ЛАБОРАТОРИСКА ВЕЖБА 2
02.12.2020

ПРОГРАМИРАЊЕ НА ВИДЕО ИГРИ И СПЕЦИЈАЛНИ ЕФЕКТИ

Slide puzzle game

Професор*:* д-р Катарина Тројачанец Динева
Студент*:* Кирил Зеленковски

0. ***Барање 1***

Бројот на полиња во редиците и колоните да биде различен (на пример 8 на 6)

```
13  ##########################################################################################################
14  BOARDWIDTH = 4  # number of columns in the board is changed to 4
15  BOARDHEIGHT = 3  # number of rows in the board is changed to 3
16  ##########################################################################################################
```

1. ***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.

Новото копче го правам на сличен начин како што се првобитните копчиња:
Reset, **New Game** и **Solve**

Сите три копчиња е потребно да се нацртаат користејќи ја функцијата **DISPLAYSURF.blit().** Оваа функија на влез прима тоа што сите три кочиња имаат заедничко (две променливи):

- **Surface** променлива (**RESET_SURF, NEW_SURF, SOLVE_SURF**)
- **Rectangle** променлива (**RESET_RECT, NEW_RECT, SOLVE_RECT**)

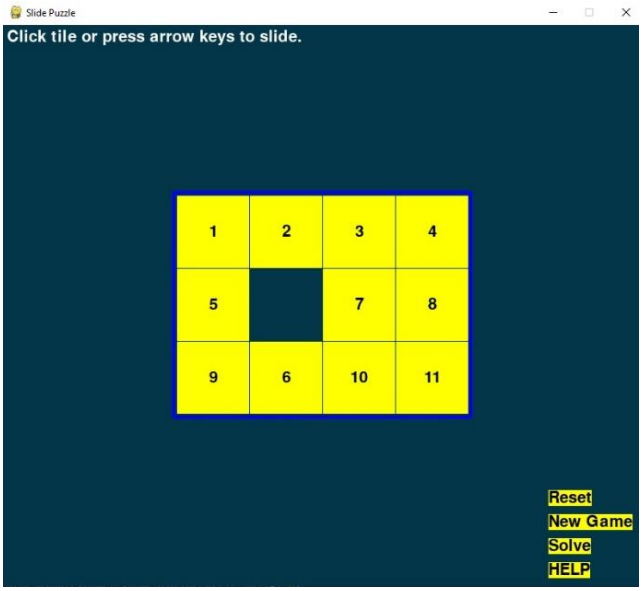
Со таа цел дефинирам две нови променливи за новото копче (многу важно и двете ги поставувам да бидат „глобални“ бидејќи ги користиме надвор од функцијата main())

```
67      global HELP_SURF, HELP_RECT, COMBINE_SURF, COMBINE_RECT, PLAYER_SURF, PLAYER_RECT
68      HELP_SURF, HELP_RECT = makeText('HELP', TEXTCOLOR, TILECOLOR, WINDOWWIDTH - 120, WINDOWHEIGHT - 30)
```

Со цел да се исцрта копчето потребно е да се употреби погоре споменатата функција која како влезни параметри и ги задаваме новите променливи:

```
287     DISPLAYSURF.blit(RESET_SURF, RESET_RECT)
288     DISPLAYSURF.blit(NEW_SURF, NEW_RECT)
289     DISPLAYSURF.blit(SOLVE_SURF, SOLVE_RECT)
290     # Цртај HELP копче
291     DISPLAYSURF.blit(HELP_SURF, HELP_RECT)
```

Со овие команди до оваа фаза во прозорецот од играта ќе гледаме копче за помош, меѓутоа истото нема никаков “*listener*” и нема повратна функција.



Со цел да оневозможиме некаква функција на копчето “**HELP**” мора да поставиме некој вид на “*listener*” кој кога копчето ќе се кликне ќе ги исцрта соседните полиња црвени. (Тоа го правиме во 97 ред каде правиме проверка и потоа во следниот повикуваме ф-ја)

```
89          checkForQuit()
90          for event in pygame.event.get():  # event handling loop
91              if event.type == MOUSEBUTTONUP:
92                  spotx, spoty = getSpotClicked(mainBoard, event.pos[0], event.pos[1])
93
94                  if (spotx, spoty) == (None, None):
95                      # check if the user clicked on an option button
96                      if RESET_RECT.collidepoint(event.pos):
97                          resetAnimation(mainBoard, allMoves)  # clicked on Reset button
98                          allMoves = []
99                      elif NEW_RECT.collidepoint(event.pos):
100                         mainBoard, solutionSeq = generateNewPuzzle(random.randint(1, 100))  # clicked on New Game button
101                         allMoves = []
102                     elif SOLVE_RECT.collidepoint(event.pos):
103                         resetAnimation(mainBoard, solutionSeq + allMoves)  # clicked on Solve button
104                         allMoves = []
105                     elif HELP_RECT.collidepoint(event.pos):  ################################################## HELP
106                         help_timer(mainBoard, msg)  # Повикуваме главна функција за помош
```

Слично како сите event *listeners* за другите 3 копчиња таков е и за 4-тото копче. Задачата иако беше само да се обојат црвени можните чекори, јас сакав да имитирам нешто слично на игра и да има како тајмер за колку време помошта ќе биде на екран и ова го реализирав внатре во функцијата help_timer().

За овој тајмер користам две функции:

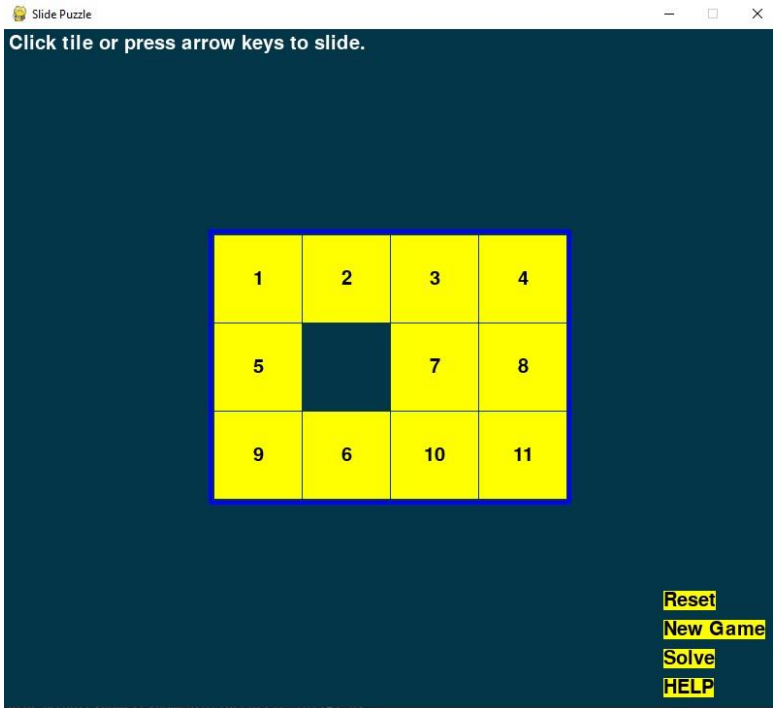
- Функција 1 **drawHelpTile()**: За цртање на tiles црвени. Идејата е да имаме посебна функција но и обичната е океј ако се модифицира да има некој влезен параметар за боја бидејќи обичната drawTile() ги бои со предефинирана боја

```
373 # Функија 1: Цртање на tiles црвени, може да се модифицира е обичната со цел да ставаме боја како аргумент исто така
374 def drawHelpTile(tilex, tiley, number, adjx=0, adjy=0):
375     left, top = getLeftTopOfTile(tilex, tiley)
376     # Променуваме бојата да биде "red" од главната функција сите други редови се исти од обичната drawTile()
377     pygame.draw.rect(DISPLAYSURF, pygame.Color('red'), (left + adjx, top + adjy, TILESIZE, TILESIZE))
378     textSurf = BASICFONT.render(str(number), True, TEXTCOLOR)
379     textRect = textSurf.get_rect()
380     textRect.center = left + int(TILESIZE / 2) + adjx, top + int(TILESIZE / 2) + adjy
381     DISPLAYSURF.blit(textSurf, textRect)
382
```

- Функција 2

  help_timer(): Главна функција, која бои коцки во околина (десно, лево, горе доле доколку се валидни црвени) и во исто време почнува тајмер од 5 секунди со цел да имитира време за помош

```python
# Функија 2: Главна функција за помош
def help_timer(myBoard, msg):
    # 1 Чекор: Дефинирај ги сите помошни променливи
    counter = 5  # Бројач за секунди (иницијално 5)
    text = "Help time: " + str(counter).rjust(3) if counter > 0 else 'Time up!'
    pygame.time.set_timer(pygame.USEREVENT, 1000)
    font = pygame.font.SysFont('Consolas', 30)  # Фонт кој сакаме да се рендерира
    before_draw = myBoard  # Зачувај ја првата сотојба пред цртање за секој случај (force of habit не е нужно)

    # Главен циклус за бројач
    while True:
        drawBoard(myBoard, msg)
        # 2 Чекор: Земи кординати на празно поле
        pos_x, pos_y = getBlankPosition(myBoard)
        # 3: Бој ги сите полиња кој се достапни; Во најдобар случај можни се 4 полиња, во најлош 3
        if (isValidMove(myBoard, RIGHT) and pos_x != (len(myBoard) - 1)) \
                or pos_x == 0:  # Поле ДЕСНО од нас
            drawHelpTile(pos_x + 1, pos_y, myBoard[pos_x + 1][pos_y])

        if (isValidMove(myBoard, LEFT) and pos_x != 0) \
                or pos_x == (len(myBoard) - 1):  # Поле ЛЕВО од нас
            drawHelpTile(pos_x - 1, pos_y, myBoard[pos_x - 1][pos_y])

        if isValidMove(myBoard, UP):  # Поле НАД од нас
            drawHelpTile(pos_x, pos_y + 1, myBoard[pos_x][pos_y + 1])

        if isValidMove(myBoard, DOWN):  # Поле ДОЛУ од нас
            drawHelpTile(pos_x, pos_y - 1, myBoard[pos_x][pos_y - 1])

        # 4 Чекор: Креирај timer (да брои до кога ќе стои помоштач); 5 секунди
        for e in pygame.event.get():
            if e.type == pygame.USEREVENT:
                counter -= 1  # Намали го бројачот
                text = "Move one of the red squares. Help time: " + str(counter).rjust(3) \
                    if counter > 0 else 'Time up!'  # Ако дојде 0=Time up!
            if text == 'Time up!':  # Доколку си на 0-та секунда изгаси го циклусот врати се назад во главниот циклус
                DISPLAYSURF.blit(font.render(text, True, (255, 255, 255)),
                                 (32, 48))  # Кординати за испишување x:32, y:48
                pygame.display.flip()
                pygame.time.Clock().tick(3)  # Повикај часовнк за секоја секунда
                break
        else:
            DISPLAYSURF.blit(font.render(text, True, (255, 255, 255)), (32, 48))  # Кординати за испишување x:32, y:48
            pygame.display.flip()
            pygame.time.Clock().tick(60)  # Повикај часовнк за секоја секунда
            continue
        break
    # Цртај ја таблата секој пат со цел да стојат сите квадратчиња
    drawBoard(before_draw, msg)
```



Слика 1: Додадено HELP копче

Овој бројач одбројува (5 – 1, кога доаѓа 0 вика “*Time up*!”) и се враќа на Слика 1.

Слика 2,3: По кликнување на HELP копчето

2. ***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").

Како што пишува во барањето ова решение ја согледува логиката на “Solve” потребни counters за:

- Бројот на чекори направени од почеток до решавање
- Бројот на чекори од страна на компјутер

Верзија 1:

Бројач на чекори од компјутер

Овие се добиваат како збир од листата **allMoves** (каде се наоѓаат во обратен редослед сите преместување од играта што играчот ги прави) и листата од чекори **solutionSeq** каде се наоѓаат чекорите од тој “random scrabble” што самиот компјутер ги прави кога почнува играта со цел да ја размести од почетната состојба.

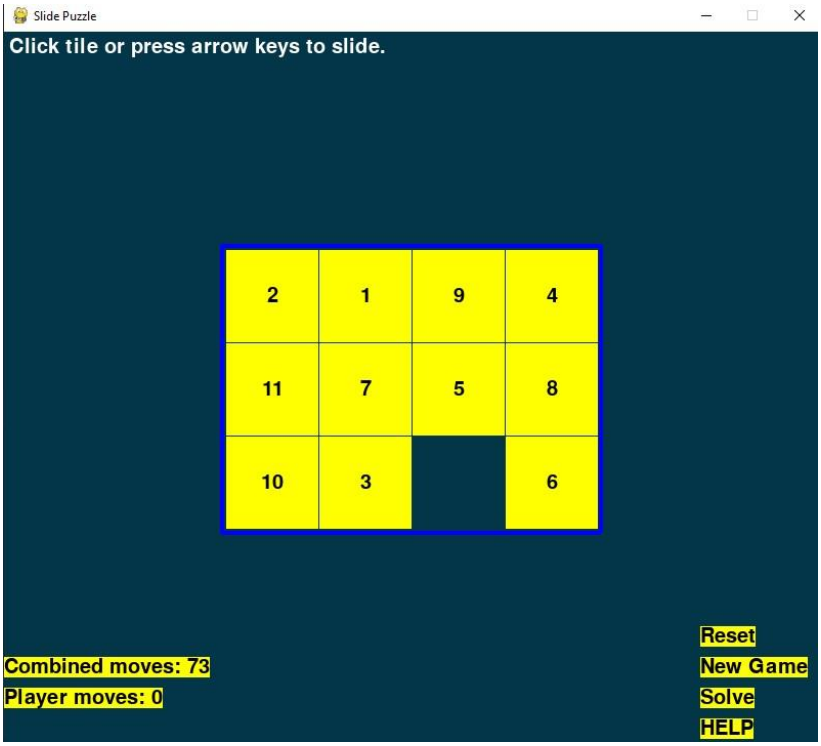
Бројач на чекори од играч

```
141 ############################################################################################################ БАРАЊЕ 3. ВЕРЗИЈА 1
142         text_com = f'Combined moves: {len(allMoves) + len(solutionSeq)}'    # текст за сите чекори
143         COMBINE_SURF, COMBINE_RECT = makeText(text_com, TEXTCOLOR, TILECOLOR, WINDOWWIDTH - 800, WINDOWHEIGHT - 90)
144
145         text_player = f'Player moves: {len(allMoves)}'  # текст за чекори од играч
146         PLAYER_SURF, PLAYER_RECT = makeText(text_player, TEXTCOLOR, TILECOLOR, WINDOWWIDTH - 800, WINDOWHEIGHT - 60)
147
148         DISPLAYSURF.blit(COMBINE_SURF, COMBINE_RECT)
149         DISPLAYSURF.blit(PLAYER_SURF, PLAYER_RECT)
150 ############################################################################################################
151
```

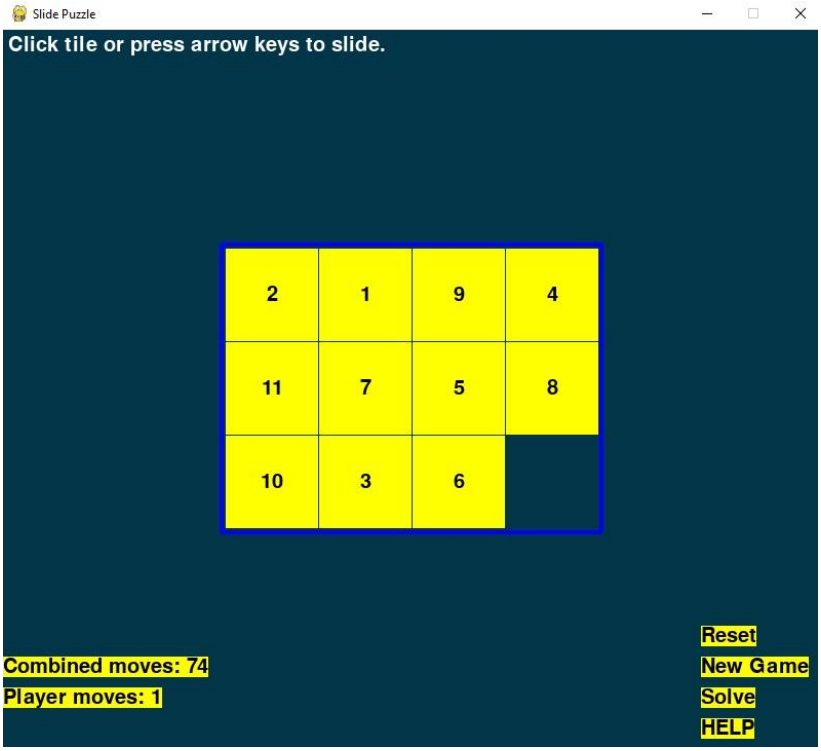
Овие се добиваат со читање на должината на **allMoves** листата.

И двете се претставуваат како текст со surface

идејата за вака е слична од копчето за “**HELP**”
може и со обичен фонт но, вака повеќе ми се допаѓаат и брзо ги направив



Слика 4: Почетна состојба на бројачи



Слика 5: Состојба на бројачи по едно поместување

Верзија 2:

Бидејќи не бев сигурен дали треба да се испише текстот на крај или цело време да се прикажуваат го модифицирав кодот и го прикачив со оваа втора верзија. Тука ги испишувам чекорите само кога е решена играта.

За оваа верзија сепак бројач беше нужен бидејќи при ресетирање на играта сите чекори од allmoves листата се бришат (од каде и ми дојде логиката дека тоа што го има во барањето има смисла со бројач а не само со len(allmoves)).

Бројачот не е класичен туку секој пат кога се додава чекор, го иницирам да биде еднаков на должината од листата allmoves во тој момент.

```
135         if slideTo:
136             slideAnimation(mainBoard, slideTo, 'Click tile or press arrow keys to slide.', 8)  # show slide on screen
137             makeMove(mainBoard, slideTo)
138             allMoves.append(slideTo)  # record the slide
139             my_moves = len(allMoves)  # чувај си чекорите до тогаш за да не ги изгубиш
140
```

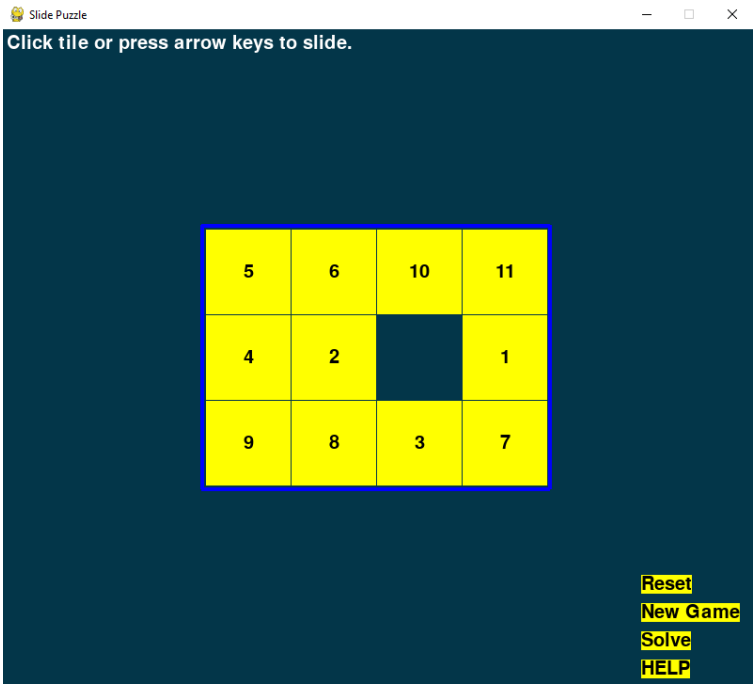
Оваа верзија малку покомплицирано ја направив од што треба но навистина сакав да видам дали можам и на крајот на играта да испишувам.

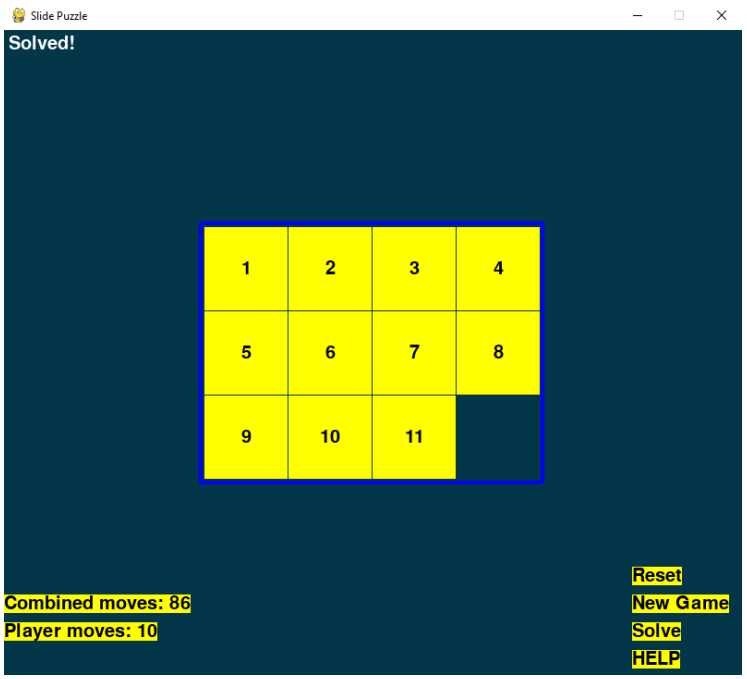
```
77      while True:  # main game loop
78          slideTo = None  # the direction, if any, a tile should slide
79          msg = 'Click tile or press arrow keys to slide.'  # contains the message to show in the upper left corner.
80          text_lists = []
81          if mainBoard == SOLVEDBOARD:
82              msg = 'Solved!'
83              # Додади листа од текстови за копчиња
84              text_com = f'Combined moves: {my_moves + len(solutionSeq)}'  # текст за чекори од компјутер
85              text_player = f'Player moves: {my_moves}'  # текст за чекори од играч
86              text_lists.append(text_com)
87              text_lists.append(text_player)
88
89          drawBoard(mainBoard, msg, text_lists)
90
```

Правам листа од двете пораки и истите ги праќам во главната функција бидејќи секогаш се повикува. Тука само исто како во верзија 1 ги испишувам повторно.

```python
def drawBoard(board, message, texts=[]):
    DISPLAYSURF.fill(BGCOLOR)
    if message:
        textSurf, textRect = makeText(message, MESSAGECOLOR, BGCOLOR, 5, 5)
        DISPLAYSURF.blit(textSurf, textRect)

        # Ако е пораката SOLVED испиши ги чекорите потребни и од играчот и од компјутерот
        if message == 'Solved!':
            text_com = texts[0]
            COMBINE_SURF, COMBINE_RECT = makeText(text_com, TEXTCOLOR, TILECOLOR, WINDOWWIDTH - 800, WINDOWHEIGHT - 90)

            text_player = texts[1]
            PLAYER_SURF, PLAYER_RECT = makeText(text_player, TEXTCOLOR, TILECOLOR, WINDOWWIDTH - 800, WINDOWHEIGHT - 60)

            DISPLAYSURF.blit(COMBINE_SURF, COMBINE_RECT)
            DISPLAYSURF.blit(PLAYER_SURF, PLAYER_RECT)
            pygame.display.update()
```

Резултатот е тоа што при играње не ги гледаме но ако ја решиме играта (или одиме со solve) ќе ги видиме сите чекори потребни.





Слика 6, 7: Состојба без чекори (лево). Состојба по 10 чекори и кликнување на Solve (десно)