ЛАБОРАТОРИСКА ВЕЖБА 3
30.12.2020

ПРОГРАМИРАЊЕ НА ВИДЕО ИГРИ И СПЕЦИЈАЛНИ ЕФЕКТИ

Simulate puzzle game

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

1. *Барање 1*

Секвенцата што играчот треба да ја повтори потребно е да биде комплетно различна од таа во претходната итерација на играта така што секој елемент од секвенцата ќе биде додаден со случаен избор од можните полиња.

Ова барање се врти околу манипулација на главната листа од чекори. По default таа секој пат кога играчот ќе погоди чекор ја зголемува листата со тоа што рандом прикачува елемент од можните бои (коцки со боја кои се црвено, зелено, жолто, плаво). И потоа од 0 го прикажува тој pattern, елемент по елемент.

Јас прво дефинирав бројач, counter кој го поставувам иницијално на 1 и потоа секој пат кога играчот ќе погоди боја, следната итерација ја сетирам листата pattern = [] да биде празна и во јамка од ново додавам рандом елементи од 0 до counter (бројачот на погодени доаѓа). И секој пат кога ќе погодам го зголемувам бројачот и автоматски ја полнам листата со нови елементи.

Додавање на променливата counter = 1, пред јамката за главната игра.

```
67      # step counter
68      counter = 1
69      while True: # main game loop
70          clickedButton = None # button that was clicked (set to YELLOW, RED, GREEN, or BLUE)
71          DISPLAYSURF.fill(bgColor)
```

Потоа кога сме погодиле чекаме да ни се зголеми резултатот и да се промени секвенцата. Тука ја поставуваме листата да биде празна, потоа ја двжиме јамката од 0 до бројачот и се доделуваме вредности рандом за таа секвенца (ред 102-104).

```
98          if not waitingForInput:
99              # play the pattern
100             pygame.display.update()
101             pygame.time.wait(1000)
102             pattern = []
103             for c in range(0, counter):
104                 pattern.append(random.choice((YELLOW, BLUE, RED, GREEN)))
105             for button in pattern:
106                 flashButtonAnimation(button)
107                 pygame.time.wait(FLASHDELAY)
108             waitingForInput = True
109         else:
110             # wait for the player to enter buttons
111             if clickedButton and clickedButton == pattern[currentStep]:
112                 # pushed the correct button
113                 flashButtonAnimation(clickedButton)
114                 currentStep += 1
115                 lastClickTime = time.time()
```

Потоа кога ќе се стигне до истиот број на чекори при повторување од играчот се зголемува бројачот за листата (ред 124).

```
109         else:
110             # wait for the player to enter buttons
111             if clickedButton and clickedButton == pattern[currentStep]:
112                 # pushed the correct button
113                 flashButtonAnimation(clickedButton)
114                 currentStep += 1
115                 lastClickTime = time.time()
116 
117 
118                 if currentStep == len(pattern):
119                     # pushed the last button in the pattern
120                     changeBackgroundAnimation()
121                     score += 1
122                     waitingForInput = False
123                     currentStep = 0 # reset back to first step
124                     counter += 1
```

Резултатот од една итерација (видео линк подолу) на мојот компјутер од ваквиот код беше следна:

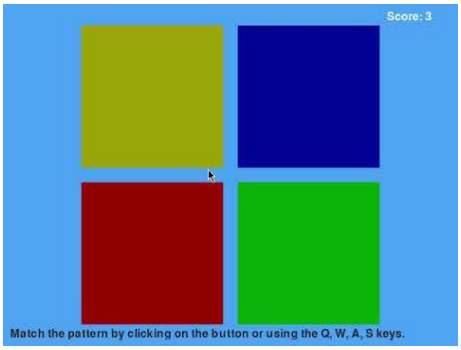
1 итерација: црвена
2 итерација: зелена, црвена
3 итерација: зелена, жолта, жолта
...

Исто така доста важно, откако ќе згрешиме и ќе имаме **gameOver** треба исто така да се ресетира бројачот на 1 повторно за да не продолжи нова игра од пример нивото или итерацијата до која сме стигнале.

```
125             elif (clickedButton and clickedButton != pattern[currentStep]) or (currentStep
126                 # pushed the incorrect button, or has timed out
127                 gameOverAnimation()
128                 # reset the variables for a new game:
129                 pattern = []
130                 currentStep = 0
131                 waitingForInput = False
132                 score = 0
133                 counter = 1  # reset counter to 1
134                 pygame.time.wait(1000)
135                 changeBackgroundAnimation()
```

Бидејќи сметав дека подобро ќе биде визуелно да се види промената место screenshots снимив видео и е достапно тука:

https://youtu.be/T98T4Va1STw



2. Барање 2

Намалете ја вредноста на timeout (почнувајќи од 5 до 3) на секои 10 промени на секвенцата. По достигнување на вредност 3, повеќе не се прави намалување.

Најпрво ја поставуваме променливата TIMEOUT на 5 во самиот почеток од кодот и ја додаваме како глобална променлива за да избегнеме конфилкт за доделување на вредност пред да се дефинира бидејќи ако сакаме да и смениме вредност а не е глобална ќе имаме грешка (линија 16 и линија 42).

Главна јамка:

```
 9    FPS = 30
10    WINDOWWIDTH = 640
11    WINDOWHEIGHT = 480
12    FLASHSPEED = 500 # in milliseconds
13    FLASHDELAY = 200 # in milliseconds
14    BUTTONSIZE = 200
15    BUTTONGAPSIZE = 20
16    TIMEOUT = 5 # seconds before game over if no button is pushed.
```

Дефинирање како глобална:

```
40
41  def main():
42      global FPSCLOCK, DISPLAYSURF, BASICFONT, BEEP1, BEEP2, BEEP3, BEEP4, TIMEOUT
43
44      pygame.init()
45      FPSCLOCK = pygame.time.Clock()
46      DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
47      pygame.display.set_caption('Simulate')
```

Потоа правиме промена во главната јамка каде идејата е следна ако должината на чекорите (секвенцата) која играчот треба да ја повтори е делива со 10 (тоа е случај на 10, 20, 30 .. должина) и TIMEOUT е поголем од 2 (тоа е сучај за 5, 4, 3) тогаш, намали го TIMEOUT. Со тоа намалуваме на 10 чекори од 4 на 5, на 20 од 4 на 3 и потоа секогаш е 3.

```
125
126                 if len(pattern) % 10 == 0 and TIMEOUT > 2:
127                     TIMEOUT -= 1
128
```

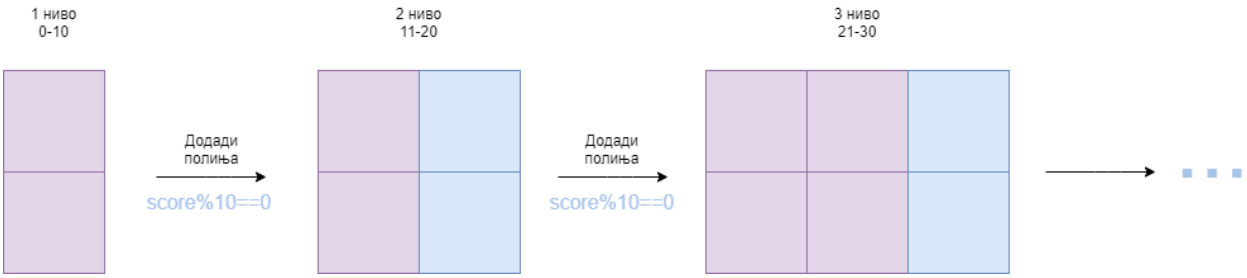
Ова се случува во самата јамка и после промената на 3 нема да влезе во овој услов и вредноста за TIMEOUT останува 3.

3. Барање 3

Да се зголеми матрицата од полиња за едно по двете димензии (ширина и висина) секогаш кога ќе се достигне резултат со вредност што цел број пати содржи 10 (10, 20, 30, ...).

За ова барање проверката е слична како претходното барање, кога резултатот % 10 ќе биде 0 тогаш ќе додаваме едно поле во ширина и едно во висина.

Бидејќи повеќе бои ќе беа неопходни барањето го правев за 0-10 резултат, 11-20, 21-30 и потем 30 останува истата матрица. Визуелно ги прикажувам додавањата на полињата со следниот цртеж (правен во draw.io)



Прво рештката ја намалувам на 2 полиња и потоа зголемувам по 2 полиња (едно висина едно ширина). Причина зошто намалувам рештката е бидејќи функционалноста е иста независно дали започнувам од 1x2 или од 2x2, само повеќе бои ќе треба доколку полињата се од 2x2 (3то ниво вака ќе биде 2x4).

Чекорите се слични како во работата на час, променливи кој ги менувам се следни:

- **XMARGIN:** оваа променлива е од големо значење треба да ја менуваме за да може да собира 1 колона, 2 колони, 3 колони, ..
- **color_list:** ова е листа од торки, каде секоја торка е боја.
    - 1 ниво: Иницијално е поставена на 2 бои – YELLOW, RED
    - 2 ниво: Потоа се проширува оваа листа за уште 2 бои – YELLOW, RED, BLUE, GREEN
    - 3 ниво: И на крај се сите 6 – YELLOW, RED, BLUE, GREEN, ORANGE, PURPLE
- **level:** ова е обичен цел број кој го чувам за подоцнежни проверки во самите функции го подавам како аргумент и исто така го користам за испиушување на нивото на самиот екран слично како поените собрани

NOTE: За ова барање при тестирање морав да ја сменам промената од барање 1 за рандом иницирање на секвенцата бидејќи тешко станување за играње. И исто така при тестирање користам помали резултати за нивоата (1 ниво: 0-3, 2 ниво: 4-6, 3 ниво: 7-..). Во финалниот код прикачен ќе ги модифицирам на 10, 20, 30.

Иницијални чекори:

Менување на иницијалните големи на прозорецот (800 на 500):

```
6   import ...
8
9   FPS = 30
10  WINDOWWIDTH = 800
11  WINDOWHEIGHT = 500
12  FLASHSPEED = 500 # in milliseconds
13  FLASHDELAY = 200 # in milliseconds
14  BUTTONSIZE = 100
15  BUTTONGAPSIZE = 20
16  TIMEOUT = 5 # seconds before game over if no button is pushed.
```

Додавање бои (обични и светли за трепкање) за сите копчиња:

```
18  #               R    G    B
19  WHITE       = (255, 255, 255)
20  BLACK       = (  0,   0,   0)
21  BRIGHTRED   = (200,   0,   0)  # 1ва боја: RED1
22  RED         = (100,   0,   0)  # 1ва посветла боја: RED2
23  BRIGHTGREEN = (  0, 200,   0)  # 2ра боја: GREEN1
24  GREEN       = (  0, 100,   0)  # 2ра посветла боја: GREEN2
25  BRIGHTBLUE  = (  0,   0, 200)  # 3та боја: BLUE1
26  BLUE        = (  0,   0, 100)  # 3та посветла боја: BLUE2
27  BRIGHTYELLOW = (200, 200,   0)  # 4та боја: YELLOW1
28  YELLOW      = (100, 100,   0)  # 4та посветла боја: YELLOW2
29  BRIGHTPURPLE = (186,  85, 211)  # 5та боја: PURPLE1
30  PURPLE      = ( 75,   0, 130)  # 5та посветла боја: PURPLE2
31  BRIGHTORANGE = (255, 165,  0)   # 6та боја: ORANGE1
32  ORANGE      = (255, 140,  0)   # 6та посветла боја: ORANGE2
33  DARKGRAY    = ( 40,  40,  40)
34  bgColor = BLACK
```

Дефинирање на почетна маргина (1 на 2 матрица почетна) како и дефинирање на повеќето од променливите како глобални за да овозможиме подоцнежен пристап:

```
XMARGIN = int((WINDOWWIDTH - (1 * BUTTONSIZE) - BUTTONGAPSIZE) / 2)
YMARGIN = int((WINDOWHEIGHT - (2 * BUTTONSIZE) - BUTTONGAPSIZE) / 2)


def main():
    global FPSCLOCK, DISPLAYSURF, BASICFONT, BEEP1, BEEP2, BEEP3, BEEP4, BEEP5, BEEP6
    global TIMEOUT, XMARGIN, level, color_list
    global YELLOWRECT, REDRECT, BLUERECT, GREENRECT, ORANGERECT, PURPLERECT

    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    pygame.display.set_caption('Simulate')
```

Дефинирање на правоаголник со текст за score, и за level, додавање на сите звуци:

```

    # load the sound files
    BEEP1 = pygame.mixer.Sound('beep1.ogg')
    BEEP2 = pygame.mixer.Sound('beep2.ogg')
    BEEP3 = pygame.mixer.Sound('beep3.ogg')
    BEEP4 = pygame.mixer.Sound('beep4.ogg')
    BEEP5 = pygame.mixer.Sound('beep4.ogg')
    BEEP6 = pygame.mixer.Sound('beep4.ogg')

    # Initialize some variables for a new game
    pattern = []
    currentStep = 0
    lastClickTime = 0
    score = 0
    # when False, the pattern is playing. when True, waiting for the player to click a colored button:
    waitingForInput = False
    # step counter
    counter = 1
    # list of colors
    color_list = [YELLOW, RED]
    # level
    level = 0

```

До сега сите дефиниции се фокусирани на имплементација на самите копчиња. Сега веќе започнуваме со самите модификации на нивоа. Прво проверуваме дали сме на ниво потоа ако идеме во ново ниво:

- Дефинираме ново ниво
- Пресметуваме маргина нова
- Додавање на новите бои во листата (color_list)
- Редефинирање на копчињата (ова го правам за поубаво да ги поместува при промена на маргина во самиот почетен код тие се дефинираат надвор но вака ќе бидат статични јас сакав да ги направам подинамични со тоа што ќе се менуваат на секое ниво)

Еве ги промените во самиот код:

```
        # Барање 3
        # if 11 <= score <= 20:
        if 3 <= score <= 5:
            level = 1
            XMARGIN = int((WINDOWWIDTH - (2 * BUTTONSIZE) - BUTTONGAPSIZE) / 2)  # 1+1 = 2
            color_list.append(GREEN)
            color_list.append(BLUE)
            YELLOWRECT = pygame.Rect(XMARGIN, YMARGIN, BUTTONSIZE, BUTTONSIZE)
            REDRECT = pygame.Rect(XMARGIN, YMARGIN + BUTTONSIZE + BUTTONGAPSIZE, BUTTONSIZE, BUTTONSIZE)
            BLUERECT = pygame.Rect(XMARGIN + BUTTONSIZE + BUTTONGAPSIZE, YMARGIN, BUTTONSIZE, BUTTONSIZE)
            GREENRECT = pygame.Rect(XMARGIN + BUTTONSIZE + BUTTONGAPSIZE, YMARGIN + BUTTONSIZE + BUTTONGAPSIZE,
                                    BUTTONSIZE, BUTTONSIZE)

        # elif 21 <= score <= 30:
        elif 6 <= score <= 8:
            level = 2
            XMARGIN = int((WINDOWWIDTH - (3 * BUTTONSIZE) - BUTTONGAPSIZE) / 2)  # 2+1 = 3
            color_list.append(ORANGE)
            color_list.append(PURPLE)
            YELLOWRECT = pygame.Rect(XMARGIN, YMARGIN, BUTTONSIZE, BUTTONSIZE)
            REDRECT = pygame.Rect(XMARGIN, YMARGIN + BUTTONSIZE + BUTTONGAPSIZE, BUTTONSIZE, BUTTONSIZE)
            BLUERECT = pygame.Rect(XMARGIN + BUTTONSIZE + BUTTONGAPSIZE, YMARGIN, BUTTONSIZE, BUTTONSIZE)
            GREENRECT = pygame.Rect(XMARGIN + BUTTONSIZE + BUTTONGAPSIZE, YMARGIN + BUTTONSIZE + BUTTONGAPSIZE,
                                    BUTTONSIZE, BUTTONSIZE)
            ORANGERECT = pygame.Rect(XMARGIN + BUTTONSIZE * 2 + BUTTONGAPSIZE * 2, YMARGIN, BUTTONSIZE,
                                     BUTTONSIZE)
            PURPLERECT = pygame.Rect(XMARGIN + BUTTONSIZE * 2 + BUTTONGAPSIZE * 2,
                                     YMARGIN + BUTTONSIZE + BUTTONGAPSIZE, BUTTONSIZE, BUTTONSIZE)
```

Потоа останува да се додат на одредени места исцртувањата на копчињата а тоа е ф-јата draw_buttons(), која ја модифицирав така што прима аргумент level.

<u>1 ниво:</u> жолта, црвена

<u>2 ниво:</u> жолта, црвена, плава, зелена

<u>3 ниво:</u> сите 6

```
def drawButtons(level):
    if level == 0:
        pygame.draw.rect(DISPLAYSURF, YELLOW, YELLOWRECT)
        pygame.draw.rect(DISPLAYSURF, RED,    REDRECT)
    elif level == 1:
        pygame.draw.rect(DISPLAYSURF, YELLOW, YELLOWRECT)
        pygame.draw.rect(DISPLAYSURF, RED, REDRECT)
        pygame.draw.rect(DISPLAYSURF, BLUE,   BLUERECT)
        pygame.draw.rect(DISPLAYSURF, GREEN,  GREENRECT)
    elif level > 1:
        pygame.draw.rect(DISPLAYSURF, YELLOW, YELLOWRECT)
        pygame.draw.rect(DISPLAYSURF, RED, REDRECT)
        pygame.draw.rect(DISPLAYSURF, BLUE, BLUERECT)
        pygame.draw.rect(DISPLAYSURF, GREEN, GREENRECT)
        pygame.draw.rect(DISPLAYSURF, ORANGE, ORANGERECT)
        pygame.draw.rect(DISPLAYSURF, PURPLE, PURPLERECT)
```

Потоа на одредени места само остана да се додадат опциите за притиснување на буква од тастатура за копче и некаде ако требаше да се додадат копчињата.

Притоа играта кога завршува треба да се ресетираат сотојбите на почетни:

- Маргината на првобитна состојба
- Нивото (level) на 0
- Листата од бои само на првите две бои
- Секвенцата на празна

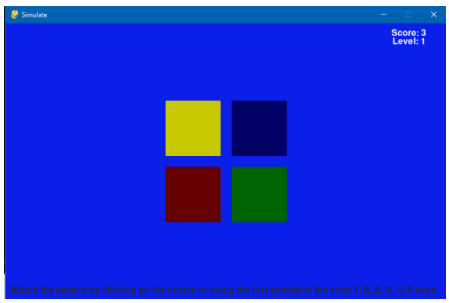
```
179             elif (clickedButton and clickedButton != pattern[currentStep]) or (currentStep != 0 and time.time() - TIMEOUT > lastClickTime):
180                 # pushed the incorrect button, or has timed out
181                 gameOverAnimation()
182                 # reset the variables for a new game:
183                 pattern = []
184                 currentStep = 0
185                 waitingForInput = False
186                 score = 0
187                 counter = 1  # reset counter to 1
188                 level = 0
189                 color_list = [YELLOW, RED]
190                 XMARGIN = int((WINDOWWIDTH - (1 * BUTTONSIZE) - BUTTONGAPSIZE) / 2)
191                 YELLOWRECT = pygame.Rect(XMARGIN, YMARGIN, BUTTONSIZE, BUTTONSIZE)
192                 REDRECT = pygame.Rect(XMARGIN, YMARGIN + BUTTONSIZE + BUTTONGAPSIZE, BUTTONSIZE, BUTTONSIZE)
193                 pygame.time.wait(1000)
194                 changeBackgroundAnimation()
195
```

За поубаво да се види имплементацијата на ова барање снимив видео кое е достапно на следниот линк:

https://youtu.be/TGrtMdM-csA

Притоа ова се неколку screenshots од играта низ првите 3 нивоа (со 6 место со 10 бидејќи полесно беше за тестирање). Промените за %10 ги имам наведено во кодот кој го прикачувам може и со %10 и со фиксни вредности и двете работеа.

1 Level



2 Level

3 Level