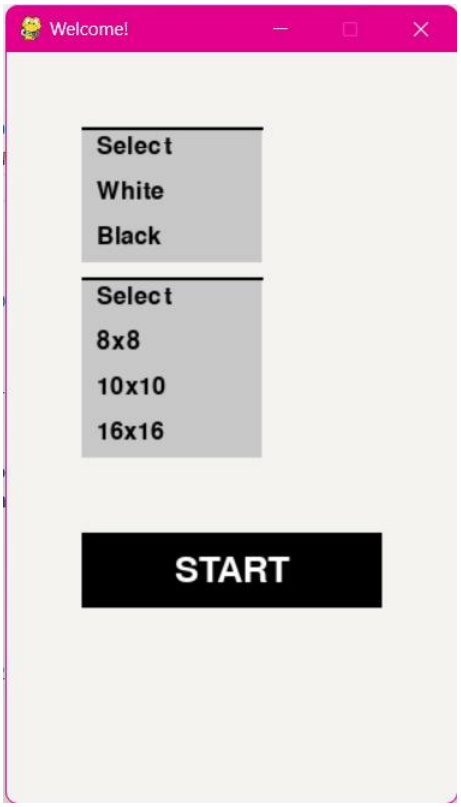
Veštačka inteligencija - izveštaj I faze

PinkTeam
Mijajlović Anđelija 18247
Joksimović Kristina 18203

1. **Izbor ko će igrati prvi (čovek ili računar)**
   Beli igrač uvek igra prvi, a omogućen je izbor boje na početku igre.

2. **Funkcije koje obezbeđuju unos početnih parametara igre**
   Dozvoljen je izbor 3 moguće dimenzije table na osnovu kojih se inicijalizuje igra.

   Kod koji omogućava sledeće, poziva se iz main.py

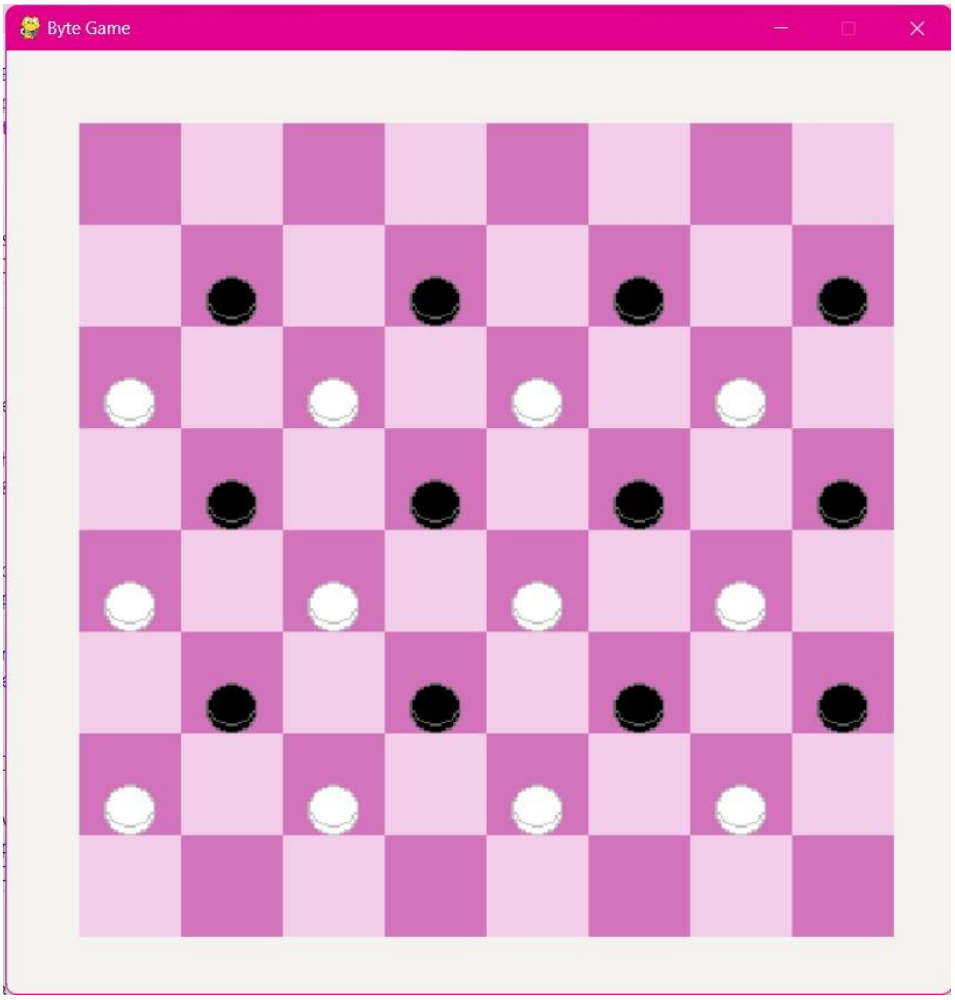


3. **Funkcije koje obezbeđuju pravljenje inicijalnog stanja problema (igre)**
   Tabla se inicijalizuje na osnovu izabranih dimenzija. Na osnovu tražene dimanzije, postavljaju se figure(bitovi) na odgovarajuće pozicije.

```
class Board:
    def __init__(self, dim, rectSize, rectStart):
        self.dim = dim
        self.board=[[(bytes([0]), 0) for _ in range(dim)] for _ in range(dim)]
        self.bit = (dim-2)*dim/2
        self.byte = self.bit/8
        self.squareSize = rectSize / dim
        self.rectStart = rectStart
        self.currentPlayer = 1
        self.fillMatrix()


    def fillMatrix(self):
        for row in range(self.dim):
            for col in range(self.dim):
                if(row != 0 and row != self.dim - 1):
                    if(row%2 == 0 and col %2 == 0 or row%2 == 1 and col%2 == 1):
                        if(row %2 == 0):
                            self.writeBit(row, col, 1, self.board[row][col][1])
                        else:
                            self.writeBit(row, col, 0, self.board[row][col][1])
```

```
def drawMatrix(self, screen):
    x_offset = 0
    y_offset = 0
    color = ()

    for row in range(self.dim):
        for col in range(self.dim):
            rect = [self.rectStart[0] + x_offset, self.rectStart[1] + y_offset, self.squareSize, self.squareSize]
            stack_offset = 0

            if(row%2 == 0 and col %2 == 0 or row%2 == 1 and col%2 == 1):
                color = (210,115,187)
                pygame.draw.rect(screen, color, rect)

                if(self.board[row][col][1]>0):
                    for (i) in range(self.board[row][col][1]):
                        if(self.readBit(row, col, i+1)):
                            bit_image = pygame.image.load('BYTE\\assets\\white.gif')
                        else:
                            bit_image = pygame.image.load('BYTE\\assets\\black.gif')

                        bit_image = pygame.transform.scale(bit_image, (self.squareSize/2, self.squareSize/2))
                        pygame.Surface.blit(screen, bit_image, (rect[0] + self.squareSize / 4, rect[1] + self.squareSize / 2 + stack_offset))
                        stack_offset -= 10 #bice druga vrednost

            else:
                color = (242,206,234)
                pygame.draw.rect(screen, color, rect)

            x_offset += self.squareSize

        x_offset = 0
        y_offset += self.squareSize
```

Stanje igre je predstavljeno matricom, koja čuva parove (BYTE, Int), gde BYTE predstavlja upravo niz figura(bitova) na jednom polju, a Integer vrednosti prestavlja broj figura na polju. U okviru byte-a, jedinicom su predstavljene bele, a nulom crne figure



4. **Funkcije koje obezbeđuju prikaz proizvoljnog stanja problema (igre)**
5. **Funkcije za unos poteza**

   Stanje se menja tako što izmenimo vrednosti u matrici, a nakon toga ih čitamo i iscrtavamo, sve na osnovu pokreta miša.

   Main.py:

```
while running:
    background = screenGame.fill((245, 243, 240))
    board.drawMatrix(screenGame)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        if event.type == pygame.MOUSEBUTTONDOWN:
            movement[0], movement[1] = pygame.mouse.get_pos()
        if event.type == pygame.MOUSEBUTTONUP:
            movement[2], movement[3] = pygame.mouse.get_pos()
            board.move(screenGame, movement, 0)

    board.drawMatrix(screenGame)
```

Board.py:

```
def move(self, screen, movement, positionFrom):

    x1, y1 = self.get_field_start(movement[0], movement[1])
    x2, y2 = self.get_field_start(movement[2], movement[3])

    row1 = int(y1 / self.squareSize)
    col1 = int(x1 / self.squareSize)
    row2 = int(y2 / self.squareSize)
    col2 = int(x2 / self.squareSize)

    isValid = self.valid_move(row1, col1, row2, col2)
    if( isValid == None):
        return

    if(self.board[row1][col1][1] == 1):
        self.writeBit(row1, col1, 0, positionFrom)
        pos = self.board[row1][col1][1]
        self.board[row1][col1] = (self.board[row1][col1][0], pos)

        self.writeBit(row2, col2, self.currentPlayer, self.board[row2][col2][1])

        self.currentPlayer = 0 if self.currentPlayer == 1 else 1
        return

#prvo procita bitove sa pozicije sa koje se pomera i cuvam ih u nizu
bits = []

#brise sa pozicije sa koje se pomera
numOfBits = self.board[row1][col1][1]
for i in range(positionFrom, numOfBits):
    bits.append(self.readBit(row1, col1, i + 1))

    self.writeBit(row1, col1, 0, i)

    pos = self.board[row1][col1][1]
    self.board[row1][col1] = (self.board[row1][col1][0], pos)

#dodaje na poziciju na koju se pomera
j=0
for i in range(positionFrom, numOfBits):
    self.writeBit(row2, col2, bits[j], numOfBits + i)
    j+=1
```

6. Funkcije koje proveravaju da li je unos poteza tačan

```
def valid_move(self, row1, col1, row2, col2):
    if(row1 == row2 or col1 == col2):
        return None
    if(row1 < 0 or row1 >= self.dim or col1 < 0 or col1 >= self.dim):
        return None
    if(row2 < 0 or row2 >= self.dim or col2 < 0 or col2 >= self.dim):
        return None
    if(self.board[row1][col1][1] == 0):
        return None
    if(self.board[row2][col2][1] == 0):
        return None

    if(self.currentPlayer == 1 and row1 % 2 != 0 or self.currentPlayer == 0 and row1 % 2 != 1):
        return None

    diag = self.diagonal(row1, col1, row2, col2)
    if(diag == None):
        return None
    else:
        return diag


def diagonal(self, row1, col1, row2, col2):
    if(row2 == row1-1 and col2 == col1-1):
        return "GL"
    elif(row2 == row1-1 and col2 == col1+1):
        return "GD"
    elif(row2 == row1+1 and col2 == col1-1):
        return "DL"
    elif(row2 == row1+1 and col2 == col1+1):
        return "DD"
```

Neka od stanja prikazana su sledećim slikama: