mport random, time, pygame, sys  
from pygame.locals import \*  
  
FPS = 25  
WINDOWWIDTH = 1280  
WINDOWHEIGHT = 720  
BOXSIZE = 30  
BOARDWIDTH = 10  
BOARDHEIGHT = 20  
BLANK = '-'  
MOVESIDEWAYSFREQ = 0.15  
XMARGIN = int((WINDOWWIDTH - BOARDWIDTH \* BOXSIZE) / 2)  
TOPMARGIN = WINDOWHEIGHT - (BOARDHEIGHT \* BOXSIZE) - 50  
  
WHITE = (255, 255, 255)  
GRAY = (105, 105, 105)  
BLACK = ( 0, 0, 0)  
RED = (255, 0, 0)  
GREEN = ( 0, 255, 0)  
BLUE = ( 0, 191, 255)  
YELLOW = (255, 255, 0)  
PINK = (255, 0, 255)  
TEAL = (184, 134, 11)  
VIOLET = (138, 43, 226)  
NAVY = ( 25, 25, 72)  
  
BORDERCOLOR = VIOLET  
BGCOLOR = BLACK  
TEXTCOLOR = WHITE  
COLORS = (RED, PINK, YELLOW, BLUE, GREEN, TEAL)  
  
TEMPLATEWIDTH = 5  
TEMPLATEHEIGHT = 5  
O\_TEMPLATE = [['-----',  
 '-----',  
 '-OO--',  
 '-OO--',  
 '-----']]  
  
PIECES = {'O': O\_TEMPLATE}  
  
def main():  
 global FPSCLOCK, DISPLAYSURF, BASICFONT, BIGFONT  
 pygame.init()  
 FPSCLOCK = pygame.time.Clock()  
 DISPLAYSURF = pygame.display.set\_mode((WINDOWWIDTH, WINDOWHEIGHT))  
 BASICFONT = pygame.font.Font('calibrib.ttf', 20)  
 BIGFONT = pygame.font.Font('calibrib.ttf', 60)  
 pygame.display.set\_caption("POLLY'S FIRST GAME")  
 showStart()  
 while True:  
 runGame()  
 showOps()  
  
def runGame():  
 # Настройка переменных для начала игры (time.time() - время, выраженное в секундах с начала эпохи)  
 board = getBlankBoard()  
 lastMoveDownTime = time.time()  
 lastMoveSidewaysTime = time.time()  
 lastFallTime = time.time()  
 movingLeft = False  
 movingRight = False  
 score = 0  
 level, fallFreq = calculateLevelAndFallFreq(score)  
 fallingPiece = getNewPiece()  
 nextPiece = getNewPiece()  
  
 while True:  
 if fallingPiece == None:  
 # Нет падающей фигуры в игре, поэтому начинает падать новая сверху  
 fallingPiece = nextPiece  
 nextPiece = getNewPiece()  
 lastFallTime = time.time() # сброс последнего времени  
 # не может поместиться новая фигура на доске, так что игра окончена  
 if not isValidPosition(board, fallingPiece):  
 return  
 checkForQuit()  
 for event in pygame.event.get(): # цикл обработки событий  
 if event.type == KEYUP:  
 if (event.key == K\_LEFT):  
 movingLeft = False  
 elif (event.key == K\_RIGHT):  
 movingRight = False  
 elif event.type == KEYDOWN:  
 # перемещение фигуры влево  
 if (event.key == K\_LEFT) and isValidPosition(board, fallingPiece, adjX=-1):  
 fallingPiece['x'] -= 1  
 movingLeft = True  
 movingRight = False  
 lastMoveSidewaysTime = time.time()  
 # перемещение фигуры вправо  
 elif (event.key == K\_RIGHT) and isValidPosition(board, fallingPiece, adjX=1):  
 fallingPiece['x'] += 1  
 movingRight = True  
 movingLeft = False  
 lastMoveSidewaysTime = time.time()  
 # переместить фигуру полностью вниз  
 elif event.key == K\_SPACE:  
 movingDown = False  
 movingLeft = False  
 movingRight = False  
 for i in range(1, BOARDHEIGHT):  
 if not isValidPosition(board, fallingPiece, adjY=i):  
 break  
 fallingPiece['y'] += i - 1  
  
 # выполнять перемещение фигуры по пользовательскому вводу  
 if (movingLeft or movingRight) and time.time() - lastMoveSidewaysTime > MOVESIDEWAYSFREQ:  
 if movingLeft and isValidPosition(board, fallingPiece, adjX=-1):  
 fallingPiece['x'] -= 1  
 elif movingRight and isValidPosition(board, fallingPiece, adjX=1):  
 fallingPiece['x'] += 1  
 lastMoveSidewaysTime = time.time()  
  
 if time.time() - lastFallTime > fallFreq:  
 if not isValidPosition(board, fallingPiece, adjY=1):  
 addToBoard(board, fallingPiece)  
 score += removeCompleteLines(board)  
 level, fallFreq = calculateLevelAndFallFreq(score)  
 fallingPiece = None  
 else:  
 fallingPiece['y'] += 1  
 lastFallTime = time.time()  
  
 # рисование всего на экране  
 DISPLAYSURF.fill(BGCOLOR)  
 drawBoard(board)  
 drawStatus(score, level)  
 drawNextPiece(nextPiece)  
 if fallingPiece != None:  
 drawPiece(fallingPiece)  
 pygame.display.update()  
 FPSCLOCK.tick(FPS)  
  
# вывод надписи  
def makeTextObjs(text, font, color):  
 surf = font.render(text, True, color)  
 return surf, surf.get\_rect()  
  
# выход из игры  
def terminate():  
 pygame.quit()  
 sys.exit()  
  
# проверка нажима на кнопки  
def checkForKeyPress():  
 checkForQuit()  
 for event in pygame.event.get([KEYDOWN, KEYUP]):  
 if event.type == KEYDOWN:  
 continue  
 return event.key  
 return None  
  
# ESC - выход из игры  
def checkForQuit():  
 for event in pygame.event.get(KEYUP):  
 if event.key == K\_ESCAPE:  
 terminate()  
 pygame.event.post(event)  
  
# Основываясь на счете, возвращает уровень, на котором находится игрок, и  
# сколько секунд проходит, пока падающая фигура не упадет на один пробел.  
def calculateLevelAndFallFreq(score):  
 level = int(score / 10) + 1  
 fallFreq = 0.25 - (level \* 0.02)  
 return level, fallFreq  
  
# случайнная генерация новой фигуры  
def getNewPiece():  
 shape = random.choice(list(PIECES.keys()))  
 newPiece = {'shape': shape,  
 'rotation': random.randint(0, len(PIECES[shape]) - 1),  
 'x': int(BOARDWIDTH / 2) - int(TEMPLATEWIDTH / 2),  
 'y': -2, # начинает его над доской (то есть меньше 0)  
 'color': random.randint(0, len(COLORS)-1)}  
 return newPiece  
  
# заполняет игровое поле в зависимости от расположения, формы и вращения фигуры.  
def addToBoard(board, piece):  
 for x in range(TEMPLATEWIDTH):  
 for y in range(TEMPLATEHEIGHT):  
 if PIECES[piece['shape']][piece['rotation']][y][x] != BLANK:  
 board[x + piece['x']][y + piece['y']] = piece['color']  
  
# создание и возврат чистого игрового поля  
def getBlankBoard():  
 board = []  
 for i in range(BOARDWIDTH):  
 board.append([BLANK] \* BOARDHEIGHT)  
 return board  
  
# координаты x и y внутри игрового поля  
def isOnBoard(x, y):  
 return x >= 0 and x < BOARDWIDTH and y < BOARDHEIGHT  
  
# возвращает True, если фигура находится внутри доски и не сталкивается  
def isValidPosition(board, piece, adjX=0, adjY=0):  
 for x in range(TEMPLATEWIDTH):  
 for y in range(TEMPLATEHEIGHT):  
 isAboveBoard = y + piece['y'] + adjY < 0  
 if isAboveBoard or PIECES[piece['shape']][piece['rotation']][y][x] == BLANK:  
 continue  
 if not isOnBoard(x + piece['x'] + adjX, y + piece['y'] + adjY):  
 return False  
 if board[x + piece['x'] + adjX][y + piece['y'] + adjY] != BLANK:  
 return False  
 return True  
  
# возвращает True, если строка заполнена полями без пробелов.  
def isCompleteLine(board, y):  
 for x in range(BOARDWIDTH):  
 if board[x][y] == BLANK:  
 return False  
 return True  
  
# Удаляет все завершенные линии на доске, перемещает все выше них вниз  
# и возвращмет количество удаленных линий.  
def removeCompleteLines(board):  
 numLinesRemoved = 0  
 y = BOARDHEIGHT - 1 # start y at the bottom of the board  
 while y >= 0:  
 if isCompleteLine(board, y):  
 # Удаляет линию и перемещает коробки вниз на одну линию.  
 for pullDownY in range(y, 0, -1):  
 for x in range(BOARDWIDTH):  
 board[x][pullDownY] = board[x][pullDownY-1]  
 # делаем самую верхнюю строку пустой  
 for x in range(BOARDWIDTH):  
 board[x][0] = BLANK  
 numLinesRemoved += 1  
 else:  
 y -= 1 # перейти к проверке следующего ряда вверх  
 return numLinesRemoved  
  
def convertToPixelCoords(boxx, boxy):  
# Преобразование заданных координат XY доски   
# в xy координаты местоположения на экране.  
 return (XMARGIN + (boxx \* BOXSIZE)), (TOPMARGIN + (boxy \* BOXSIZE))  
  
def drawBox(boxx, boxy, color, pixelx=None, pixely=None):  
# Рисует один блок (каждая фигура состоит из 4 блоков)  
# в координатах xy на доске. Или, если заданы pixelx & pixely  
# то, рисуются в пиксельных координатах, хранящихся в  
# pixelx & pixely (это используется для "следующей" фигуры).  
 if color == BLANK:  
 return  
 if pixelx == None and pixely == None:  
 pixelx, pixely = convertToPixelCoords(boxx, boxy)  
 pygame.draw.rect(DISPLAYSURF, COLORS[color], (pixelx + 3, pixely + 3, BOXSIZE - 3, BOXSIZE - 3))  
  
def drawBoard(board):  
# фон окна заменяется изображением  
 fonimage = pygame.image.load("fon01.jpg").convert()  
 DISPLAYSURF.blit(fonimage, (0, 0))  
# рисуется граница вокруг игрового поля  
 pygame.draw.rect(DISPLAYSURF, BORDERCOLOR, (XMARGIN - 3, TOPMARGIN - 3, (BOARDWIDTH \* BOXSIZE) + 8, (BOARDHEIGHT \* BOXSIZE) + 8), 5)  
# закрашивается фон игрового поля  
 pygame.draw.rect(DISPLAYSURF, BGCOLOR, (XMARGIN, TOPMARGIN, BOXSIZE \* BOARDWIDTH, BOXSIZE \* BOARDHEIGHT))  
# рисуются отдельные квадраты на доске  
 for x in range(BOARDWIDTH):  
 for y in range(BOARDHEIGHT):  
 drawBox(x, y, board[x][y])  
  
def drawPiece(piece, pixelx=None, pixely=None):  
 shapeToDraw = PIECES[piece['shape']][piece['rotation']]  
 if pixelx == None and pixely == None:  
# если pixelx и pixely не были указаны, то рисуется на игровом поле  
 pixelx, pixely = convertToPixelCoords(piece['x'], piece['y'])  
# рисует квадраты из которых состоят фигуры  
 for x in range(TEMPLATEWIDTH):  
 for y in range(TEMPLATEHEIGHT):  
 if shapeToDraw[y][x] != BLANK:  
 drawBox(None, None, piece['color'], pixelx + (x \* BOXSIZE), pixely + (y \* BOXSIZE))  
  
def drawStatus(score, level):  
# Рисует текст "счёт" и его значение  
 scoreSurf = BASICFONT.render('Счёт: %s' % score, True, TEXTCOLOR)  
 scoreRect = scoreSurf.get\_rect()  
 scoreRect.topleft = (WINDOWWIDTH - 250, 70)  
 DISPLAYSURF.blit(scoreSurf, scoreRect)  
# Рисует текст "уровень" и его значение  
 levelSurf = BASICFONT.render('Уровень: %s' % level, True, TEXTCOLOR)  
 levelRect = levelSurf.get\_rect()  
 levelRect.topleft = (WINDOWWIDTH - 250, 100)  
 DISPLAYSURF.blit(levelSurf, levelRect)  
  
def drawNextPiece(piece):  
# Рисует текст "следующий"   
 nextSurf = BASICFONT.render('Следующий:', True, TEXTCOLOR)  
 nextRect = nextSurf.get\_rect()  
 nextRect.topleft = (WINDOWWIDTH - 250, 130)  
 DISPLAYSURF.blit(nextSurf, nextRect)  
# рисует "следующую" фигуру  
 drawPiece(piece, pixelx=WINDOWWIDTH - 260, pixely = 110)  
  
# отображает картинку в центре экрана, пока не будет нажата клавиша  
def showOps():  
 myimage = pygame.image.load("ops.png").convert\_alpha()  
 myimage = pygame.transform.scale(myimage, (500, 500))  
 DISPLAYSURF.blit(myimage, (int(WINDOWWIDTH / 2) - 250, int(WINDOWHEIGHT / 2) - 250))  
# Рисуется надпись начала игры  
 pressKeySurf, pressKeyRect = makeTextObjs('Нажать любую клавишу, чтобы начать снова', BASICFONT, TEXTCOLOR)  
 pressKeyRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2) + 240)  
 DISPLAYSURF.blit(pressKeySurf, pressKeyRect)  
 while checkForKeyPress() == None:  
 pygame.display.update()  
 FPSCLOCK.tick()  
  
# отображает картинку в центре экрана, пока не будет нажата клавиша  
def showStart():  
 myimage = pygame.image.load("start2.png").convert\_alpha()  
 myimage = pygame.transform.scale(myimage, (500, 500))  
 DISPLAYSURF.blit(myimage, (int(WINDOWWIDTH / 2) - 250, int(WINDOWHEIGHT / 2) - 250))  
 # Рисуется надпись  
 pressKeySurf, pressKeyRect = makeTextObjs('Нажать любую клавишу, чтобы начать', BASICFONT, TEXTCOLOR)  
 pressKeyRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2) + 240)  
 DISPLAYSURF.blit(pressKeySurf, pressKeyRect)  
  
 while checkForKeyPress() == None:  
 pygame.display.update()  
 FPSCLOCK.tick()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()