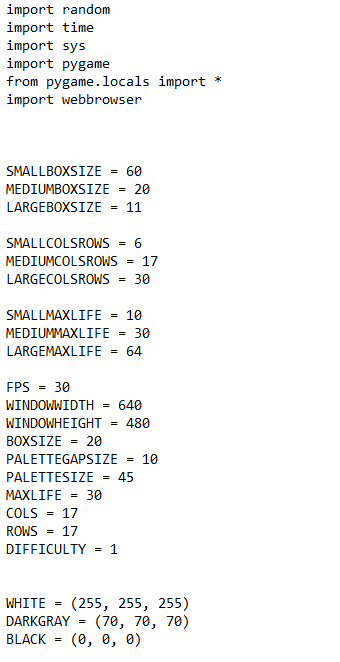
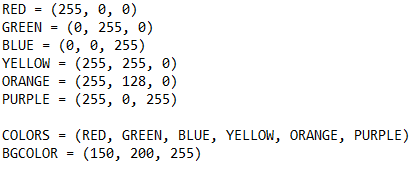
**INK SPILL GAME**





def main():

global MAINCLOCK, MAINSURF

pygame.init()

MAINCLOCK = pygame.time.Clock()

MAINSURF = pygame.display.set\_mode((WINDOWWIDTH, WINDOWHEIGHT))

loadImages()

pygame.display.set\_caption('Ink Spill')

mousex = 0

mousey = 0

mainBoard = generateRandomBoard(COLS, ROWS, DIFFICULTY)

life = MAXLIFE

lastPaletteClicked = None

# Main game loop:

while True:

paletteClicked = None

resetGame = False

# Draw the screen.

MAINSURF.fill(BGCOLOR)

drawLogo()

drawBoard(mainBoard)

drawLifeMeter(life)

drawPalettes()

# Handle any events.

for event in pygame.event.get():

if event.type == QUIT:

terminate()

if event.type == MOUSEMOTION:

mousex, mousey = event.pos

if event.type == MOUSEBUTTONUP:

mousex, mousey = event.pos

if pygame.Rect(WINDOWWIDTH - SETTINGSBUTTONIMAGE.get\_width(),

WINDOWHEIGHT - SETTINGSBUTTONIMAGE.get\_height(),

SETTINGSBUTTONIMAGE.get\_width(),

SETTINGSBUTTONIMAGE.get\_height()).collidepoint(mousex, mousey):

resetGame = showSettingsScreen()

elif pygame.Rect(WINDOWWIDTH - RESETBUTTONIMAGE.get\_width(),

WINDOWHEIGHT - SETTINGSBUTTONIMAGE.get\_height() - RESETBUTTONIMAGE.get\_height(),

RESETBUTTONIMAGE.get\_width(),

RESETBUTTONIMAGE.get\_height()).collidepoint(mousex, mousey):

resetGame = True

else:

paletteClicked = getClickedPalette(mousex, mousey)

if event.type == KEYUP:

if event.key == K\_ESCAPE:

terminate()

if paletteClicked != None and paletteClicked != lastPaletteClicked:

lastPaletteClicked = paletteClicked

floodAnimation(mainBoard, paletteClicked)

life -= 1

resetGame = False

if hasWon(mainBoard):

for i in range(4):

flashBorderAnimation(WHITE, mainBoard)

resetGame = True

time.sleep(2)

elif life == 0:

drawLifeMeter(0)

pygame.display.update()

time.sleep(0.4)

for i in range(4):

flashBorderAnimation(BLACK, mainBoard)

resetGame = True

time.sleep(2)

if resetGame:

mainBoard = generateRandomBoard(COLS, ROWS, DIFFICULTY)

life = MAXLIFE

lastPaletteClicked = None

pygame.display.update()

MAINCLOCK.tick(FPS)

def loadImages():

global LOGOIMAGE, NOLOGO, SPOTIMAGE, SETTINGSIMAGE, SETTINGSBUTTONIMAGE, RESETBUTTONIMAGE

NOLOGO = False

try:

LOGOIMAGE = pygame.image.load('inkspilllogo.png')

except pygame.error:

NOLOGO = True

SPOTIMAGE = pygame.image.load('inkspillspot.png')

SETTINGSIMAGE = pygame.image.load('inkspillsettings.png')

SETTINGSBUTTONIMAGE = pygame.image.load('inkspillsettingsbutton.png')

RESETBUTTONIMAGE = pygame.image.load('inkspillresetbutton.png')

def terminate():

pygame.quit()

sys.exit()

def hasWon(board):

color = board[0][0]

for x in range(COLS):

for y in range(ROWS):

if board[x][y] != color:

return False

return True

def showSettingsScreen():

global DIFFICULTY, BOXSIZE, COLS, ROWS, MAXLIFE, COLORS, BGCOLOR

origDifficulty = DIFFICULTY

origBoxSize = BOXSIZE

while True:

MAINSURF.fill(BGCOLOR)

MAINSURF.blit(SETTINGSIMAGE, (0,0))

if DIFFICULTY == 0:

MAINSURF.blit(SPOTIMAGE, (30, 4))

if DIFFICULTY == 1:

MAINSURF.blit(SPOTIMAGE, (8, 41))

if DIFFICULTY == 2:

MAINSURF.blit(SPOTIMAGE, (30, 76))

if BOXSIZE == SMALLBOXSIZE:

MAINSURF.blit(SPOTIMAGE, (22, 150))

if BOXSIZE == MEDIUMBOXSIZE:

MAINSURF.blit(SPOTIMAGE, (11, 185))

if BOXSIZE == LARGEBOXSIZE:

MAINSURF.blit(SPOTIMAGE, (24, 220))

for i in range(len(COLORSCHEMES)):

drawColorSchemeBoxes(500, i \* 60 + 30, i)

pygame.display.update()

for event in pygame.event.get():

if event.type == QUIT:

terminate()

if event.type == KEYUP:

if event.key == K\_ESCAPE:

return not (origDifficulty == DIFFICULTY and origBoxSize == BOXSIZE)

if event.type == MOUSEBUTTONUP:

mousex, mousey = event.pos

if pygame.Rect(74, 16, 111, 30).collidepoint(mousex, mousey):

DIFFICULTY = 0

if pygame.Rect(53, 50, 104, 29).collidepoint(mousex, mousey):

DIFFICULTY = 1

if pygame.Rect(72, 85, 65, 31).collidepoint(mousex, mousey):

DIFFICULTY = 2

if pygame.Rect(63, 156, 84, 31).collidepoint(mousex, mousey):

BOXSIZE = SMALLBOXSIZE

COLS = SMALLCOLSROWS

ROWS = SMALLCOLSROWS

MAXLIFE = SMALLMAXLIFE

if pygame.Rect(52, 192, 106,32).collidepoint(mousex, mousey):

BOXSIZE = MEDIUMBOXSIZE

COLS = MEDIUMCOLSROWS

ROWS = MEDIUMCOLSROWS

MAXLIFE = MEDIUMMAXLIFE

if pygame.Rect(67, 228, 58, 37).collidepoint(mousex, mousey):

BOXSIZE = LARGEBOXSIZE

COLS = LARGECOLSROWS

ROWS = LARGECOLSROWS

MAXLIFE = LARGEMAXLIFE

for i in range(len(COLORSCHEMES)):

if pygame.Rect(500, 30 + i \* 60, MEDIUMBOXSIZE \* 3, MEDIUMBOXSIZE \* 2).collidepoint(mousex, mousey):

COLORS = COLORSCHEMES[i][1:]

BGCOLOR = COLORSCHEMES[i][0]

if pygame.Rect(14, 299, 371, 97).collidepoint(mousex, mousey):

webbrowser.open('http://inventwithpython.com')

if pygame.Rect(178, 418, 215, 34).collidepoint(mousex, mousey):

return not (origDifficulty == DIFFICULTY and origBoxSize == BOXSIZE)

def drawColorSchemeBoxes(x, y, schemeNum):

for boxy in range(2):

for boxx in range(3):

pygame.draw.rect(MAINSURF, COLORSCHEMES[schemeNum][3 \* boxy + boxx + 1], (x + MEDIUMBOXSIZE \* boxx, y + MEDIUMBOXSIZE \* boxy, MEDIUMBOXSIZE, MEDIUMBOXSIZE))

if COLORS == COLORSCHEMES[schemeNum][1:]:

MAINSURF.blit(SPOTIMAGE, (x - 50, y))

def flashBorderAnimation(color, board, animationSpeed=30):

origSurf = MAINSURF.copy()

flashSurf = pygame.Surface(MAINSURF.get\_size())

flashSurf = flashSurf.convert\_alpha()

for start, end, step in ((0, 256, 1), (255, 0, -1)):

for transparency in range(start, end, animationSpeed \* step):

MAINSURF.blit(origSurf, (0, 0))

r, g, b = color

flashSurf.fill((r, g, b, transparency))

MAINSURF.blit(flashSurf, (0, 0))

drawBoard(board)

pygame.display.update()

MAINCLOCK.tick(FPS)

MAINSURF.blit(origSurf, (0, 0))

def getBoardCopy(board):

dupe = []

for x in range(COLS):

column = []

for y in range(ROWS):

column.append(board[x][y])

dupe.append(column)

return dupe

def floodAnimation(board, paletteClicked, animationSpeed=25):

origBoard = getBoardCopy(board)

flood(board, board[0][0], paletteClicked, 0, 0)

for transparency in range(0, 255, animationSpeed):

drawBoard(origBoard)

drawBoard(board, transparency)

pygame.display.update()

MAINCLOCK.tick(FPS)

def generateRandomBoard(width, height, difficulty=1):

board = []

for x in range(width):

column = []

for y in range(height):

column.append(random.randint(0, len(COLORS) - 1))

board.append(column)

# Make the board easier by setting some boxes to be the same color as their neighbor.

if difficulty == 0:

difficulty = 1500

elif difficulty == 1:

difficulty = 200

if difficulty == 0 and BOXSIZE == SMALLBOXSIZE:

difficulty = 100

elif difficulty == 1 and BOXSIZE == SMALLBOXSIZE:

difficulty = 5

elif BOXSIZE == LARGEBOXSIZE:

difficulty \*= 8

else:

difficulty = 0

for i in range(difficulty):

x = random.randint(1, width-2)

y = random.randint(1, height-2)

direction = random.randint(0, 3)

if direction == 0:

board[x-1][y] == board[x][y]

board[x][y-1] == board[x][y]

elif direction == 1:

board[x+1][y] == board[x][y]

board[x][y+1] == board[x][y]

elif direction == 2:

board[x][y-1] == board[x][y]

board[x+1][y] == board[x][y]

else:

board[x][y+1] == board[x][y]

board[x-1][y] == board[x][y]

return board

def drawLogo():

if not NOLOGO:

MAINSURF.blit(LOGOIMAGE, (WINDOWWIDTH - LOGOIMAGE.get\_width(), 0))

MAINSURF.blit(SETTINGSBUTTONIMAGE, (WINDOWWIDTH - SETTINGSBUTTONIMAGE.get\_width(), WINDOWHEIGHT - SETTINGSBUTTONIMAGE.get\_height()))

MAINSURF.blit(RESETBUTTONIMAGE, (WINDOWWIDTH - RESETBUTTONIMAGE.get\_width(), WINDOWHEIGHT - SETTINGSBUTTONIMAGE.get\_height() - RESETBUTTONIMAGE.get\_height()))

def drawBoard(board, transparency=255):

tempSurf = pygame.Surface(MAINSURF.get\_size())

tempSurf = tempSurf.convert\_alpha()

tempSurf.fill((0, 0, 0, 0))

for x in range(COLS):

for y in range(ROWS):

left, top = leftTopOfBox(x, y)

r, g, b = COLORS[board[x][y]]

pygame.draw.rect(tempSurf, (r, g, b, transparency), (left, top, BOXSIZE, BOXSIZE))

left, top = leftTopOfBox(0, 0)

pygame.draw.rect(tempSurf, BLACK, (left-1, top-1, BOXSIZE \* COLS + 1, BOXSIZE \* ROWS + 1), 1)

MAINSURF.blit(tempSurf, (0, 0))

def drawPalettes():

numColors = len(COLORS)

xmargin = int((WINDOWWIDTH - ((PALETTESIZE \* numColors) + (PALETTEGAPSIZE \* (numColors - 1)))) / 2)

for i in range(numColors):

left = xmargin + (i \* PALETTESIZE) + (i \* PALETTEGAPSIZE)

top = WINDOWHEIGHT - PALETTESIZE - 10

pygame.draw.rect(MAINSURF, COLORS[i], (left, top, PALETTESIZE, PALETTESIZE))

pygame.draw.rect(MAINSURF, BGCOLOR, (left + 2, top + 2, PALETTESIZE - 4, PALETTESIZE - 4), 2)

def drawLifeMeter(currentLife):

lifeBoxSize = int((WINDOWHEIGHT - 40) / MAXLIFE)

# Draw background of life box.

pygame.draw.rect(MAINSURF, BGCOLOR, (20, 20, 20, 20 + (MAXLIFE \* lifeBoxSize)))

for i in range(MAXLIFE):

if currentLife >= (MAXLIFE - i):

pygame.draw.rect(MAINSURF, RED, (20, 20 + (i \* lifeBoxSize), 20, lifeBoxSize))

pygame.draw.rect(MAINSURF, WHITE, (20, 20 + (i \* lifeBoxSize), 20, lifeBoxSize), 1)

def getClickedPalette(x, y):

numColors = len(COLORS)

xmargin = int((WINDOWWIDTH - ((PALETTESIZE \* numColors) + (PALETTEGAPSIZE \* (numColors - 1)))) / 2)

top = WINDOWHEIGHT - PALETTESIZE - 10

for i in range(numColors):

# Determine if the xy coordinates of the mouse click is inside any of the palettes.

left = xmargin + (i \* PALETTESIZE) + (i \* PALETTEGAPSIZE)

r = pygame.Rect(left, top, PALETTESIZE, PALETTESIZE)

if r.collidepoint(x, y):

return i

return None

def flood(board, oldColorNum, newColorNum, x, y):

if oldColorNum == newColorNum or board[x][y] != oldColorNum:

return

board[x][y] = newColorNum # change the color of the current box

# Make the recursive call for any neighboring boxes:

if x > 0:

flood(board, oldColorNum, newColorNum, x - 1, y)

if x < COLS - 1:

flood(board, oldColorNum, newColorNum, x + 1, y)

if y > 0:

flood(board, oldColorNum, newColorNum, x, y - 1)

if y < ROWS - 1:

flood(board, oldColorNum, newColorNum, x, y + 1)

def leftTopOfBox(boxx, boxy):

# Determine size of the margins for each side.

xmargin = int((WINDOWWIDTH - (COLS \* BOXSIZE)) / 2)

ymargin = int((WINDOWHEIGHT - (ROWS \* BOXSIZE)) / 2)

return (boxx \* BOXSIZE + xmargin, boxy \* BOXSIZE + ymargin)

if \_\_name\_\_ == '\_\_main\_\_':

main()

**OUTPUT:**

