PAC DOM

CHARACTER / NICKNAME

 - SHADOW "DOC"

 - SPEEDY "PINKY"

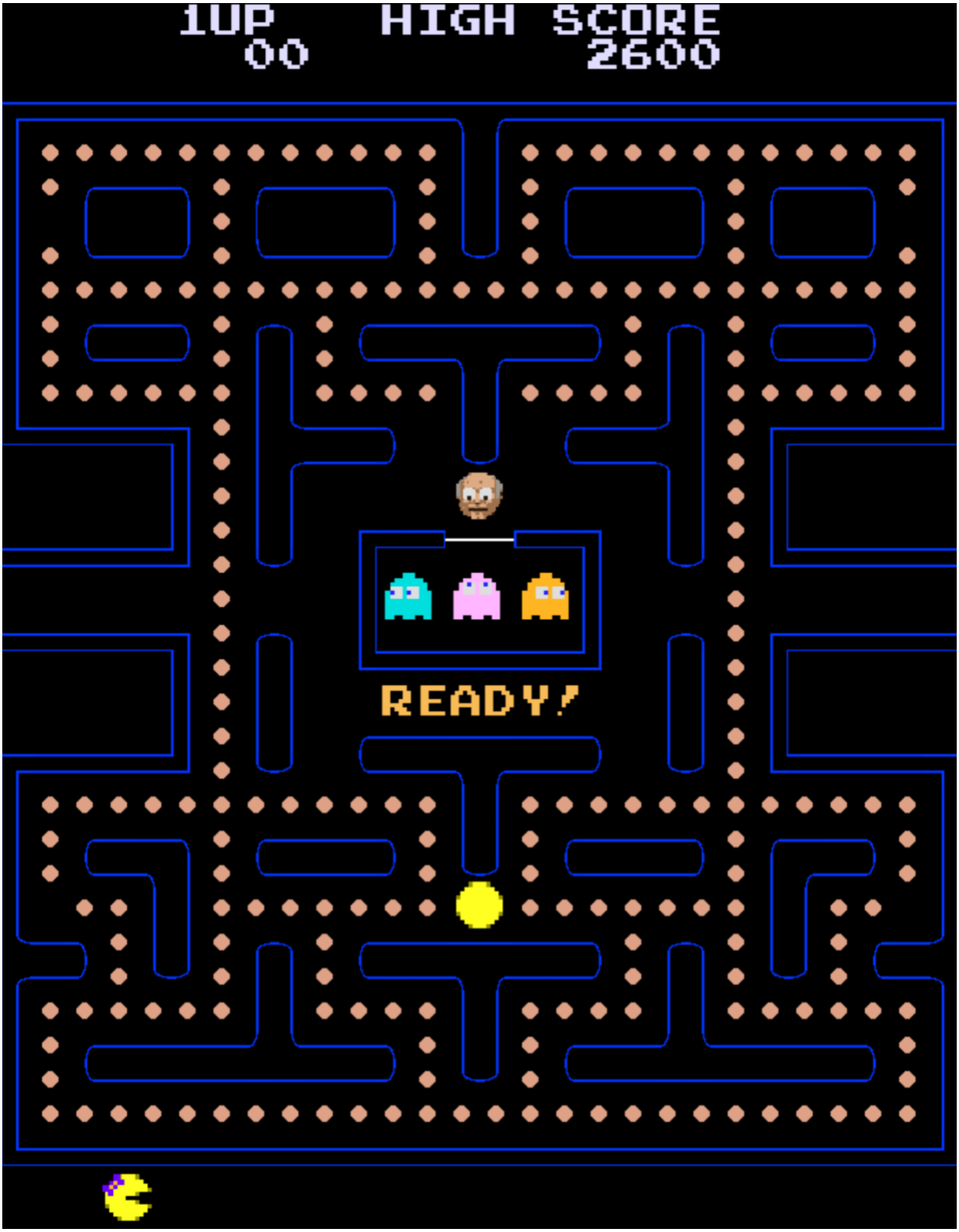
 - BASHFUL "INKY"

 - POKEY "CLYDE"




DOMINIQUE KELLAM 2023

PRESS SPACE TO PLAY


1UP
00
HIGH SCORE
2600
READY!

```
def setDir(self): #Informed Breadth-First-Search based on direction of Pacman
    dirs = [[0, -self.ghostSpeed, 0],
            [1, 0, self.ghostSpeed],
            [2, self.ghostSpeed, 0],
            [3, 0, -self.ghostSpeed]
    ]

    if self.algorithm == 'BFS':
        random.shuffle(dirs)
        best = 10000
        bestDir = -1
        for newDir in dirs:
            if self.calcDistance(self.target, [self.row + newDir[1], self.col + newDir[2]]) < best:
                if not (self.lastLoc[0] == self.row + newDir[1] and self.lastLoc[1] == self.col + newDir[2]):
                    if newDir[0] == 0 and self.col % 1.0 == 0:
                        if self.isValid(math.floor(self.row + newDir[1]), int(self.col + newDir[2])):
                            bestDir = newDir[0]
                            best = self.calcDistance(self.target, [self.row + newDir[1], self.col + newDir[2]])
                    elif newDir[0] == 1 and self.row % 1.0 == 0:
                        if self.isValid(int(self.row + newDir[1]), math.ceil(self.col + newDir[2])):
                            bestDir = newDir[0]
                            best = self.calcDistance(self.target, [self.row + newDir[1], self.col + newDir[2]])
                    elif newDir[0] == 2 and self.col % 1.0 == 0:
                        if self.isValid(math.ceil(self.row + newDir[1]), int(self.col + newDir[2])):
                            bestDir = newDir[0]
                            best = self.calcDistance(self.target, [self.row + newDir[1], self.col + newDir[2]])
                    elif newDir[0] == 3 and self.row % 1.0 == 0:
                        if self.isValid(int(self.row + newDir[1]), math.floor(self.col + newDir[2])):
                            bestDir = newDir[0]
                            best = self.calcDistance(self.target, [self.row + newDir[1], self.col + newDir[2]])
    else: #A-Star *!!!
        best_f = 100000000000
        bestDir = -1
        for newDir in dirs:
            g = 1
            h = abs(self.target[0] - (self.row + newDir[1])) + abs(self.target[1] - (self.row + newDir[2])) # Manhattan Distance - Approximation stics
            f = g + h #It calculates the total cost f of the path from the current location to the target location, by adding the heuristic cost h to a fixed cost g of 1.

            #  loops over each direction in the dirs list. calculates a heuristic function h based on the current location
            #  If the calculated f value is better than the current best_f value, the code checks if the next location in this direction is not the previous location
            #  (self.lastLoc), and if it is a valid location (checked using the isValid method)
            if best_f > f:
                if not (self.lastLoc[0] == self.row + newDir[1] and self.lastLoc[1] == self.col + newDir[2]):
                    if newDir[0] == 0 and self.col % 1.0 == 0:
                        if self.isValid(math.floor(self.row + newDir[1]), int(self.col + newDir[2])):
                            bestDir = newDir[0]
                            best_f = f
                    elif newDir[0] == 1 and self.row % 1.0 == 0:
                        if self.isValid(int(self.row + newDir[1]), math.ceil(self.col + newDir[2])):
                            bestDir = newDir[0]
                            best_f = f
                    elif newDir[0] == 2 and self.col % 1.0 == 0:
                        if self.isValid(math.ceil(self.row + newDir[1]), int(self.col + newDir[2])):
                            bestDir = newDir[0]
                            best_f = f
                    elif newDir[0] == 3 and self.row % 1.0 == 0:
                        if self.isValid(int(self.row + newDir[1]), math.floor(self.col + newDir[2])):
                            bestDir = newDir[0]
                            best_f = f
     #  After all directions are evaluated, self.dir is set to bestDir.
    self.dir = bestDir
```