Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №3

з дисципліни «Системи штучного інтелекту»

на тему

«Методи пошуку в умовах протидії»

Виконали:
студентки групи ІС-12
Павлова Софія
Гоголь Софія

Викладач:
Коломоєць С. О.

Київ – 2023

1. Постановка задачі

Мета: ознайомитись з методами пошуку в умовах протидії та дослідити їх для агенту в типовому ігровому середовищі.

Завдання:

1. Обрати середовище моделювання та задачу, що містить декілька агентів, що протидіють один одному (середовище має моделювати гру з нульовою сумою).
2. В обраному середовищі вирішити поставлену задачу, реалізувавши один з методів пошуку в умовах протидії (MiniMax, альфа-бета відсікання або ExpectiMax).
3. Реалізувати власну функцію оцінки станів. Реалізація цієї функції є дослідженням даної роботи.
4. Описати використаний метод, власну функцію та результати його застосування.
5. Виконати дослідження впливу деякого фактора середовища.

8	Альфа-бета відсікання	Вплив кількості агентів-привидів

2. Виконання

2.1. Середовище

У якості середовища було обрано середовище для гри Pacman з 10 різними картами.

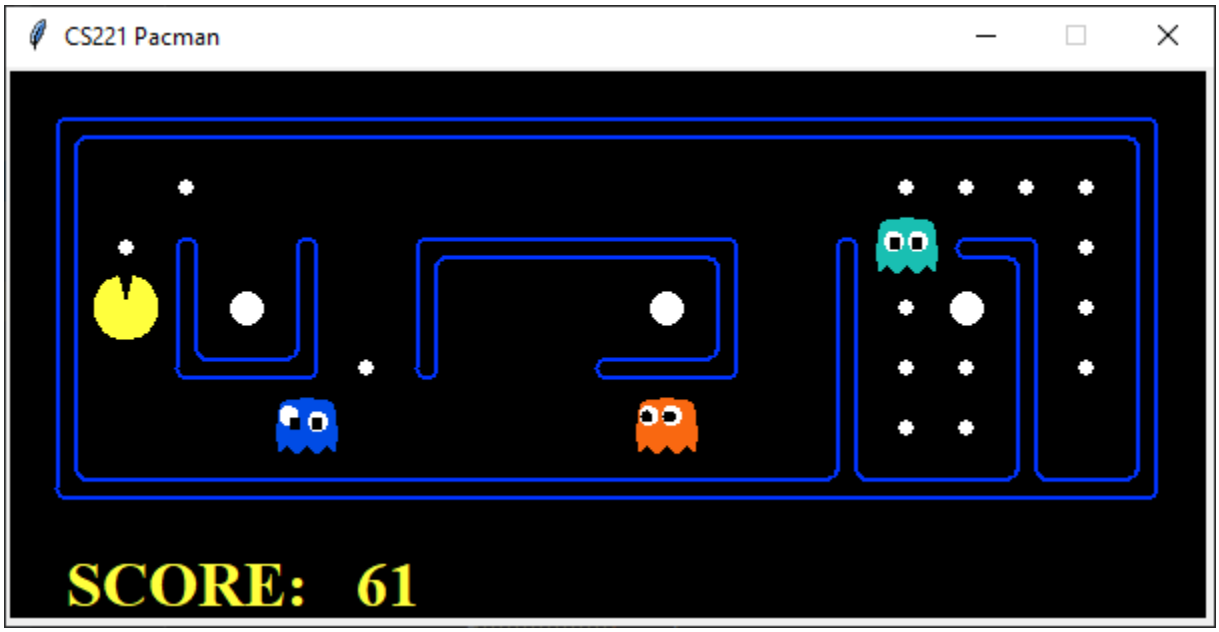


Рисунок 1 – capsuleClassic

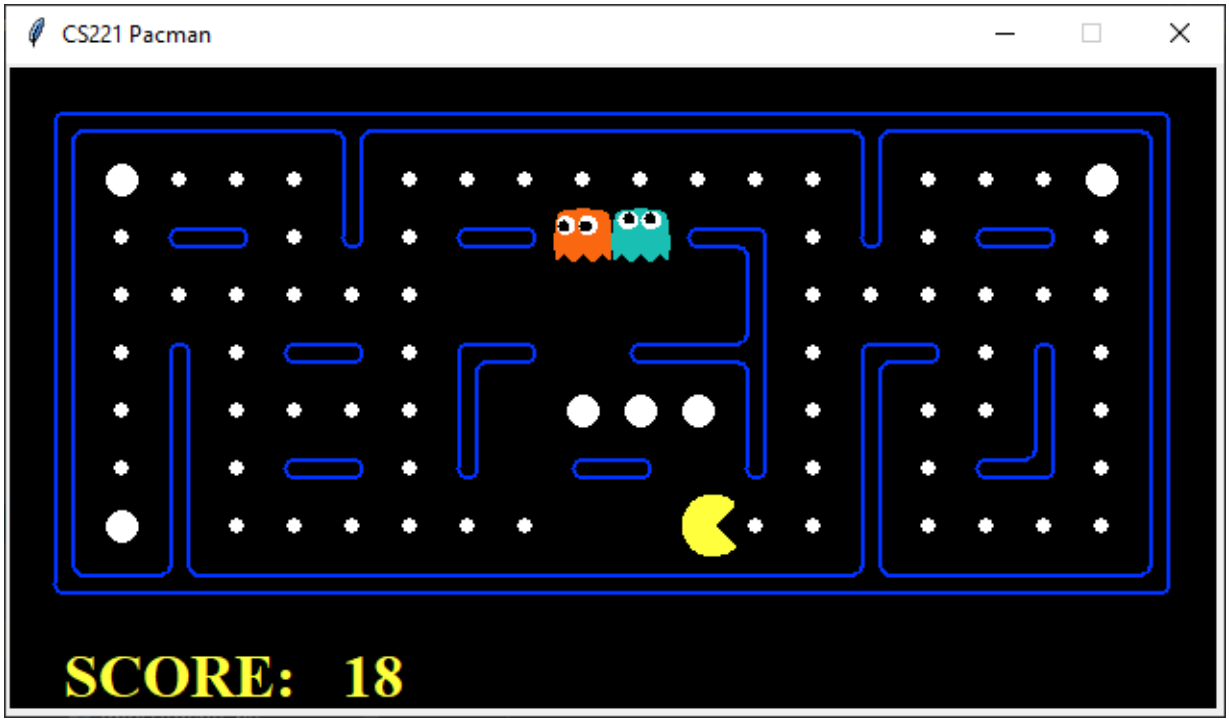


Рисунок 2 – contestClassic

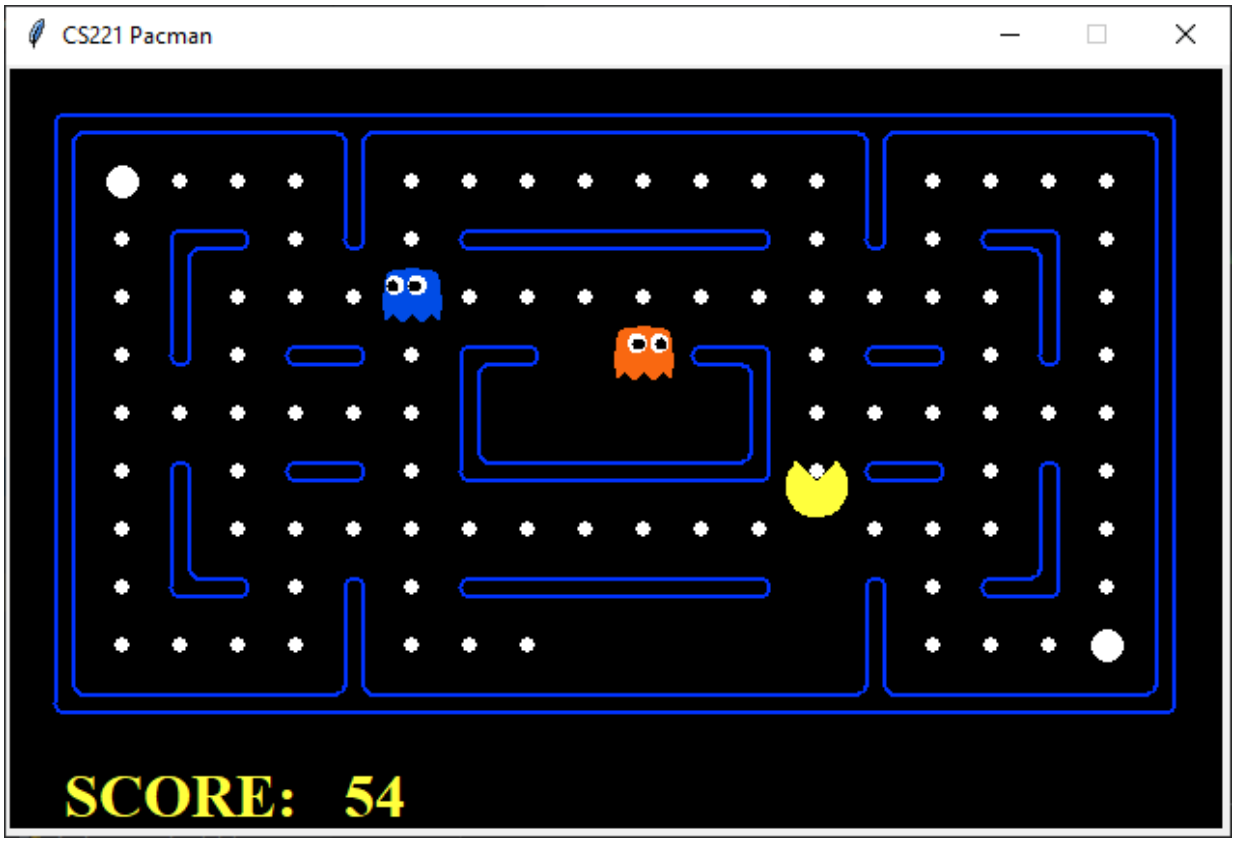


Рисунок 3 – mediumClassic

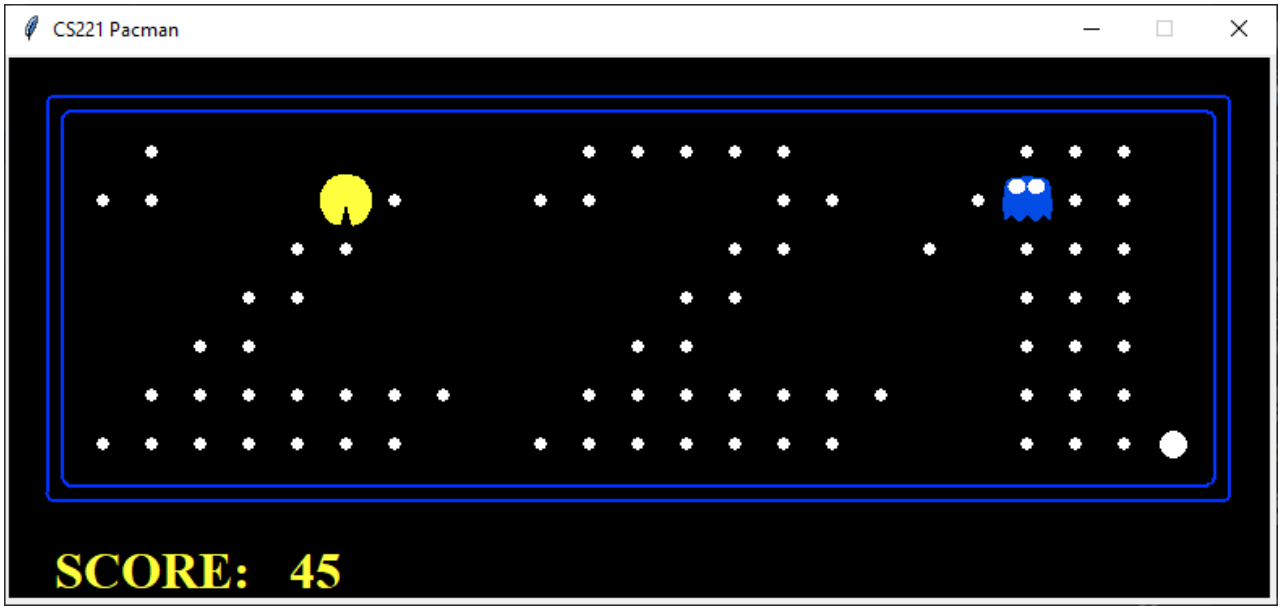


Рисунок 4 – openClassic

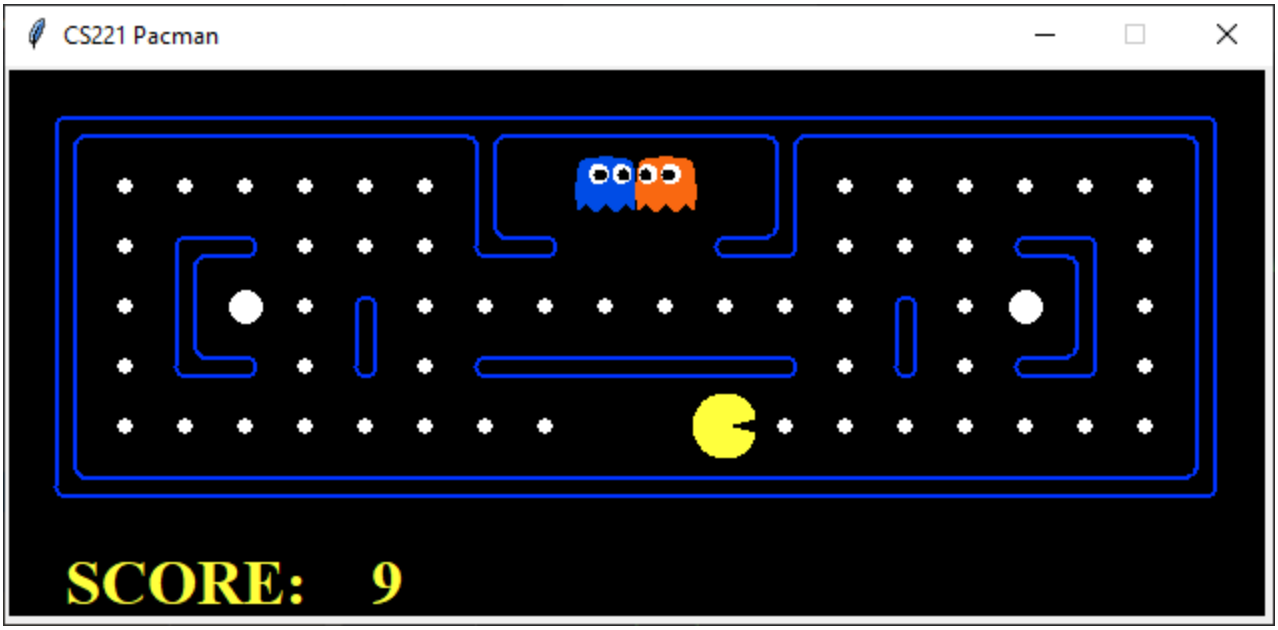


Рисунок 5 – smallClassic

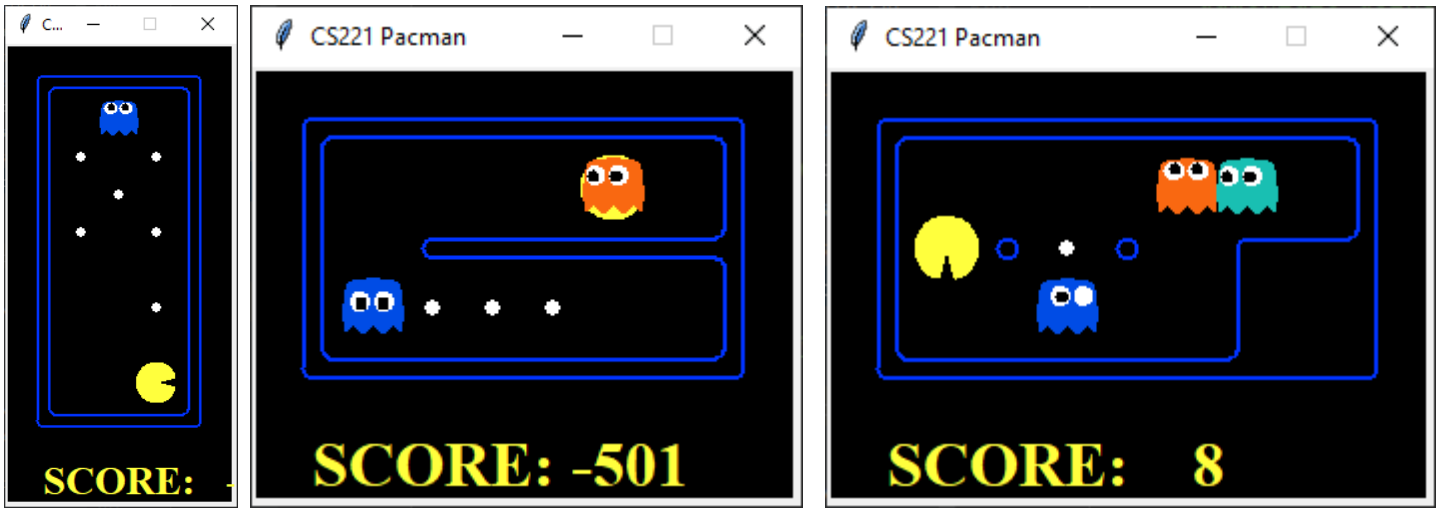


Рисунок 6 – testClassic, trappedClassic, minimaxClassic

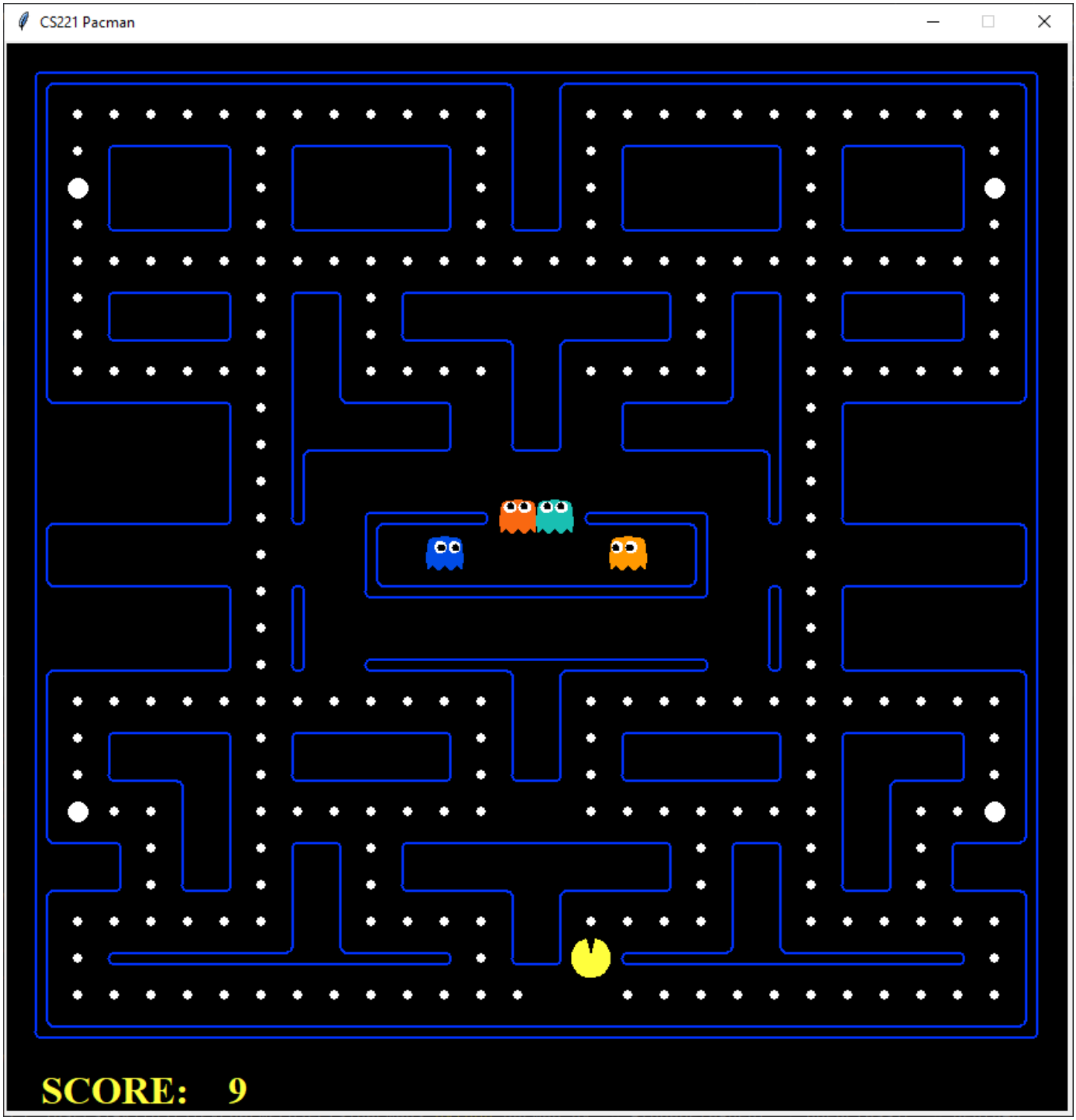


Рисунок 7 – originalClassic

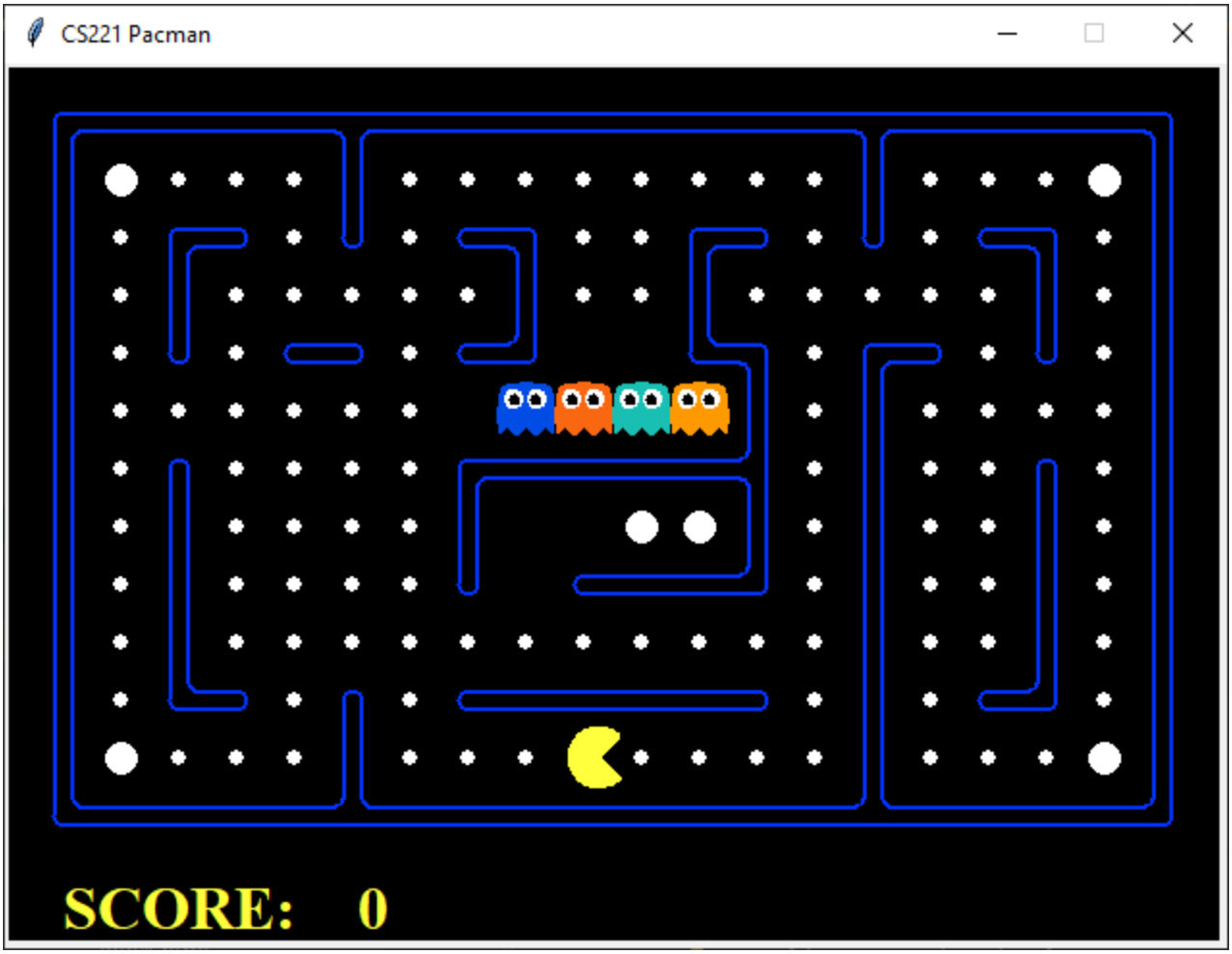


Рисунок 8 – trickyClassic

2.2. Алгоритм

Відповідно до завдання, необхідно реалізувати **альфа-бета відсікання** – модифікація методу MiniMax, що дозволяє не розкривати ряд гілок дерева в процесі роботи.

Задача:

Необхідно створити нового агента, який використовує альфа-бета спрощення для більше ефективного дослідження мінімаксного дерева у **AlphaBetaAgent**.

Мінімаксні значення **AlphaBetaAgent** повинні бути ідентичні до мінімаксних значень **MinimaxAgent**, хоча вибрані дії можуть різнитися через різну поведінку.

Мінімаксні значення для початкового стану на карті **minimaxClassic** мають бути **9, 8, 7**, та **-493** для глибини **1, 2, 3**, та **4** відповідно.

Мінімаксні значення для початкового стану на карті **mediumClassic** повинні бути **9, 18, 27**, та **36** для глибини **1, 2, 3** та **4**, відповідно.

Лістинг коду – MiniMaxAgent:

```
class MinimaxAgent(MultiAgentSearchAgent):

  def getAction(self, gameState: GameState) -> str:

    " Max Value for computing the best direction of the pacman "
    def max_value(gameState, depth):
      " Cases checking "
      actionList = gameState.getLegalActions(0)  # Get actions of pacman
      if len(actionList) == 0 or gameState.isWin() or gameState.isLose() or depth ==
self.depth:
        return (self.evaluationFunction(gameState), None)
      " Initializing the value of v and action to be returned "
      v = -(float("inf"))
      goAction = None
      for thisAction in actionList:
        successorValue = min_value(gameState.generateSuccessor(0, thisAction), 1,
depth)[0]
        " Get value of v and action, max(v, successorValue) "
        if (successorValue > v):
          v, goAction = successorValue, thisAction

      # Для тестування значень
      # print(v)
      return (v, goAction)

    " Min Value for computing the worst case direction of the ghost "
    def min_value(gameState, agentID, depth):
      " Cases checking "
      actionList = gameState.getLegalActions(agentID)  # Get the actions of the ghost
      if len(actionList) == 0:
        return (self.evaluationFunction(gameState), None)
      " Initializing the value of v and action to be returned "
      v = float("inf")
      goAction = None
      for thisAction in actionList:
        if (agentID == gameState.getNumAgents() - 1):
          successorValue = max_value(gameState.generateSuccessor(agentID, thisAction),
depth + 1)[0]
        else:
          successorValue = min_value(gameState.generateSuccessor(agentID, thisAction),
agentID + 1, depth)[0]
        " Get value of v and action, min(v, successorValue) "
        if (successorValue < v):
          v, goAction = successorValue, thisAction
      return (v, goAction)

    return max_value(gameState, 0)[1]
```

Лістинг коду – AlphaBetaAgent:

```
class AlphaBetaAgent(MultiAgentSearchAgent):

    " Max value "
    def max_value(gameState, depth, alpha, beta):
      " Cases checking "
      actionList = gameState.getLegalActions(0)  # Get actions of pacman
      if len(actionList) == 0 or gameState.isWin() or gameState.isLose() or depth ==
self.depth:
        return (self.evaluationFunction(gameState), None)
      " Initializing the value of v and action to be returned "
      v = -(float("inf"))
      goAction = None
      for thisAction in actionList:
        successorValue = min_value(gameState.generateSuccessor(0, thisAction), 1, depth,
alpha, beta)[0]
        " v = max(v, successorValue) "
        if (v < successorValue):
          v, goAction = successorValue, thisAction
        if (v > beta):
          return (v, goAction)
        alpha = max(alpha, v)

      # Для тестування значень
      # print(v)
      return (v, goAction)

    " Min value "
    def min_value(gameState, agentID, depth, alpha, beta):
      " Cases checking "
      actionList = gameState.getLegalActions(agentID)  # Get the actions of the ghost
      if len(actionList) == 0:
        return (self.evaluationFunction(gameState), None)
      " Initializing the value of v and action to be returned "
      v = float("inf")
      goAction = None
      for thisAction in actionList:
        if (agentID == gameState.getNumAgents() - 1):
          successorValue = max_value(gameState.generateSuccessor(agentID, thisAction),
depth + 1, alpha, beta)[0]
        else:
          successorValue = \
          min_value(gameState.generateSuccessor(agentID, thisAction), agentID + 1, depth,
alpha, beta)[0]
        " v = min(v, successorValue) "
        if (successorValue < v):
          v, goAction = successorValue, thisAction
        if (v < alpha):
          return (v, goAction)
        beta = min(beta, v)
      return (v, goAction)
    alpha = -(float("inf"))
    beta = float("inf")
    return max_value(gameState, 0, alpha, beta)[1]
```

Перевіримо спочатку карту **minimaxClassic.**

Результат:

```
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p MinimaxAgent -l minimaxClassic -a depth=1
9
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p MinimaxAgent -l minimaxClassic -a depth=2
8
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p MinimaxAgent -l minimaxClassic -a depth=3
7
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p MinimaxAgent -l minimaxClassic -a depth=4
6
6
6
6
6
6
6
6
6
6
6
6
6
-493
```

Рисунок 9 – Значення для початкового стану на карті **minimaxClassic** для **MiniMaxAgent**

```
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -l minimaxClassic -a depth=1
9
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -l minimaxClassic -a depth=2
8
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -l minimaxClassic -a depth=3
7
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -l minimaxClassic -a depth=4
6
6
6
6
6
6
6
6
6
6
6
6
6
-493
```

Рисунок 10 – Значення для початкового стану на карті **minimaxClassic** для **AlphaBetaAgent**

Отримаємо наступні значення:

Таблиця 1 – Початкові стани на карті **minimaxClassic** для **MiniMaxAgent**

Алгоритм	*Карта*	*Глибина*	*Очікуване знач.*	*Реальне знач.*
MiniMax	minimaxClassic	1	9	9
		2	8	8
		3	7	7
		4	-493	-493

Таблиця 2 – Початкові стани на карті **minimaxClassic** для **AlphaBetaAgent**

Алгоритм	*Карта*	*Глибина*	*MiniMax знач.*	*AlphaBeta знач.*
AlphaBeta	minimaxClassic	1	9	9
		2	8	8
		3	7	7
		4	-493	-493

Далі перевіримо карту **mediumClassic**.

Результат:

```
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p MinimaxAgent -l mediumClassic -a depth=1
9
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p MinimaxAgent -l mediumClassic -a depth=2
18
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p MinimaxAgent -l mediumClassic -a depth=3
27
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p MinimaxAgent -l mediumClassic -a depth=4
36
```

Рисунок 11 – Значення для початкового стану на карті **mediumClassic** для **MiniMaxAgent**

```
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -l mediumClassic -a depth=1
9
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -l mediumClassic -a depth=2
18
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -l mediumClassic -a depth=3
27
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -l mediumClassic -a depth=4
36
```

Рисунок 12 – Значення для початкового стану на карті **mediumClassic** для **AlphaBetaAgent**

Отримаємо наступні значення:

Таблиця 3 – Початкові стани на карті **mediumClassic** для **MiniMaxAgent**

Алгоритм	*Карта*	*Глибина*	*Очікуване знач.*	*Реальне знач.*
MiniMax	mediumClassic	1	9	9
		2	18	18
		3	27	27
		4	36	36

Таблиця 4 – Початкові стани на карті **mediumClassic** для **AlphaBetaAgent**

Алгоритм	*Карта*	*Глибина*	*MiniMax знач.*	*AlphaBeta знач.*
AlphaBeta	mediumClassic	1	9	9
		2	18	18
		3	27	27
		4	36	36

Бачимо, що для обох карт вимоги виконані.

2.3. Спеціалізована функція оцінки

Необхідно написати кращу функцію оцінки для Пакмена, яку забезпечує функція **betterEvaluationFunction**.

Дана функція оцінки повинна оцінювати стани (а не дії).

Задача:

Для глибин пошуку 2, агент з покращеною функцією повинен перемагати на карті **smallClassic** з **двома випадковими привидами** принаймні **в половині випадків** і мати розумні часові витрати.

Лістинг коду:

```
def betterEvaluationFunction(currentGameState: GameState) -> float:

  pacmanPos = currentGameState.getPacmanPosition()
  ghostList = currentGameState.getGhostStates()
  foods = currentGameState.getFood()
  capsules = currentGameState.getCapsules()
  # Return based on game state
  if currentGameState.isWin():
    return float("inf")
  if currentGameState.isLose():
    return float("-inf")
  # Populate foodDistList and find minFoodDist
  foodDistList = []
  for each in foods.asList():
    foodDistList = foodDistList + [util.manhattanDistance(each, pacmanPos)]
  minFoodDist = min(foodDistList)
  # Populate ghostDistList and scaredGhostDistList, find minGhostDist and
minScaredGhostDist
  ghostDistList = []
  scaredGhostDistList = []
  for each in ghostList:
    if each.scaredTimer == 0:
      ghostDistList = ghostDistList + [util.manhattanDistance(pacmanPos,
each.getPosition())]
    elif each.scaredTimer > 0:
      scaredGhostDistList = scaredGhostDistList + [util.manhattanDistance(pacmanPos,
each.getPosition())]
  minGhostDist = -1
  if len(ghostDistList) > 0:
    minGhostDist = min(ghostDistList)
  minScaredGhostDist = -1
  if len(scaredGhostDistList) > 0:
    minScaredGhostDist = min(scaredGhostDistList)
  # Evaluate score
  # score = scoreEvaluationFunction(currentGameState)
  score = 0
  # Distance to closest food
```

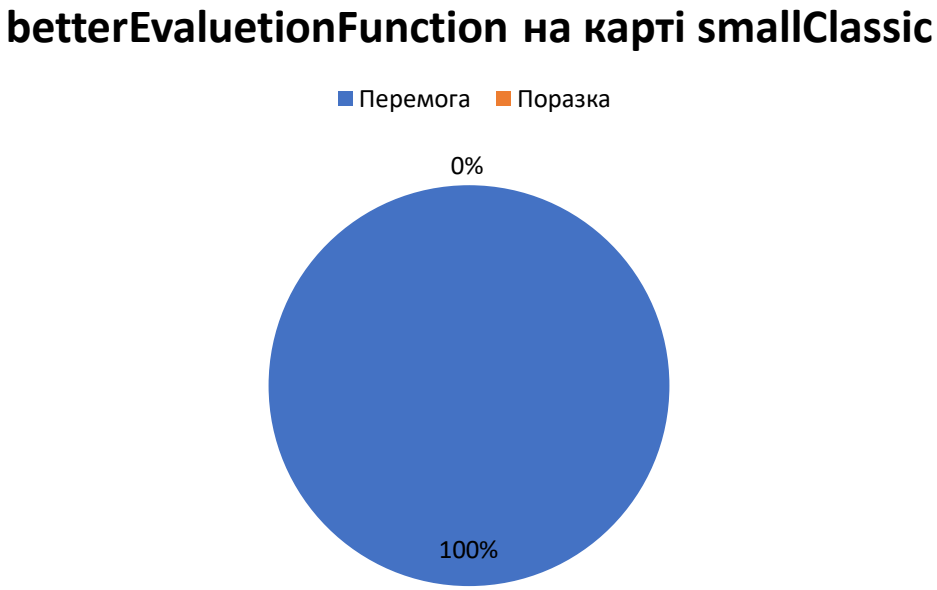
```
  score = score + (-1.5 * minFoodDist)
  # Distance to closest ghost
  score = score + (-2 * (1.0 / minGhostDist))
  # Distance to closest scared ghost
  score = score + (-2 * minScaredGhostDist)
  # Number of capsules
  score = score + (-20 * len(capsules))
  # Number of food
  score = score + (-4 * len(foods.asList()))
  return score
```

Запустимо на карті **smallClassic** з **AlphaBetaAgent** і **2 привидами**, щоб переконатися, що функція оцінки працює.

Результат:

```
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -l smallClassic -p AlphaBetaAgent -a evalFn=better -n
 10
Pacman emerges victorious! Score: 929
Pacman emerges victorious! Score: 1340
Pacman emerges victorious! Score: 1050
Pacman emerges victorious! Score: 1143
Pacman emerges victorious! Score: 1476
Pacman emerges victorious! Score: 972
Pacman emerges victorious! Score: 1321
Pacman emerges victorious! Score: 1499
Pacman emerges victorious! Score: 916
Pacman emerges victorious! Score: 950
```

Рисунок 13 – Тестування на карті **smallClassic** з 2 привидами



Діаграма 1 – Результат тестування **betterEvsluationFunction**

2.4. Дослідження впливу кількості привидів

Для дослідження впливу кількості привидів на успішне завершення гри в пакмена, проведемо по **10 ігор** для карти **mediumClassic** з глибинами **2, 3** та **4** для **1 та 2 привидів** відповідно.

Почнімо з **2 привидів**.

Результат:

```
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=4 -k 2 -n 10
Pacman emerges victorious! Score: 1463
Pacman died! Score: 490
Pacman emerges victorious! Score: 1826
Pacman died! Score: 663
Pacman died! Score: -150
Pacman died! Score: -224
Pacman died! Score: -836
Pacman died! Score: 1100
Pacman emerges victorious! Score: 1874
Pacman died! Score: 780
('Average Score:', 698.6)
('Scores:       ', '1463, 490, 1826, 663, -150, -224, -836, 1100, 1874, 780')
Win Rate:      3/10 (0.30)
('Record:       ', 'Win, Loss, Win, Loss, Loss, Loss, Loss, Loss, Win, Loss')
```

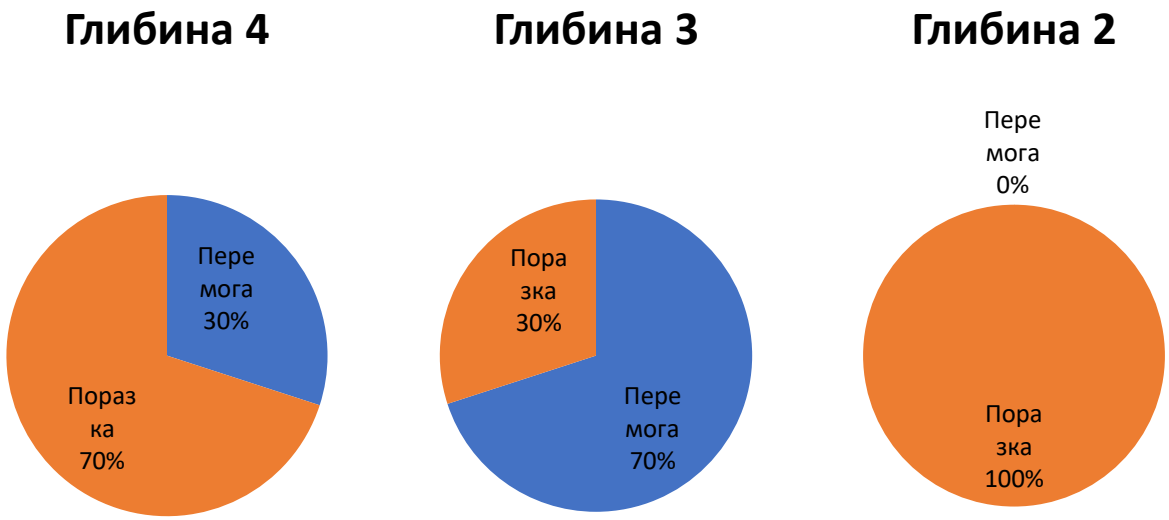
Рисунок 14 – **2 агенти** з **глибиною 4**

```
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=3 -k 2 -n 10
Pacman emerges victorious! Score: 1354
Pacman emerges victorious! Score: 1432
Pacman emerges victorious! Score: 1507
Pacman emerges victorious! Score: 1724
Pacman emerges victorious! Score: 1324
Pacman died! Score: -189
Pacman emerges victorious! Score: 1405
Pacman died! Score: 546
Pacman emerges victorious! Score: 960
Pacman died! Score: -220
('Average Score:', 984.3)
('Scores:       ', '1354, 1432, 1507, 1724, 1324, -189, 1405, 546, 960, -220')
Win Rate:      7/10 (0.70)
('Record:       ', 'Win, Win, Win, Win, Win, Loss, Win, Loss, Win, Loss')
```

Рисунок 15 – **2 агенти** з **глибиною 3**

```
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=2 -k 2 -n 10
Pacman died! Score: -45
Pacman died! Score: 362
Pacman died! Score: -285
Pacman died! Score: 12
Pacman died! Score: 231
Pacman died! Score: 257
Pacman died! Score: -701
Pacman died! Score: 331
Pacman died! Score: 518
Pacman died! Score: -360
('Average Score:', 32.0)
('Scores:       ', '-45, 362, -285, 12, 231, 257, -701, 331, 518, -360')
Win Rate:      0/10 (0.00)
('Record:       ', 'Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss')
```

Рисунок 16 – **2 агенти** з **глибиною 2**



Діаграма 2 – Результат дослідження впливу 2 привидів

Тепер дослідимо вплив **1 привида**.

Результат:

```
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=4 -k 1 -n 10
Pacman emerges victorious! Score: 1153
Pacman emerges victorious! Score: 1367
Pacman emerges victorious! Score: 1485
Pacman emerges victorious! Score: 834
Pacman emerges victorious! Score: 1291
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=4 -k 1 -n 10
Pacman emerges victorious! Score: 1072
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=4 -k 1 -n 10
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=4 -k 1 -n 10
Pacman emerges victorious! Score: 1180
Pacman emerges victorious! Score: 1035
Pacman emerges victorious! Score: 1364
```

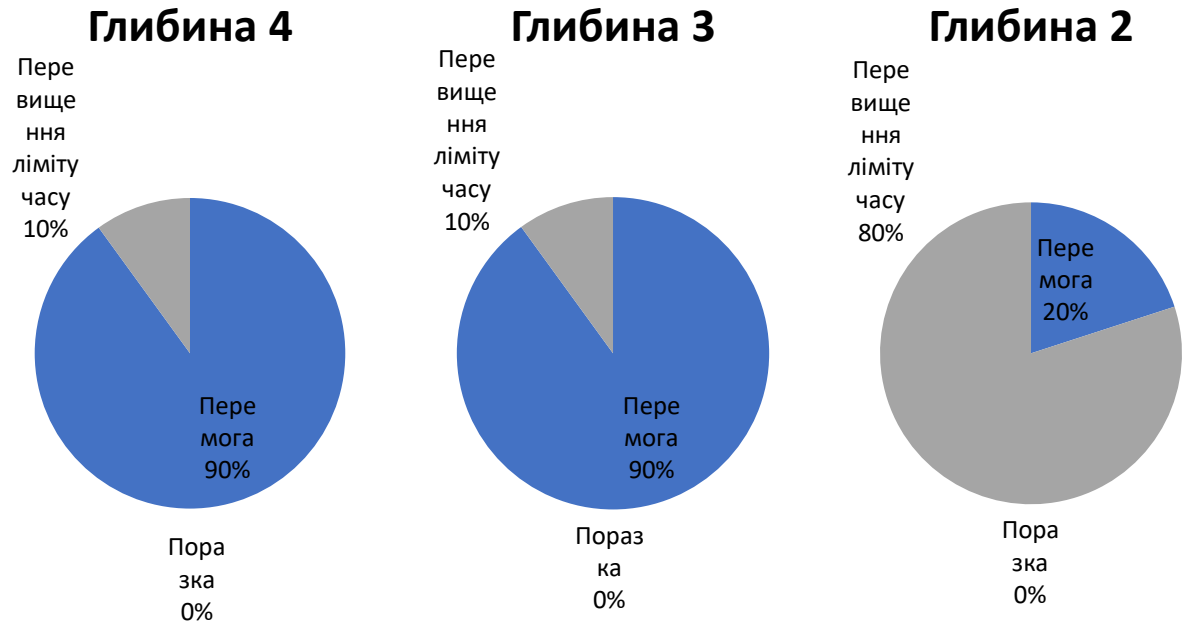
Рисунок 17 – **1 агент** з **глибиною 4**

```
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=3 -k 1 -n 10
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=3 -k 1 -n 10
Pacman emerges victorious! Score: 966
Pacman emerges victorious! Score: 1128
Pacman emerges victorious! Score: 1315
Pacman emerges victorious! Score: 879
Pacman emerges victorious! Score: 1104
Pacman emerges victorious! Score: 2
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=3 -k 1 -n 10
Pacman emerges victorious! Score: 1193
Pacman emerges victorious! Score: 1419
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=3 -k 1 -n 10
Pacman emerges victorious! Score: 1088
```

Рисунок 18 – **1 агент** з **глибиною 3**

```
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=2 -k 1 -n 10
Pacman emerges victorious! Score: 975
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=2 -k 1 -n 10
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=2 -k 1 -n 10
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=2 -k 1 -n 10
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=2 -k 1 -n 10
Pacman emerges victorious! Score: 151
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=2 -k 1 -n 10
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=2 -k 1 -n 10
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=2 -k 1 -n 10
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=2 -k 1 -n 10
PS C:\Users\Akil0515\PycharmProjects\pacman> python pacman.py -p AlphaBetaAgent -a depth=2 -k 1 -n 10
```

Рисунок 19 – **1 агент** з **глибиною 2**



Діаграма 3 – Результат дослідження впливу 1 привида

2.5. Аналіз отриманих результатів

На основі отриманих результатів дослідження можна помітити, що збільшення кількості привидів призводить до частішого програшу пакмена. При цьому на великих глибинах пакмен має велику обчислювальну складність, що призводить до сплутання дій пакмена і частіших програшів. А на малих глибинах пакмен робить недостатньо розумні дії, щоб виграти у двох привидів, тому постійно програє.

Тоді як зменшення кількості привидів анульовує програшні ситуації, але при цьому часто призводить до зацикленості пакмена і вичерпання «адекватного» часу на виграш. При чому чим менша глибина – тим частіше пакмен буде зациклюватись.

Висновок:

Для виконання даної лабораторної роботи за середовище моделювання гри ми обрали середовище гри пакмен, що містить декілька агентів, які протидіють один одному.

Обрана задача була вирішена за допомогою методу пошуку в умовах протидії – альфа-бета відсікання.

У результаті виконання лабораторної роботи ми розробили власну функцію оцінки станів, що використовується в процесі прийняття рішень агентом та дослідили вплив кількості агентів-привидів на результати вирішення задачі.