Simulate лабараториска вежба

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

```
for i in range(pattern_len):
    pattern.append(random.choice((YELLOW, BLUE, RED, GREEN)))
pattern.append(random.choice((YELLOW, BLUE, RED, GREEN)))
```

Во main loop во делот кога се додава копче во секвенцата го менуваме кодот така што ќе го исполнува барањето. Најпрви ја зечувуваме должината на секвенцата потоа ја чистиме секвенцата (ја поставуваме на празна низа) и потоа додаваме копчиња по случаен избор онолку пати колку што е потребно (должината на секвенцата + еден)

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.

```
# Change for requirement 2:
# TIMEOUT = 4 # seconds before game over if no button is pushed.
```

```
# Change for requirement 2:
TIMEOUT = 5  # seconds before game over if no button is pushed.
```

```
if score % 10 == 0 and score != 0 and TIMEOUT > 3:
    TIMEOUT -= 1
```

Глобалната константа TIMEOUT ќе ја избришеме (искоментираме) и ќе поставиме локална променлива TIMEOUT во main функцијата со почетна вредност 5. Потоа во main loop додаваме код кој проверува дали score е делив со 10 не е еднаков со 0 и TIMEOUT е поголем од 3 ако сите овие услови се исполнети тогаш е потребно да ја намалиме вредноста на TIMEOUT за 1. Со тоа се задоволува ова 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, ...).

Најпрво правиме промени во константите: го зголемуваме прозорот со цел да може да ги собере дополнителните копчиња, маргините по x и y ги поставуваме да бидат

статички додека големината на копчињата ќе ја пресметуваме динамички во зависност од бројот на копчиња

```
# Change for requirement 3:
# WINDOWWIDTH = 640
# WINDOWHEIGHT = 480
WINDOWWIDTH = 960
WINDOWHEIGHT = 720
# BUTTONSIZE = 200

# Change for requirement 3:
# XMARGIN = int((WINDOWWIDTH - (2 * BUTTONSIZE) - BUTTONGAPSIZE) / 2)
# YMARGIN = int((WINDOWHEIGHT - (2 * BUTTONSIZE) - BUTTONGAPSIZE) /
2)
XMARGIN = 165
YMARGIN = 60
```

Во main функцијата додаваме глобална променлива која ќе ја претставува листата од копчиња

```
def main():
    global FPSCLOCK, DISPLAYSURF, BASICFONT, BEEP1, BEEP2, BEEP3,
BEEP4
    # Change for requirement 3:
    global buttons
```

Ја иницијализираме листата со копчиња така што додаваме 4 копчиња. Со функцијата getButtonsSize ја пресметуваме големината на копчињата, а со функцијата createButton креираме речник кој ќе го означува копчето на соодветна позиција.

```
# Change for requirement 3:
buttons = []
button_size = getButtonsSize(2)
for i,j in [(0,0),(0,1),(1,0),(1,1)]:
    buttons.append(createButton(i, j, button_size))
```

Ја остранuваме функционалноста на избор на копче со клик на тастер од тастатурата затоа што бројот на копчиња динамички се менува е не е ограничен.

```
# Change for requirement 3:
# elif event.type == KEYDOWN:
#     if event.key == K_q:
#         clickedButton = YELLOW
#     elif event.key == K_w:
#         clickedButton = BLUE
#     elif event.key == K_a:
#         clickedButton = RED
#     elif event.key == K_s:
#         clickedButton = GREEN
```

Во main loop во делот кога не се чека на влез го менуваме кодот така што доколку score е делив со 10 ги ажурираме копчињата и додаваме нови. Со функцијата getButtonsSize ја пресметуваме големината на копчињата, со функцијата createButton креираме речник кој ќе го означува копчето на соодветна позиција, а со updateButtons ги ажурираме постоечките копчиња.

```
if not waitingForInput:
    # play the pattern
    pygame.display.update()
    pygame.time.wait(1000)
    # Change for requirement 2 & 3:
    if score % 10 == 0 and score != 0:
        num_buttons_row = int(score / 10) + 2
        button_size = getButtonsSize(num_buttons_row)
        updateButtons(buttons, button_size)
        for i in range(num_buttons_row):
            buttons.append(createButton(i, num_buttons_row - 1,
button_size))
        for i in range(num_buttons_row - 1):
            buttons.append(createButton(num_buttons_row - 1, i,
button_size))
        if TIMEOUT > 3:
            TIMEOUT -= 1
        DISPLAYSURF.fill(bgColor)
        drawButtons()
```

дополнително при додавање на копче во секвенца тоа го правиме преку избор на копче од листата со копчиња buttons наспроти досегашното со избор на боја од неколку предефинирани

```
    # Change for requirement 1 & 3:
    pattern_len = len(pattern)
    pattern = []
    for i in range(pattern_len):
        pattern.append(random.choice(buttons))
        # pattern.append(random.choice((YELLOW, BLUE, RED, GREEN)))
    pattern.append(random.choice(buttons))
    # pattern.append(random.choice((YELLOW, BLUE, RED, GREEN)))
```

Во main loop во делот кој што се извршува при gam over додаваме код со кој ја ресетираме buttons листата на листа од четири копчиња.

```
# pushed the incorrect button, or has timed out
gameOverAnimation()
# reset the variables for a new game:
# Change for requirement 3:
buttons = []
button_size = getButtonsSize(2)
for i, j in [(0, 0), (0, 1), (1, 0), (1, 1)]:
    buttons.append(createButton(i, j, button_size))
```

Дефинираме функција createButton која како аргументи прима позиција каде треба да се креира копче и големина на копчето, а како излез враќа реќник кој го претставува копчето. Речникот ги има следниве својства

- xpos и ypos: позиција на копчето
- color и lightColor: боја на копчето и незијна светла варијанта бојата се избира случајно при генерирање на копчето
- size: големина на копчето
- rect: објект кој ја представува поворшината на копчето на екранот
- beep: аудио кое се избира по случаен избор при генерирање на копчето

```
# Change for requirement 3:
def createButton(xpos, ypos, buttonSize):
    button = {}
    button["xpos"] = xpos
    button["ypos"] = ypos
    button['color'] = (random.randint(20, 170), random.randint(20,
170), random.randint(20, 170))
    button['lightColor'] = (button['color'][0] * 1.5,
button['color'][1] * 1.5, button['color'][2] * 1.5)
    button['size'] = buttonSize
    button['rect'] = pygame.Rect(XMARGIN + xpos * buttonSize + (xpos
- 1) * BUTTONGAPSIZE,
                                 YMARGIN + ypos * buttonSize + (ypos
- 1) * BUTTONGAPSIZE,
                                 buttonSize, buttonSize)
    button['beep'] = random.choice((BEEP1, BEEP2, BEEP3, BEEP4))
    return button
```

Дефинираме функција getButtonSize која на влез прима број на копчина во еден ред а н на излез враќа големината на копчињата. Големината на копчињата се пресметува така што ќе се исполни целиот прозорец а притоа ќе се задржат предефинираните вредности за x и y маргините и за просторот помеѓу копчињата

```
# Change for requirement 3:
def getButtonsSize(numButtonsInRow):
    return int((WINDOWWIDTH - (XMARGIN * 2) - (BUTTONGAPSIZE *
numButtonsInRow - 1)) / numButtonsInRow)
```

Дефинираме функција updateButtons која на влез прима листа со копчиња и новата големина. Се изминуваат сите копчиња во листата и се ажурираат size и rect својствата.

```
# Change for requirement 3:
def updateButtons(buttons, buttonSize):
    for button in buttons:
        button['size'] = buttonSize
        xpos = button["xpos"]
        ypos = button["ypos"]
        button['rect'] = pygame.Rect(
XMARGIN + xpos * buttonSize + (xpos - 1) * BUTTONGAPSIZE,
YMARGIN + ypos * buttonSize + (ypos - 1) * BUTTONGAPSIZE,
buttonSize, buttonSize)
```

Ја менуваме функцијата flashButtonAnimation така што наместо на влез да прима боја ќе прима копче (речник) и потребните променливи ќе ги зема од соодветните својства на речникот копче.

```
# Change for requirement 3:
def flashButtonAnimation(button, animationSpeed=50):
    # if color == YELLOW:
    #     sound = BEEP1
    #     flashColor = BRIGHTYELLOW
    #     rectangle = YELLOWRECT
    # elif color == BLUE:
    #     sound = BEEP2
    #     flashColor = BRIGHTBLUE
    #     rectangle = BLUERECT
    # elif color == RED:
```

```
    #     sound = BEEP3
    #     flashColor = BRIGHTRED
    #     rectangle = REDRECT
    # elif color == GREEN:
    #     sound = BEEP4
    #     flashColor = BRIGHTGREEN
    #     rectangle = GREENRECT

    sound = button['beep']
    flashColor = button['lightColor']
    rectangle = button['rect']
    buttonSize = button['size']
```

Ја менуваме функцијата drawButtons така што ќе се исцртуваат сите копчиња во листа со копчиња buttons

```
# Change for requirement 3:
def drawButtons():
    # pygame.draw.rect(DISPLAYSURF, YELLOW, YELLOWRECT)
    # pygame.draw.rect(DISPLAYSURF, BLUE, BLUERECT)
    # pygame.draw.rect(DISPLAYSURF, RED, REDRECT)
    # pygame.draw.rect(DISPLAYSURF, GREEN, GREENRECT)
    for button in buttons:
        pygame.draw.rect(DISPLAYSURF, button['color'],
button['rect'])
```

Ја менуваме функцијата getButtonClicked така што ќе се проверува за rect објектот од секое копче во листата buttons

```
# Change for requirement 3:
def getButtonClicked(x, y):
    for button in buttons:
        if button['rect'].collidepoint((x, y)):
            return button
    # if YELLOWRECT.collidepoint((x, y)):
    #     return YELLOW
    # elif BLUERECT.collidepoint((x, y)):
    #     return BLUE
    # elif REDRECT.collidepoint((x, y)):
    #     return RED
    # elif GREENRECT.collidepoint((x, y)):
    #     return GREEN
    return None
```

Слика од почетниот екран

Слика од екранот на играта при score=40