Лабораториска вежба 2
Slide puzzle

Изработил: Стефан Бојаџиев – 181114

Барање 1:

Бројот на полиња во редиците и колоните да биде различен (на пример 8x6).

```
#Барање 1
BOARDWIDTH = 4  # number of columns in the board
BOARDHEIGHT = 3 # number of rows in the board
TILESIZE = 80
WINDOWWIDTH = 640
WINDOWHEIGHT = 480
FPS = 30
BLANK = None
```

Со промена на соодветните променливи за висина и ширина на самата табла, го менуваме и бројот на полиња со кои располагаме.

Барање 2:

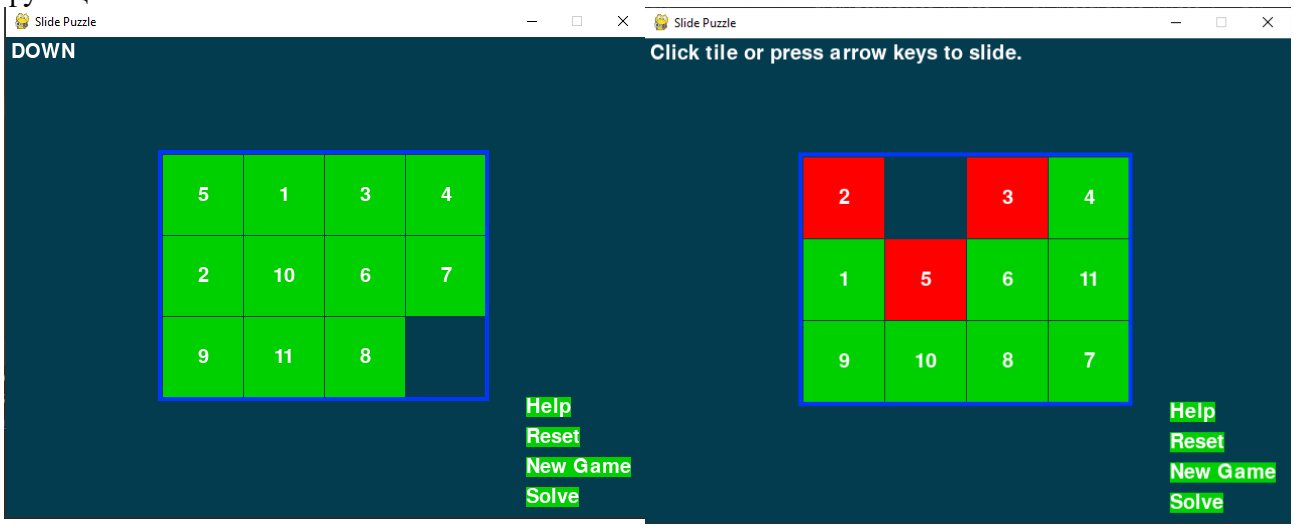
Додадете ново копче "HELP". Ако играчот кликне на копчето, на екранот треба да му се прикаже порака со можна насока за придвижување во следниот чекор. Покрај тоа, сите соседни полиња на празното треба да се обоени во црвена боја.

```
DISPLAYSURF.blit(HELP_SURF, HELP_RECT) #Барање 2
DISPLAYSURF.blit(RESET_SURF, RESET_RECT)
DISPLAYSURF.blit(NEW_SURF, NEW_RECT)
DISPLAYSURF.blit(SOLVE_SURF, SOLVE_RECT)
```

```
def get_help(myBoard, msg):
    counter = 3
    pygame.time.set_timer(pygame.USEREVENT, millis: 1000)
    before_draw = myBoard

    while True:
        drawBoard(myBoard, msg)
        pos_x, pos_y = getBlankPosition(myBoard)
        hint = None

        if isValidMove(myBoard, RIGHT) and (pos_x != len(myBoard) - 1 or pos_x == 0):
            drawTileRed(pos_x + 1, pos_y, myBoard[pos_x + 1][pos_y])
            hint = 'RIGHT'
        if isValidMove(myBoard, LEFT) and (pos_x != 0 or pos_x == len(myBoard) - 1):
            drawTileRed(pos_x - 1, pos_y, myBoard[pos_x - 1][pos_y])
            hint = 'LEFT'
        if isValidMove(myBoard, UP):
            drawTileRed(pos_x, pos_y + 1, myBoard[pos_x][pos_y + 1])
            hint = 'UP'
        if isValidMove(myBoard, DOWN):
            drawTileRed(pos_x, pos_y - 1, myBoard[pos_x][pos_y - 1])
            hint = 'DOWN'

        if hint is not None:
            drawBoard(myBoard, hint)

        for event in pygame.event.get():
            if event.type == pygame.USEREVENT:
                counter -= 1
                if counter == 0:
                    break
        else:
            pygame.display.flip()
            pygame.time.Clock().tick(60)
            continue
        break

    drawBoard(before_draw, msg)
```

```
#Барање 2
4 usages
def drawTileRed(tilex, tiley, number, adjx=0, adjy=0):

    left, top = getLeftTopOfTile(tilex, tiley)
    pygame.draw.rect(DISPLAYSURF, color: (255,0,0), rect: (left + adjx, top + adjy, TILESIZE, TILESIZE))
    textSurf = BASICFONT.render(str(number), antialias: True, TEXTCOLOR)
    textRect = textSurf.get_rect()
    textRect.center = left + int(TILESIZE / 2) + adjx, top + int(TILESIZE / 2) + adjy
    DISPLAYSURF.blit(textSurf, textRect)
```

Со додавање на соодветните функции за боење на самите полиња во различни бои и прикажување на пораки за помош при самото решавање на задачата, го добиваме резултатот илустриран со следните слики. Притоа имаме ново копче Help кое при клик ги повикува функциите.



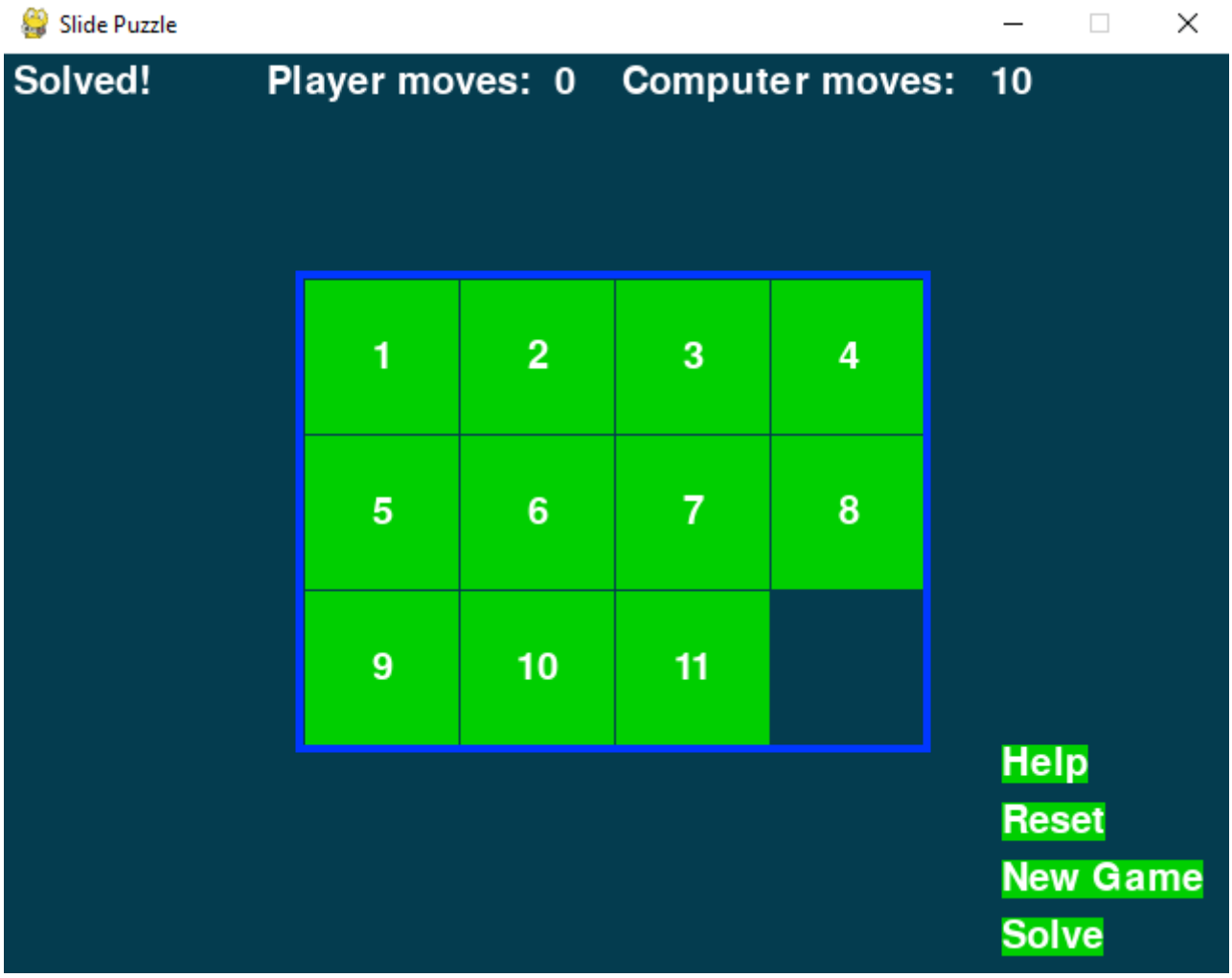
Барање 3:

Додадете бројач на поместувањата што ги прави играчот. Ако корисникот успешно ја заврши играта, информирајте го за бројот на чекори што му бил потребен да ја заврши играта и бројот на чекори што би биле потребни за решавање на играта од страна на компјутерот (имајќи ја предвид логиката што за таа цел се користи при кликнување на копчето "Solve").

```
global FPSCLOCK, DISPLAYSURF, BASICFONT, RESET_SURF, RESET_RECT, NEW_SURF, NEW_RECT, SOLVE_SURF, SOLVE_RECT, HELP_SURF, HELP_RECT,COUNT #Барање 3
```

```
if mainBoard == SOLVEDBOARD:
    #Барање 3
    moves_player = len(allMoves)
    moves_computer = COUNT
    msg = 'Solved!          Player moves:  '+str(moves_player)+'    Computer moves:   '+str(moves_computer
```

Со додавање на соодветните променливи и со имплементација на едноставен бројач ги броиме чекорите кои ги направил корисникот, додека чекорите на компјутерот се бројат при самото мешање на полињата.


Slide Puzzle
Solved!
Player moves: 0
Computer moves: 10
1
2
3
4
5
6
7
8
9
10
11
Help
Reset
New Game
Solve