```python
# Simulate (a Simon clone)
# By Al Sweigart al@inventwithpython.com
# http://inventwithpython.com/pygame
# Released under a "Simplified BSD" license

import random, sys, time, pygame
from pygame.locals import *

FPS = 30
WINDOWWIDTH = 640
WINDOWHEIGHT = 480
FLASHSPEED = 500 # in milliseconds
FLASHDELAY = 200 # in milliseconds
BUTTONSIZE = 200
BUTTONGAPSIZE = 20
TIMEOUT = 4 # seconds before game over if no button is pushed.

#                R    G    B
WHITE        = (255, 255, 255)
BLACK        = (  0,   0,   0)
BRIGHTRED    = (255,   0,   0)
RED          = (155,   0,   0)
BRIGHTGREEN  = (  0, 255,   0)
GREEN        = (  0, 155,   0)
BRIGHTBLUE   = (  0,   0, 255)
BLUE         = (  0,   0, 155)
BRIGHTYELLOW = (255, 255,   0)
YELLOW       = (155, 155,   0)
DARKGRAY     = ( 40,  40,  40)
bgColor = BLACK

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

# Rect objects for each of the four buttons
YELLOWRECT = pygame.Rect(XMARGIN, YMARGIN, BUTTONSIZE, BUTTONSIZE)
BLUERECT   = pygame.Rect(XMARGIN + BUTTONSIZE + BUTTONGAPSIZE, YMARGIN, BUTTONSIZE, BUTTONSIZE)
REDRECT    = pygame.Rect(XMARGIN, YMARGIN + BUTTONSIZE + BUTTONGAPSIZE, BUTTONSIZE, BUTTONSIZE)
GREENRECT  = pygame.Rect(XMARGIN + BUTTONSIZE + BUTTONGAPSIZE, YMARGIN + BUTTONSIZE + BUTTONGAPSIZE, BUTTONSIZE, BUTTONSIZE

def main():
    global FPSCLOCK, DISPLAYSURF, BASICFONT, BEEP1, BEEP2, BEEP3, BEEP4

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

BASICFONT = pygame.font.Font('freesansbold.ttf', 16)
infoSurf = BASICFONT.render('Match the pattern by clicking on the button or using the Q, W, A, S keys.', 1, DARKGRAY)
infoRect = infoSurf.get_rect()
infoRect.topleft = (10, WINDOWHEIGHT - 25)

# 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')

# Initialize some variables for a new game
pattern = [] # stores the pattern of colors
currentStep = 0 # the color the player must push next
lastClickTime = 0 # timestamp of the player's last button push
score = 0
# when False, the pattern is playing. when True, waiting for the player to click a colored button:
waitingForInput = False

while True: # main game loop
    clickedButton = None # button that was clicked (set to YELLOW, RED, GREEN, or BLUE)
DISPLAYSURF.fill(bgColor)
drawButtons()

scoreSurf = BASICFONT.render('Score: ' + str(score), 1, WHITE)
scoreRect = scoreSurf.get_rect()
scoreRect.topleft = (WINDOWWIDTH - 100, 10)
DISPLAYSURF.blit(scoreSurf, scoreRect)

DISPLAYSURF.blit(infoSurf, infoRect)

checkForQuit()
for event in pygame.event.get(): # event handling loop
if event.type == MOUSEBUTTONUP:
    mousex, mousey = event.pos
    clickedButton = getButtonClicked(mousex, mousey)
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:
```

```python
                    clickedButton = GREEN



        if not waitingForInput:
            # play the pattern
            pygame.display.update()
            pygame.time.wait(1000)
            pattern.append(random.choice((YELLOW, BLUE, RED, GREEN)))
            for button in pattern:
                flashButtonAnimation(button)
                pygame.time.wait(FLASHDELAY)
            waitingForInput = True
        else:
            # wait for the player to enter buttons
            if clickedButton and clickedButton == pattern[currentStep]:
                # pushed the correct button
                flashButtonAnimation(clickedButton)
                currentStep += 1
                lastClickTime = time.time()

                if currentStep == len(pattern):
                    # pushed the last button in the pattern
                    changeBackgroundAnimation()
                    score += 1
                    waitingForInput = False
                    currentStep = 0 # reset back to first step

            elif (clickedButton and clickedButton != pattern[currentStep]) or (currentStep != 0 and time.time() - TIMEOUT >
                # pushed the incorrect button, or has timed out
                gameOverAnimation()
                # reset the variables for a new game:
                pattern = []
                currentStep = 0
                waitingForInput = False
                score = 0
                pygame.time.wait(1000)
                changeBackgroundAnimation()

        pygame.display.update()
        FPSCLOCK.tick(FPS)


def terminate():
    pygame.quit()
    sys.exit()
```

```python


def checkForQuit():
    for event in pygame.event.get(QUIT): # get all the QUIT events
        terminate() # terminate if any QUIT events are present
    for event in pygame.event.get(KEYUP): # get all the KEYUP events
        if event.key == K_ESCAPE:
            terminate() # terminate if the KEYUP event was for the Esc key
        pygame.event.post(event) # put the other KEYUP event objects back


def flashButtonAnimation(color, 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

    origSurf = DISPLAYSURF.copy()
    flashSurf = pygame.Surface((BUTTONSIZE, BUTTONSIZE))
    flashSurf = flashSurf.convert_alpha()
    r, g, b = flashColor
    sound.play()
    for start, end, step in ((0, 255, 1), (255, 0, -1)): # animation loop
        for alpha in range(start, end, animationSpeed * step):
            checkForQuit()
            DISPLAYSURF.blit(origSurf, (0, 0))
            flashSurf.fill((r, g, b, alpha))
            DISPLAYSURF.blit(flashSurf, rectangle.topleft)
            pygame.display.update()
            FPSCLOCK.tick(FPS)
    DISPLAYSURF.blit(origSurf, (0, 0))


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)


def changeBackgroundAnimation(animationSpeed=40):
    global bgColor
    newBgColor = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))

    newBgSurf = pygame.Surface((WINDOWWIDTH, WINDOWHEIGHT))
    newBgSurf = newBgSurf.convert_alpha()
    r, g, b = newBgColor
    for alpha in range(0, 255, animationSpeed): # animation loop
        checkForQuit()
        DISPLAYSURF.fill(bgColor)

        newBgSurf.fill((r, g, b, alpha))
        DISPLAYSURF.blit(newBgSurf, (0, 0))

        drawButtons() # redraw the buttons on top of the tint

        pygame.display.update()
        FPSCLOCK.tick(FPS)
    bgColor = newBgColor


def gameOverAnimation(color=WHITE, animationSpeed=50):
    # play all beeps at once, then flash the background
    origSurf = DISPLAYSURF.copy()
    flashSurf = pygame.Surface(DISPLAYSURF.get_size())
    flashSurf = flashSurf.convert_alpha()
    BEEP1.play() # play all four beeps at the same time, roughly.
    BEEP2.play()
    BEEP3.play()
    BEEP4.play()
    r, g, b = color
    for i in range(3): # do the flash 3 times
        for start, end, step in ((0, 255, 1), (255, 0, -1)):
            # The first iteration in this loop sets the following for loop
            # to go from 0 to 255, the second from 255 to 0.
            for alpha in range(start, end, animationSpeed * step): # animation loop
                # alpha means transparency. 255 is opaque, 0 is invisible
                checkForQuit()
                flashSurf.fill((r, g, b, alpha))
                DISPLAYSURF.blit(origSurf, (0, 0))
                DISPLAYSURF.blit(flashSurf, (0, 0))
                drawButtons()
                pygame.display.update()
                FPSCLOCK.tick(FPS)



def getButtonClicked(x, y):
    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


if __name__ == '__main__':
    main()
```