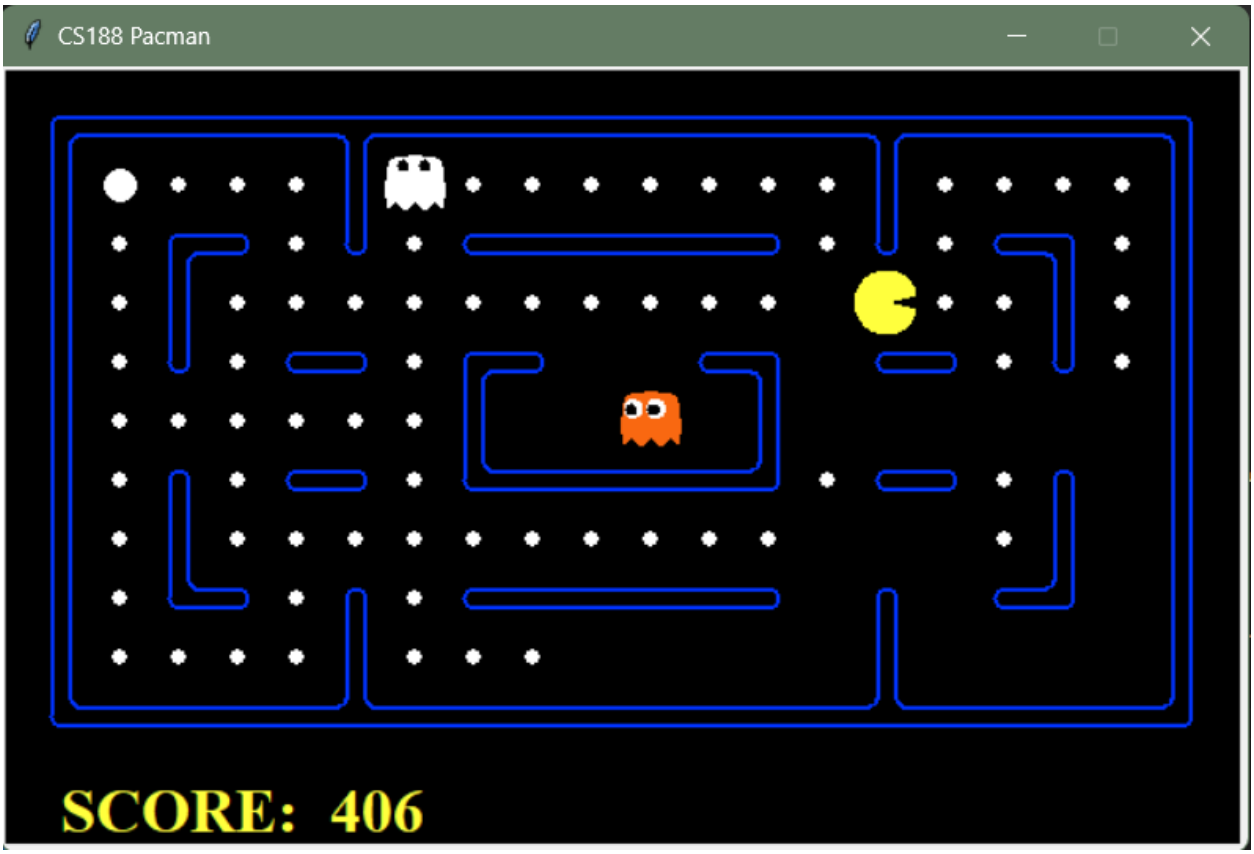
I. Thiết kế hàm lượng giá

1. *Chiến thuật thiết kế:*

Sau nhiều lần chơi thử bằng tay và bằng hàm lượng giá sẵn có, ta nhận ra rằng điểm đạt được sẽ càng cao hơn khi Pacman có thể ăn được càng nhiều ghost. Vậy nên ý tưởng chính của hàm lượng giá tự thiết kế (myEvalFunc) sẽ xoay quanh việc giúp Pacman ưu tiên hấp thụ capsules và ăn những con ma đang sợ hãi, đồng thời tránh né những con ma không sợ và ăn hết các chấm thức ăn có trên màn hình.



Điểm trước khi Pacman ăn scared ghost



Điểm ngay sau khi Pacman ăn scared ghost

2. *Code thực hiện:*

Dưới đây là code của hàm lượng giá được thiết kế theo chiến thuật đã nêu trên:

```python
def myEvalFunc(currentGameState):
    """
    Your extreme ghost-hunting, pellet-nabbing, food-gobbling, unstoppable
    evaluation function (question 5).

    DESCRIPTION: <write something here so we know what you did>
    """
    "*** YOUR CODE HERE ***"
    #util.raiseNotDefined()
    newPos = currentGameState.getPacmanPosition()
    newFood = currentGameState.getFood()
    newGhostStates = currentGameState.getGhostStates()
    newCapsules = currentGameState.getCapsules()
    newScaredTimes = [ghostState.scaredTimer for ghostState in newGhostStates]


    # closest_capsule
    if newCapsules:
        closestCapsule = min([manhattanDistance(newPos, caps) for caps in newCapsules])
    else:
        closestCapsule = 0

    if closestCapsule:
        cap_dis = 3 / closestCapsule
    else:
        cap_dis = 0

    #closest ghost
    closestGhost = min([manhattanDistance(newPos, ghost.getPosition()) for ghost in newGhostStates])

    if closestGhost:
        ghost_dis = -2 / closestGhost
    else:
        ghost_dis = -500

    # food
    foodList = newFood.asList()

    foodDistance = []
    for food in foodList:
        foodDistance.append(manhattanDistance(newPos, food))
    foodDistance.sort()
    while(len(foodDistance) < 2):
        foodDistance.append(0)
```

```python
    foodDistance.append(0)

    # calculate state score
    score = -1.5*foodDistance[0] + ghost_dis + cap_dis - 10*len(foodList)
    score += -10*len(newCapsules)

    score += scoreEvaluationFunction(currentGameState)
    score += currentGameState.getScore()*2

    #try to eat capsule -> hunt ghost
    # if the ghosts are scared
    if sum(newScaredTimes) > 0:
        score += 10*len(newCapsules)

        for index,ghost in enumerate(newGhostStates):
            curr_dis = manhattanDistance(newPos, ghost.getPosition())

            if curr_dis <= closestGhost and ghost.scaredTimer > 0 :
                score += 30/curr_dis
    else:
        score += -0.5*foodDistance[1]

    # print(f"food = {-2*closestFood}, ghost = {ghost_dis}, cap_dis = {cap_dis},  len(foodList) = {-10*len(foodList)}, len(newCapsules) = {-10*len(newCapsules)}")

    return score
```

3. *Giải thích các hệ số và cách tính toán:*

- ClosestCapsule: đặc trưng này sẽ tính điểm dựa trên khoảng cách của Pacman đến vị trí của viên capsule gần nhất. Vì muốn Pacman

ưu tiên ăn nhưng viên capsule nên ta sẽ để trọng số là +3/closestCapsule, để nếu khoảng cách càng nhỏ thì biến cap_dis sẽ càng lớn, điểm càng cao. Nếu closestCapsule = 0 thì nghĩa là không còn capsule nào trên bản đồ, vậy nên biến cap_dis = 0, không ảnh hưởng đến quá trình tính điểm.

- closestGhost: đặc trưng này sẽ tính khoảng cách manhattan giữa con ma gần nhất và Pacman, giúp Pacman né tránh những con ma có khả năng hạ gục mình và kết thúc trò chơi. Vì tránh càng xa ma càng tốt nên hệ số của đặc trưng này sẽ là số âm, khoảng cách càng gần thì trừ càng nhiều điểm, ghost_dis = -2/closestGhost. Nếu closestGhost = 0 nghĩa là Pacman đã va phải con ma, nên sẽ kết thúc trò chơi, điểm sẽ bị -500 điểm.
- Food: với đặc trưng liên quan đến thức ăn, chúng ta sẽ sử dụng khoảng cách của Pacman đến chấm thức ăn gần nhất và chấm thức ăn thứ hai. Đôi khi Pacman sẽ bị đứng yên do khoảng cách giữa Pacman tới chấm thức ăn gần nhất sẽ không thay đổi, vậy nên ta sử dụng thêm đặc trưng khoảng cách đến chấm thức ăn gần thứ hai, để giảm trường hợp Pacman bị "đóng băng". Nếu chỉ còn lại 1 chấm thức ăn trên bản đồ, ta sẽ thêm vào một phần tử ảo có giá trị là 0 vào danh sách foodDistance, sẽ không làm ảnh hưởng đến điểm số. Trọng số của chấm thức ăn thứ nhất sẽ là -1.5, vì càng gần chấm thức ăn sẽ càng tốt, nên sẽ bị trừ ít điểm nhất. Với chấm thức ăn gần thứ hai, chúng ta sẽ chưa cộng hẳn vào điểm tổng, mà để dùng trong trường hợp Pacman không thể đi săn ma được nữa.
- Số chấm thức ăn còn lại và số capsule còn lại (*len(foodList) và len(newCapsules)*): Đối với 2 đặc trưng này, nếu càng ít thì càng tốt (đặc biệt là len(foodList)), vì nhiệm vụ chính của Pacman là ăn hết các chấm thức ăn còn lại. Vậy nên trọng số của đặc trưng này sẽ là -10, thể hiện đây là một đặc trưng quan trọng, càng ít thì càng trừ ít điểm, ban đầu sẽ trừ rất nhiều điểm. Nhưng chúng ta

không chỉ muốn hoàn thành trò chơi mà còn muốn đạt càng nhiều điểm càng tốt, vì thế nên trọng số của *len(newCapsules)* cũng lớn, đều là -10.

- Cộng thêm điểm từ 2 hàm *scoreEvaluationFunction(currentGameStates)* và *currentGameStates.getScore()*2:* giúp cho điểm tổng không bị âm quá nhiều, sẽ có thêm điểm khi Pacman hấp thụ được những scared ghosts.
- Chiến thuật săn ma: Khi có bất kì con ma nào đang hoảng sợ, chúng ta sẽ bỏ đi phần điểm trừ từ *len(newCapsules)*, vì lúc này Pacman sẽ tập trung vào việc đi săn hơn là hấp thụ viên capsule tiếp theo. Chúng ta lặp qua từng con ma, xét xem đâu là mục tiêu đang sợ hãi và gần nhất, Pacman sẽ lao đến và săn đuổi mục tiêu đó, nhưng vẫn đồng thời tránh những còn ma không sợ mình (giữ nguyên phần điểm trừ từ ghost_dis ở trên). Vì săn ma sợ hãi là ưu tiên lúc này, nên trọng số của đặc trưng này sẽ là +30, chia cho khoảng cách tới con ma gần nhất đang sợ, càng gần thì càng có nhiều điểm cộng. Ngược lại, khi không còn con ma nào đang sợ hãi, Pacman sẽ tiếp tục đi ăn những chấm thức ăn, trừ thêm phần điểm của đặc trưng khoảng cách đến chấm thức ăn gần thứ 2. Vì đây là một đặc trưng phụ nên hệ số của nó chỉ là -0.5 .

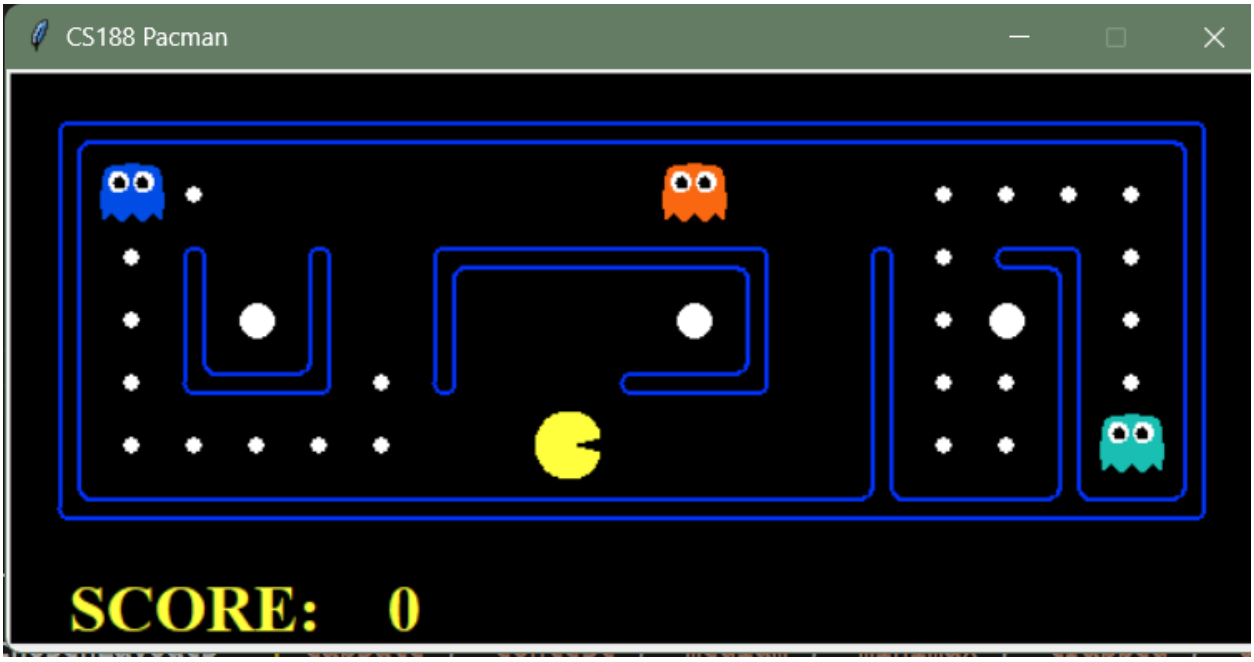
II. Số liệu thực nghiệm và giải thích

Mọi số liệu đều được chạy với deph = 3

1.capsuleClassic

capsuleClassic	ScoreEvaluationFunction		myEvalFunc		Algorithm
RandomSeed	Score	Record	Score	Record	
22520109+0	-498.0	Loss	200.0	Loss	Minimax
22520109+1	-486.0	Loss	-449.0	Loss	
22520109+2	-443.0	Loss	188.0	Loss	
22520109+3	-459.0	Loss	1456.0	Win	
22520109+4	-461.0	Loss	-211.0	Loss	
Average score	-469.4		236.8		
22520109+0	-498.0	Loss	200.0	Loss	AlphaBeta
22520109+1	-486.0	Loss	-449.0	Loss	
22520109+2	-443.0	Loss	188.0	Loss	
22520109+3	-459.0	Loss	1456.0	Win	
22520109+4	-461.0	Loss	-211.0	Loss	
Average score	-469.4		236.8		
22520109+0	-498.0	Loss	200.0	Loss	Expectimax
22520109+1	-487.0	Loss	1619.0	Win	
22520109+2	-444.0	Loss	436.0	Loss	
22520109+3	-120.0	Loss	775.0	Loss	
22520109+4	-315.0	Loss	13.0	Loss	
Average score	-372.8		608.6		

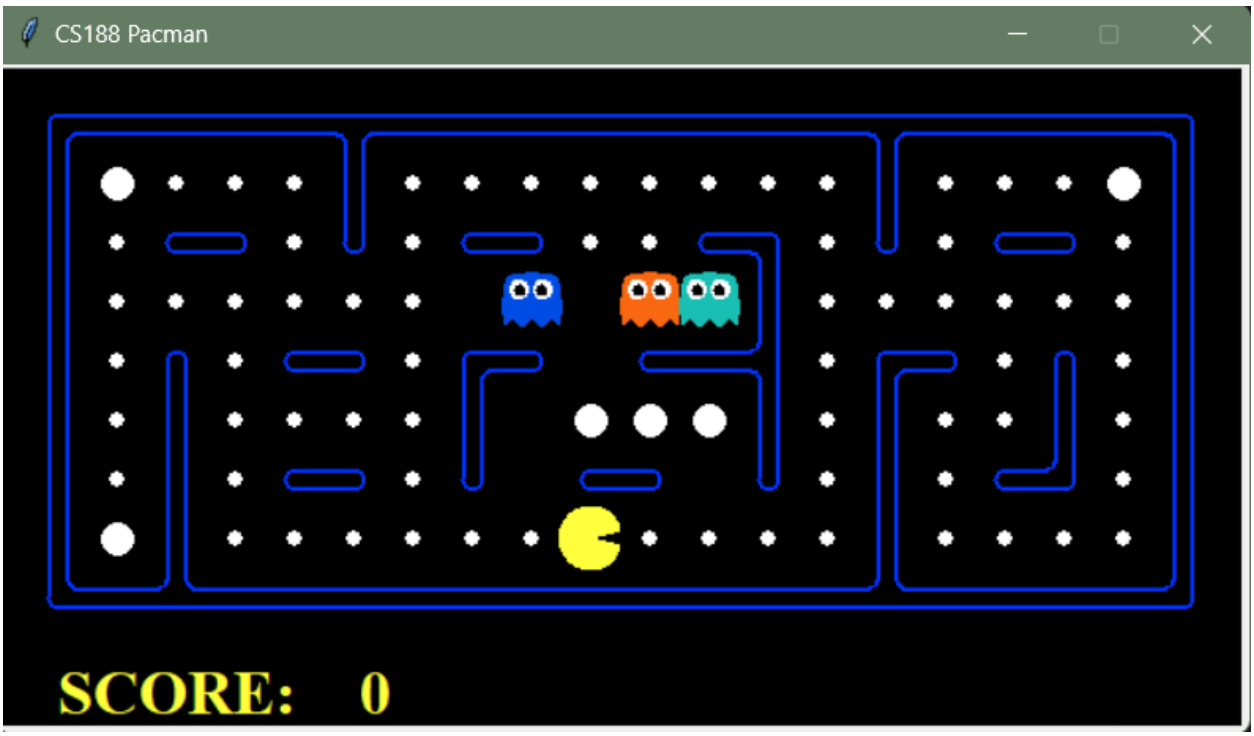
Đối với layout này, ta thấy các lần chơi khi sử dụng hàm lượng giá có sẵn đều thua và đạt kết quả không tốt. Khi sử dụng hàm lượng giá mới, kết quả cũng chỉ có duy nhất 1 lần thắng. Tuy nhiên số điểm trung bình lại cao hơn đáng kể. Ở bản đồ này, có một con đường sát bên phải, là một đường cụt, chỉ có một lối ra vào, nhưng lại chứa rất nhiều thức ăn, nên Pacman đa phần sẽ bị hạ gục khi đang cố gắng ăn những chấm thức ăn ở đó.



2.contestClassic

contestClassic	ScoreEvaluationFunction		myEvalFunc		Algorithm
RandomSeed	Score	Record	Score	Record	
22520109+0	-356.0	Loss	1446.0	Win	Minimax
22520109+1	-428.0	Loss	1832.0	Win	
22520109+2	-334.0	Loss	2624.0	Win	
22520109+3	30.0	Loss	2081.0	Win	
22520109+4	172.0	Loss	1886.0	Win	
Average score	-183,2		1973,8		
22520109+0	-356.0	Loss	1446.0	Win	AlphaBeta
22520109+1	-428.0	Loss	1832.0	Win	
22520109+2	-334.0	Loss	2624.0	Win	
22520109+3	30.0	Loss	2081.0	Win	
22520109+4	172.0	Loss	1886.0	Win	
Average score	-183,2		1973,8		
22520109+0	-300.0	Loss	2865.0	Win	Expectimax
22520109+1	2023.0	Win	3069.0	Win	
22520109+2	758.0	Loss	2074.0	Win	
22520109+3	-180.0	Loss	2383.0	Win	
22520109+4	501.0	Loss	3652.0	Win	
Average score	560,4		2808,6		

Đối với level này, có tận 3 ghosts, nhưng cũng có rất nhiều capsule phân bố đều trên bản đồ, nên với chiến thuật của hàm lượng giá cải tiến, Pacman chiến thắng tất cả các lần chơi, với tất cả các thuật toán. Do có độ rộng không quá lớn, nhiều capsules, số lượng kẻ địch ít, nên điểm số của các màn chơi thắng đều ở mức cao (>1000 điểm). Tuy nhiên kết quả lại không được khả quan khi dùng hàm lượng giá có sẵn.

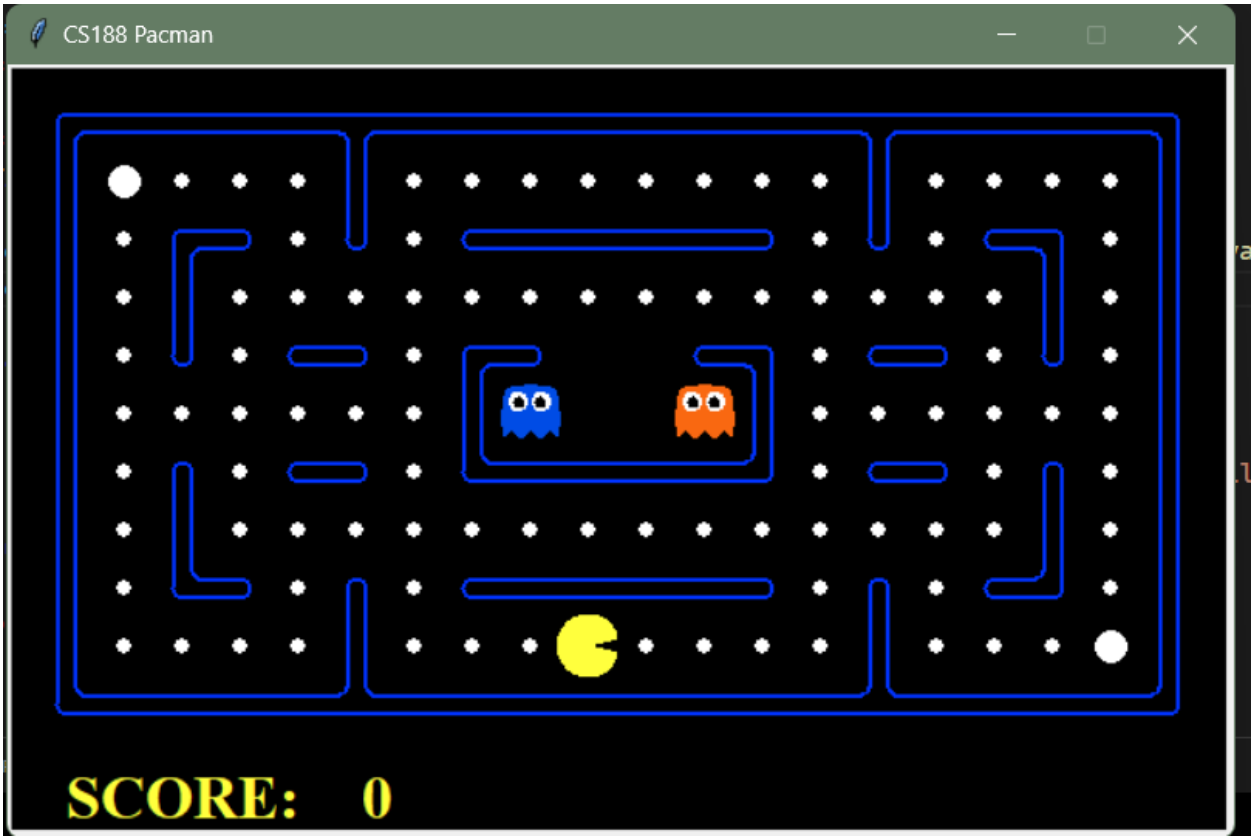


3.mediumClassic

mediumClassic	ScoreEvaluationFunction		myEvalFunc		Algorithm
RandomSeed	Score	Record	Score	Record	
22520109+0	-104.0	Loss	1908.0	Win	Minimax
22520109+1	160.0	Loss	1921.0	Win	
22520109+2	154.0	Loss	1905.0	Win	
22520109+3	1021.0	Win	2093.0	Win	
22520109+4	945.0	Win	2131.0	Win	

Average score	435,2		1991,6		
22520109+0	-104.0	Loss	1908.0	Win	AlphaBeta
22520109+1	160.0	Loss	1921.0	Win	
22520109+2	154.0	Loss	1905.0	Win	
22520109+3	1021.0	Loss	2093.0	Win	
22520109+4	945.0	Loss	2131.0	Win	
Average score	435,2		1991,6		
22520109+0	1033.0	Win	2116.0	Win	Expectimax
22520109+1	1482.0	Win	2030.0	Win	
22520109+2	-168.0	Loss	2114.0	Win	
22520109+3	1021.0	Win	2130.0	Win	
22520109+4	945.0	Win	2131.0	Win	
Average score	862,6		2104,2		

Ở level này, số lượng kẻ địch của Pacman và số capsule chỉ còn lại 2, nên chúng ta đã có được nhiều màn thắng hơn khi dùng hàm scoreEvaluationFunction kèm theo thuật toán Expectimax. Ở phía ngược lại thì hàm myEvalFunc vẫn đạt kết quả toàn thắng. Điểm đạt được trung bình cũng rất cao (>1000 điểm). Kết quả này đạt được do có ít kẻ địch hơn, nên lựa chọn di chuyển của Pacman rất thoải mái. Tuy nhiên, vẫn còn xuất hiện hiện tượng Pacman bị đóng băng nhẹ ở một vài seed khi dùng thuật toán AlphaBeta/Minimax. Lý giải cho điều này, một số trường hợp khoảng cách của Pacman đến chấm thức ăn và các ghosts đều gần như tương đương nhau, nếu lại gần thì sẽ cộng điểm từ đặc trưng foodDistance[0], nhưng lại bị trừ điểm từ đặc trưng ghost_dis, nên điểm số của việc di chuyển sẽ nhỏ hơn hoặc bằng điểm số của việc đứng yên, khiến Pacman bị đóng băng. Tuy nhiên vấn đề này đã được giải quyết khi dùng Expectimax.



4.minimaxClassic

minimaxClassic	ScoreEvaluationFunction		myEvalFunc		Algorithm
RandomSeed	Score	Record	Score	Record	
22520109+0	-492.0	Loss	-492.0	Loss	Minimax
22520109+1	-492.0	Loss	-492.0	Loss	
22520109+2	514.0	Win	514.0	Win	
22520109+3	-495.0	Loss	-495.0	Loss	
22520109+4	511.0	Win	512.0	Win	
Average score	-90,8		-90,6		
22520109+0	-492.0	Loss	-492.0	Loss	AlphaBeta
22520109+1	-492.0	Loss	-492.0	Loss	
22520109+2	514.0	Win	514.0	Win	
22520109+3	-495.0	Loss	-495.0	Loss	
22520109+4	511.0	Win	512.0	Win	
Average score	-90,8		-90,6		

22520109+0	507.0	Win	507.0	Win	Expectimax
22520109+1	-494.0	Loss	-494.0	Loss	
22520109+2	515.0	Win	515.0	Win	
22520109+3	508.0	Win	508.0	Win	
22520109+4	511.0	Win	512.0	Win	
Average score	309,4		309,6		

Trong layout minimaxClassic, bản đồ là cực kì nhỏ, số chấm thức ăn cũng ít, nhưng số lượng ghost là 3, vậy nên không gian di chuyển của Pacman là rất hạn chế. Vậy nên, khi sử dụng thuật toán Minimax/AlphaBeta, Pacman thường có xu hướng tiêu cực quá mức, cố gắng ăn 1 chấm thức ăn có thể, rồi tự kết liễu bản thân để tối đa hóa mức điểm khi thua, nhưng vẫn có ngoại lệ khi đường đi của những con ma lại tụ về phía bên trái của bản đồ, khiến Pacman có cơ hội để ăn hết thức ăn và hoàn thành màn chơi. Đổi lại khi dùng Expectimax, Pacman đã lạc quan hơn, khi dám liều lĩnh tạm thời lại gần kẻ địch để bám víu lấy cơ hội sống, từ đó mở ra những bước đi mới và hoàn thành màn chơi. Có thể thấy, điểm trung bình khi Pacman lạc quan cao hơn rất nhiều so với khi tiêu cực. Vậy nên, chúng ta cũng nên có cái nhìn lạc quan, không ngại khó trong cuộc sống như Pacman, từ đó nhiều điều tốt đẹp hơn sẽ đến với ta.

5.trappedClassic

trappedClassic	ScoreEvaluationFunction		myEvalFunc		Algorithm
RandomSeed	Score	Record	Score	Record	
22520109+0	-501.0	Loss	-501.0	Loss	Minimax
22520109+1	-501.0	Loss	-501.0	Loss	
22520109+2	-501.0	Loss	-501.0	Loss	
22520109+3	-501.0	Loss	-501.0	Loss	
22520109+4	-501.0	Loss	-501.0	Loss	
Average score	-501		-501		
22520109+0	-501.0	Loss	-501.0	Loss	AlphaBeta
22520109+1	-501.0	Loss	-501.0	Loss	
22520109+2	-501.0	Loss	-501.0	Loss	
22520109+3	-501.0	Loss	-501.0	Loss	
22520109+4	-501.0	Loss	-501.0	Loss	
Average score	-501		-501		
22520109+0	532.0	Win	532.0	Win	Expectimax
22520109+1	532.0	Win	532.0	Win	
22520109+2	-502.0	Loss	-502.0	Loss	
22520109+3	532.0	Win	532.0	Win	
22520109+4	532.0	Win	532.0	Win	
Average score	325,2		325,2		

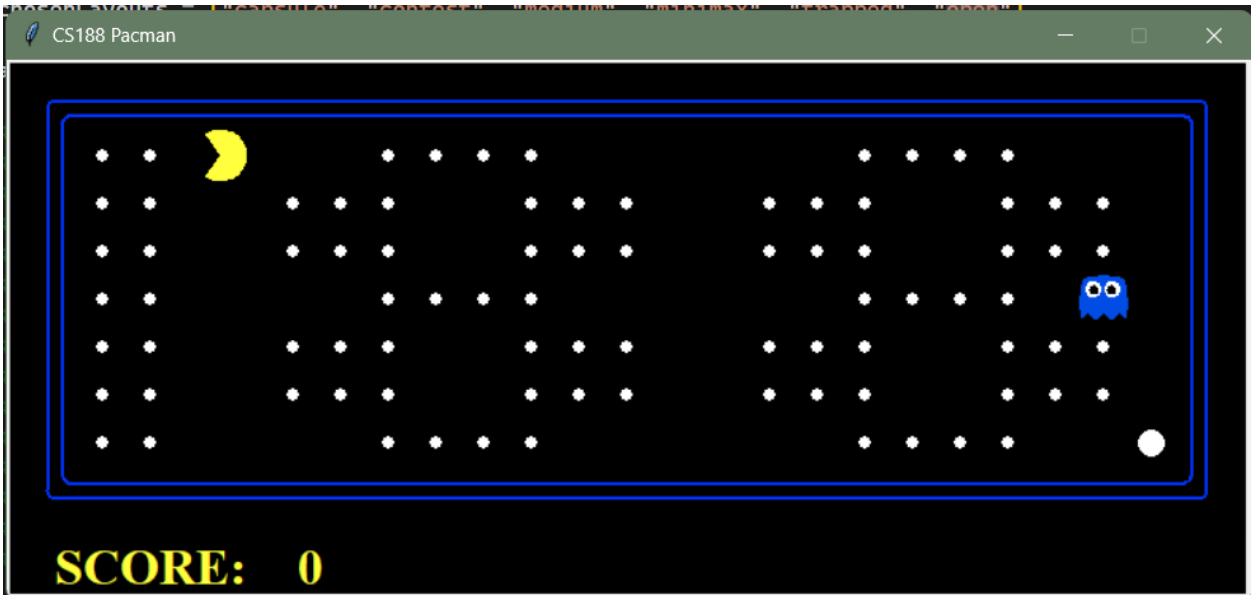
Ở màn chơi này, sự khác biệt giữa bi quan và lạc quan được thể hiện còn rõ hơn, với bản đồ thậm chí còn nhỏ hơn và ngặt nghoèo hơn cho Pacman khi chỉ có 1 đường đi duy nhất và đều bị bao vây bởi kẻ địch, những chấm thức ăn lại bị che lấp bởi địch. Lúc này khi thực hiện thuật toán Minimax/AlphaBeta, dù có là seed nào đi nữa thì Pacman đều chọn con đường kết thúc trò chơi sớm, nên kết quả đều là thua với số điểm giống nhau. Nhưng khi thực hiện Expectimax, Pacman đã chấp nhận rủi ro, tiến về phía con ma xanh và thành công ăn hết các chấm thức ăn, chỉ duy nhất 1 seed chúng ta thua khi con ma xanh cũng tiến lại gần Pacman. Ở màn chơi này, không có sự khác biệt giữa 2 hàm lượng giá.

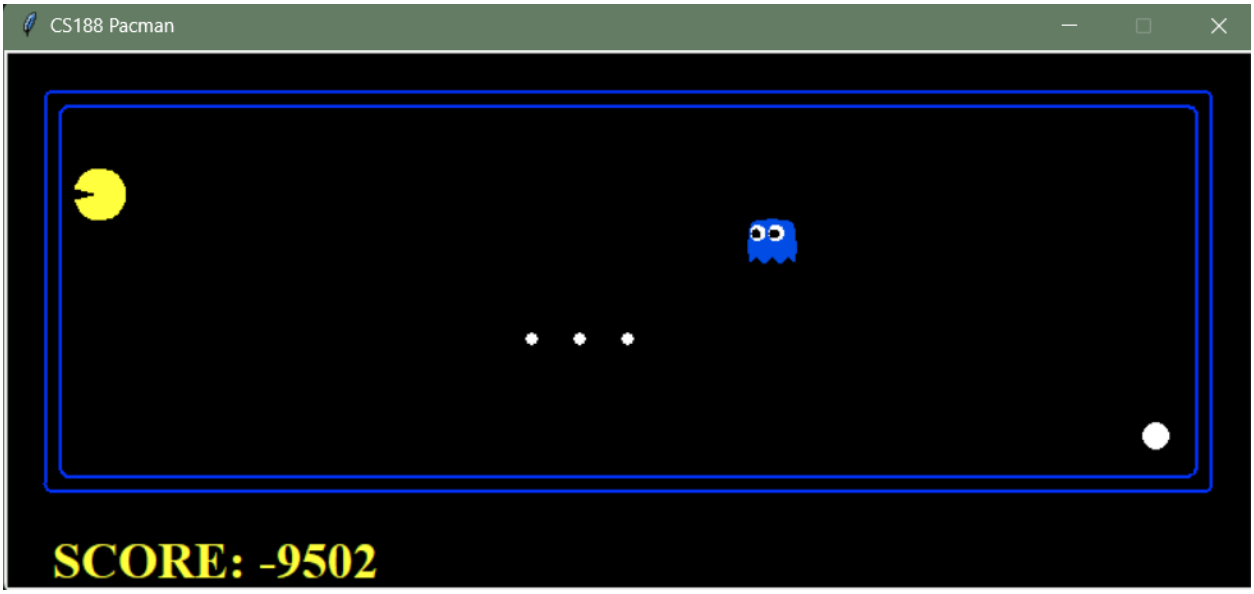
6.openClassic

openClassic	ScoreEvaluationFunction		myEvalFunc		Algorithm
RandomSeed	Score	Record	Score	Record	
22520109+0	-	?	1245.0	Win	Minimax
22520109+1	-	?	1236.0	Win	
22520109+2	-	?	1249.0	Win	
22520109+3	-	?	1244.0	Win	
22520109+4	-	?	1229.0	Win	
Average score	-		1240,6		
22520109+0	-	?	1245.0	Win	AlphaBeta
22520109+1	-	?	1236.0	Win	
22520109+2	-	?	1249.0	Win	
22520109+3	-	?	1244.0	Win	
22520109+4	-	?	1229.0	Win	
Average score	-		1240,6		
22520109+0	-	?	1239.0	Win	Expectimax
22520109+1	-	?	1244.0	Win	
22520109+2	-	?	1436.0	Win	
22520109+3	-	?	1428.0	Win	
22520109+4	-	?	1424.0	Win	
Average score	-		1354,2		

- *Chú thích: "-" và "?" nghĩa là chưa thể đo được kết quả cụ thể (dừng chương trình do thời gian chờ quá lâu, giải thích bên dưới)*

Đối với màn chơi này, có sự khác biệt rõ rệt giữa scoreEvaluationFunction và myEvalFunc. Bản đồ hoàn toàn không có tường và chỉ có các chấm thức ăn, cùng với 1 kẻ địch, 1 capsule duy nhất, vì thế nên số lượng bước đi có thể xảy ra của Pacman là rất lớn. Khi sử dụng hàm myEvalFunc, Pacman dễ dàng hoàn thành trò chơi với mức điểm tương đối cao. Tuy nhiên đây vẫn chưa phải là mức điểm tối ưu, vì Pacman vẫn chưa hấp thụ capsule và đi săn ma ở hầu hết màn chơi. Lý giải cho điều này, vì ở lúc bắt đầu, vị trí của capsule và kẻ địch đều ở rất xa, nên Pacman sẽ tập trung vào việc hấp thụ những chấm thức ăn ở gần mình hơn là ưu tiên chạy đến viên capsule. Tuy nhiên, ở chiều ngược lại, khi sử dụng hàm scoreEvaluationFunction, Pacman lại hoàn toàn đóng băng sau khi ăn xong một vài chấm thức ăn đầu tiên. Điều này là vì hàm lượng giá này đánh giá mọi bước đi đều có giá trị thấp hơn việc đứng yên, nên Pacman cũng không chịu di chuyển. Thời gian trôi qua rất lâu nhưng do con ma không có chủ ý đuổi theo Pacman nên Pacman vẫn sẽ đứng yên, nên điểm ngày càng trừ xuống dần (lâu nhất đo được là khi score = -9500 điểm sau 40 phút), vậy nên không thể kết luận được kết quả và số điểm của màn chơi ở cả 3 thuật toán.





Pacman bị đóng băng (thuật toán Expectimax, scoreEvaluationFunction, random seed = 22520109 + 3)

Pacman chỉ di chuyển để né tránh khi con ma lại gần nhưng lại không có xu hướng tiến tới những chấm thức ăn.

III. Kết luận

Sau khi phân tích kết quả thực nghiệm, ta có thể kết luận rằng:

- Hàm lượng giá được thiết kế tốt sẽ giúp Pacman đạt được nhiều điểm hơn, tránh thua và đóng băng Pacman quá lâu.
- Thuật toán Minimax và AlphaBeta đều đạt được kết quả hoàn toàn giống nhau ở mọi màn chơi, nhưng thời gian thực hiện của AlphaBeta vượt trội hơn ở những màn chơi rộng, có nhiều kẻ địch → nên dùng AlphaBeta để tiết kiệm thời gian.
- Thuật toàn lạc quan Expectimax có số màn chơi thắng nhiều hơn thuật toán bi quan Minimax/AlphaBeta, hơn nữa điểm số khi thắng đều cao hơn. Kết quả này là do đường đi của kẻ địch đều là ngẫu nhiên.

___HẾT___