오픈소스SW 과제중심수업 보고서

ICT 융합학부 미디어테크놀로지 전공
2017004802 이영은

Github repository 주소 : https://github.com/two-zerosilver/osw

1. 테트리스 소스코드 내부 함수 설명

main()

```
def main():
    global FPSCLOCK, DISPLAYSURF, BASICFONT, BIGFONT
    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    BASICFONT = pygame.font.Font('freesansbold.ttf', 18)
    BIGFONT = pygame.font.Font('freesansbold.ttf', 100)
    pygame.display.set_caption('2017004802 LEEYOUNGEUN')

    showTextScreen('MY TETRIS')
```

메인 함수에서는 초기화부터 게임 창 생성, 텍스트의 폰트 등 기본적인 설정을 해준다. 주어진 조건에 맞게, 게임 창 부분을 수정해주었다.

```
    while True: # game loop
        if random.randint(0, 2) == 0:
            pygame.mixer.music.load('Hover.mp3')
        elif random.randint(0,2) == 1:
            pygame.mixer.music.load('Our_Lives_Past.mp3')
        else:
            pygame.mixer.music.load('Platform_9.mp3')
        pygame.mixer.music.play(-1, 0.0)
        runGame()
        pygame.mixer.music.stop()
        showTextScreen('OVER T^T')
```

main() 함수는 무작위로 노래를 틀어주고 게임이 끝나면 게임 오버 화면이 뜨게 된다. 주어진 조건에 맞게 3개의 음악이 랜덤하게 플레이 되도록 설정했고, 텍스트 화면에 들어가는 부분을 수정해주었다.

새 게임 시작 runGame()

```
def runGame():
    # setup variables for the start of the game
    board = getBlankBoard()
    lastMoveDownTime = time.time()
    lastMoveSidewaysTime = time.time()
    lastFallTime = time.time()
    movingDown = False # note: there is no movingUp variable
    movingLeft = False
    movingRight = False
    playtime = time.time()
    score = 0
    level, fallFreq = calculateLevelAndFallFreq(score)

    fallingPiece = getNewPiece()
    nextPiece = getNewPiece()
```

실제 게임을 수행하는 코드들은 runGame()에 있다. 게임 시작 전에 모두 초기화한다. fallingPiece는 현재 떨어지고 있는 플레이어가 회전시킬 수 있는 피스로 설정하며 다음에 떨어질 피스는 nextPiece로 설정된다.

게임 루프

```
while True: # game loop
    if fallingPiece == None:
        # No falling piece in play, so start a new piece at the top
        fallingPiece = nextPiece
        nextPiece = getNewPiece()
        lastFallTime = time.time() # reset lastFallTime

        if not isValidPosition(board, fallingPiece):
            return # can't fit a new piece on the board, so game over

    checkForQuit()
```

떨어지고 있는 피스가 없으면 새 피스가 위에서 떨어지도록 한다.

이벤트 처리 루프

```
for event in pygame.event.get(): # event handling loop
    if event.type == KEYUP:
```

이벤트 처리 루프는 플레이어가 떨어지는 피스를 회전시키거나 피스를 이동하거나 잠시 게임을 멈출 때 발생하는 이벤트를 처리한다.

게임 잠시 멈추기

```
if event.type == KEYUP:
    if (event.key == K_p):
        # Pausing the game
        DISPLAYSURF.fill(BGCOLOR)
        pygame.mixer.music.stop()
        showTextScreen('Get a rest!') # pause until a key press
        pygame.mixer.music.play(-1, 0.0)
        lastFallTime = time.time()
        lastMoveDownTime = time.time()
        lastMoveSidewaysTime = time.time()
```

P를 누르면 게임을 잠깐 멈출 수 있다. 이때, DISPLAYSURF.fil을 통해 화면을 가리고 음악을 멈춘다. 이는 플레이어가 게임을 중단하고 블록 위치를 생각하는 경우를 예방하기 위함이다. 플레이어가 아무 키를 누르면 showTextScreen("Get a rest!")은 반환되어 게임을 재개한다.

사용자 입력을 처리하기 위해 움직임 관련 변수 사용하기

```
elif (event.key == K_LEFT or event.key == K_a):
    movingLeft = False
elif (event.key == K_RIGHT or event.key == K_d):
    movingRight = False
elif (event.key == K_DOWN or event.key == K_s):
    movingDown = False
```

키에서 손을 땠을 때 False 값으로 설정한다. 이는 플레이어가 더이상 그 방향으로 움직이려고 하지 않음을 의미한다.

밀거나 회전했을 때 유효한지 확인하기

```
elif event.type == KEYDOWN:
    # moving the piece sideways
    if (event.key == K_LEFT or event.key == K_a) and isValidPosition(board, fallingPiece, adjX=-1):
        fallingPiece['x'] -= 1
        movingLeft = True
        movingRight = False
        lastMoveSidewaysTime = time.time()

    elif (event.key == K_RIGHT or event.key == K_d) and isValidPosition(board, fallingPiece, adjX=1):
        fallingPiece['x'] += 1
        movingRight = True
        movingLeft = False
        lastMoveSidewaysTime = time.time()
```

adjx에 -1값을 주면 왼쪽으로 한 칸 이동한 위치에서 피스의 위치가 적절한지를 검사한다. +1이면 오른쪽 y는 위아래에 대해 검사한다.

블록 회전

```
# rotating the piece (if there is room to rotate)
# handle moving the piece because of user input
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 movingDown and time.time() - lastMoveDownTime > MOVEDOWNFREQ and isValidPosition(board, fallingPiece, adjY=1):
    fallingPiece['y'] += 1
    lastMoveDownTime = time.time()
```

```
elif (event.key == K_q): # rotate the other direction
    fallingPiece['rotation'] = (fallingPiece['rotation'] - 1) % len(PIECES[fallingPiece['shape']])
    if not isValidPosition(board, fallingPiece):
        fallingPiece['rotation'] = (fallingPiece['rotation'] + 1) % len(PIECES[fallingPiece['shape']])
```

```
# handle moving the piece because of user input
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 movingDown and time.time() - lastMoveDownTime > MOVEDOWNFREQ and isValidPosition(board, fallingPiece, adjY=1):
    fallingPiece['y'] += 1
    lastMoveDownTime = time.time()
```

아래쪽 화살표 키나 s를 누르면 블록이 더 빠른 속도로 내려간다.

바닥 찾아내기

```
# move the current piece all the way down
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
```

플레이어가 스페이스 바를 누르면 현재 테트리스 블록이 바로 바닥로 떨어진다. 이를 위해 프로그램은 아래로 착지하기 위해 얼마나 많은 공간이 남았는지 확인해야 한다. isValidPosition이 False를 반환하면 더 이상 아래로 내려갈 수 없고, True를 반환하면 블록이 한 칸 아래로 이동할 수 있음을 의미한다.

키를 누르고 있는 동안 움직이기

```
# handle moving the piece because of user input
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 movingDown and time.time() - lastMoveDownTime > MOVEDOWNFREQ and isValidPosition(board, fallingPiece, adjY=1):
    fallingPiece['y'] += 1
    lastMoveDownTime = time.time()
```

테트리스 블록이 자연스럽게 떨어지도록 만들기

```
# let the piece fall if it is time to fall
if time.time() - lastFallTime > fallFreq:
    # see if the piece has landed
    if not isValidPosition(board, fallingPiece, adjY=1):
        # falling piece has landed, set it on the board
        addToBoard(board, fallingPiece)
        score += removeCompleteLines(board)
        level, fallFreq = calculateLevelAndFallFreq(score)
        fallingPiece = None
    else:
        # piece did not land, just move the piece down
        fallingPiece['y'] += 1
        lastFallTime = time.time()
```

떨어질 시간이 되면 블록을 떨어뜨린다.

스크린에 모두 그리기

```
# drawing everything on the screen
DISPLAYSURF.fill(BGCOLOR)
drawBoard(board)
drawStatus(playtime, score, level)
drawNextPiece(nextPiece)
if fallingPiece != None:
    drawPiece(fallingPiece)

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

스크린에 모두 그린다.

텍스트를 만드는 단축 함수 makeTextObjs()

```
def makeTextObjs(text, font, color):
    surf = font.render(text, True, color)
    return surf, surf.get_rect()
```

text 객체를 만드는 함수이다. 파라미터로 객체를 받아 render method를 호출한 후 surf와 rect 객체를 반환한다. 스크린에 폰트와 색을 반영한 텍스트를 보여주기 위해 텍스트에 주어진 폰트와 컬러를 반영한 값을 반환한다.

terminate()

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

게임을 나가는 함수이다.

checkForKeyPress()

```
def checkForKeyPress():
    # Go through event queue looking for a KEYUP event.
    # Grab KEYDOWN events to remove them from the event queue.
    checkForQuit()

    for event in pygame.event.get([KEYDOWN, KEYUP]):
        if event.type == KEYDOWN:
            continue
        return event.key
    return None
```

키가 눌렸는지 확인하는 함수로 KEYUP 이벤트가 발생했는지 이벤트 큐를 찾는다. KEYDOWN 이벤트는 찾아서 이벤트 큐에서 제거한다. 만약 KEYUP 이벤트가 없으면 함수는 none을 반환한다.

텍스트 스크린 함수 showTextScreen()

```
def showTextScreen(text):
    # This function displays large text in the
    # center of the screen until a key is pressed.
    # Draw the text drop shadow
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, TEXTSHADOWCOLOR)
    titleRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2))
    DISPLAYSURF.blit(titleSurf, titleRect)

    # Draw the text
    titleSurf, titleRect = makeTextObjs(text, BIGFONT, TEXTCOLOR)
    titleRect.center = (int(WINDOWWIDTH / 2) - 3, int(WINDOWHEIGHT / 2) - 3)
    DISPLAYSURF.blit(titleSurf, titleRect)

    # Draw the additional "Press a key to play." text.
    pressKeySurf, pressKeyRect = makeTextObjs('Press any key to play! Pause key is p.', BASICFONT, TEXTCOLOR)
    pressKeyRect.center = (int(WINDOWWIDTH / 2), int(WINDOWHEIGHT / 2) + 100)
    DISPLAYSURF.blit(pressKeySurf, pressKeyRect)

    while checkForKeyPress() == None:
        pygame.display.update()
        FPSCLOCK.tick()
```

플레이어가 키를 누를 때까지 스크린 중간에 커다란 텍스트를 보여주는 함수이다. Surf, rect를 makeTextObjs() 함수로부터 받아서 그려주는 함수이다. 플레이어가 키를 누를 때까지 화면을 업데이트한다.

checkForQuit()

```
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
```

terminate를 호출해 게임을 종료시키는 함수이다.

calculateLevelAndFallFreq()

```
def calculateLevelAndFallFreq(score):
    # Based on the score, return the level the player is on and
    # how many seconds pass until a falling piece falls one space.
    level = int(score / 10) + 1
    fallFreq = 0.27 - (level * 0.02)
    return level, fallFreq
```

플레이어의 점수에 따라 레벨을 올리고 떨어지는 속도를 조정해주는 함수로 중요한 함수이다. Int는 반내림하기 때문에 int(score / 10) + 1을 해줘야 한다. 그렇지 않으면, 첫 레벨이 0이 된다. 레벨에 따라 속도를 정해 두 값을 반환하는 함수이다.

getNewPiece()

```
def getNewPiece():
    # return a random new piece in a random rotation and color
    shape = random.choice(list(PIECES.keys()))

    if shape == 'S':
        color = 0
    elif shape == 'Z':
        color = 1
    elif shape == 'J':
        color = 2
    elif shape == 'L':
        color = 3
    elif shape == 'I':
        color = 4
    elif shape == 'O':
        color = 5
    elif shape == 'T':
        color = 6

    newPiece = {'shape': shape,
                'rotation': random.randint(0, len(PIECES[shape]) - 1),
                'x': int(BOARDWIDTH / 2) - int(TEMPLATEWIDTH / 2),
                'y': -2, # start it above the board (i.e. less than 0)
                'color': color}
    return newPiece
```

테트리스의 피스를 만드는 함수이다. 무작위로 모양, 방향, 색을 정하고 그 값들을 딕셔너리에 반환한다. 원래 코드라면 같은 블록이라도 랜덤하게 색이 배정되었지만 주어진 조건에 맞추어, 블럭 모양마다 색을 지정해주었다.

addToBoard()

```
def addToBoard(board, piece):
    # fill in the board based on piece's location, shape, and rotation
    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']
```

이전 블록이 착지한 보드 공간을 계속 기록해 나가는 함수이다. 블록의 데이터 구조를 받아서 board 데이터 구조에 상자들을 추가한다. 블록이 바닥이나 다른 블록 위에 완전히 착지하고 나면 데이터 구조를 갱신한다.

getBlankBoard()

```
def getBlankBoard():
    # create and return a new blank board data structure
    board = []
    for i in range(BOARDWIDTH):
        board.append([BLANK] * BOARDHEIGHT)
    return board
```

비어있는 새 보드 데이터 구조를 생성하고 반환하는 함수이다.

isOnBoard()

```
def isOnBoard(x, y):
    return x >= 0 and x < BOARDWIDTH and y < BOARDHEIGHT
```

주어진 x, y 좌표값이 보드의 안인지 밖인지 판단하는 함수이다.

isValidPosition()

```
def isValidPosition(board, piece, adjX=0, adjY=0):
    # Return True if the piece is within the board and not colliding
    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
```

블록이 보드 안에 있고, 충돌하는지 판단하기 위한 함수이다. 모든 상자들이 보드에 위치하고 보드의 다른 상자와 겹치지 않으면 True 를 반환한다.

isCompleteLine()

```
def isCompleteLine(board, y):
    # Return True if the line filled with boxes with no gaps.
    for x in range(BOARDWIDTH):
        if board[x][y] == BLANK:
            return False
    return True
```

테트리스에서 한 줄이 완성되었는지 판단하는 함수이다. 완성되면 True 를 반환한다.

removeCompleteLines()

```
def removeCompleteLines(board):
    # Remove any completed lines on the board, move everything above them down, and return the number of complete lines
    numLinesRemoved = 0
    y = BOARDHEIGHT - 1 # start y at the bottom of the board
    while y >= 0:
        if isCompleteLine(board, y):
            # Remove the line and pull boxes down by one line.
            for pullDownY in range(y, 0, -1):
                for x in range(BOARDWIDTH):
                    board[x][pullDownY] = board[x][pullDownY-1]
            # Set very top line to blank.
            for x in range(BOARDWIDTH):
                board[x][0] = BLANK
            numLinesRemoved += 1
            # Note on the next iteration of the loop, y is the same.
            # This is so that if the line that was pulled down is also
            # complete, it will be removed.
        else:
            y -= 1 # move on to check next row up
    return numLinesRemoved
```

isCompleteLine() 함수를 통해 완성된 줄을 발견하면, 그 줄을 지우고 그 줄 위에 있던 상자들을 다시 아래로 한칸으로 내리는 함수이다. 지워진 줄의 수를 반환한다.

convertToPixelCoords()

```
def convertToPixelCoords(boxx, boxy):
    # Convert the given xy coordinates of the board to xy
    # coordinates of the location on the screen.
    return (XMARGIN + (boxx * BOXSIZE)), (TOPMARGIN + (boxy * BOXSIZE))
```

주어진 보드의 xy 좌표를 픽셀 좌표계로 변환하는 함수이다.

drawBox()

```
def drawBox(boxx, boxy, color, pixelx=None, pixely=None):
    # draw a single box (each tetromino piece has four boxes)
    # at xy coordinates on the board. Or, if pixelx & pixely
    # are specified, draw to the pixel coordinates stored in
    # pixelx & pixely (this is used for the "Next" piece).
    if color == BLANK:
        return
    if pixelx == None and pixely == None:
        pixelx, pixely = convertToPixelCoords(boxx, boxy)
    pygame.draw.rect(DISPLAYSURF, COLORS[color], (pixelx + 1, pixely + 1, BOXSIZE - 1, BOXSIZE - 1))
    pygame.draw.rect(DISPLAYSURF, LIGHTCOLORS[color], (pixelx + 1, pixely + 1, BOXSIZE - 4, BOXSIZE - 4))
```

주어진 위치를 convertToPixelCoords() 함수를 호출하여 xy 좌표를 얻은 후, 그 위치에 상자를 그리는 함수이다.

drawboard()

```
def drawBoard(board):
    # draw the border around the board
    pygame.draw.rect(DISPLAYSURF, BORDERCOLOR, (XMARGIN - 3, TOPMARGIN - 7, (BOARDWIDTH * BOXSIZE) + 8, (BOARDHEIGHT *
    # fill the background of the board
    pygame.draw.rect(DISPLAYSURF, BGCOLOR, (XMARGIN, TOPMARGIN, BOXSIZE * BOARDWIDTH, BOXSIZE * BOARDHEIGHT))
    # draw the individual boxes on the board
    for x in range(BOARDWIDTH):
        for y in range(BOARDHEIGHT):
            drawBox(x, y, board[x][y])
```

파란색 테두리를 그리고, 보드의 배경색을 채운 뒤 바닥에 쌓인 블록을 이루는 상자들을 그리는 함수이다.

drawStatus()

```
def drawStatus(playtime, score, level):
     # draw the playtime text
    timeSurf = BASICFONT.render('Play time: %d sec' % (time.time() - playtime), True, TEXTCOLOR)
    timeRect = timeSurf.get_rect()
    timeRect.topright = (WINDOWWIDTH - 450, 20)
    DISPLAYSURF.blit(timeSurf, timeRect)

    # draw the score text
    scoreSurf = BASICFONT.render('Score: %s' % score, True, TEXTCOLOR)
    scoreRect = scoreSurf.get_rect()
    scoreRect.topleft = (WINDOWWIDTH - 150, 20)
    DISPLAYSURF.blit(scoreSurf, scoreRect)

    # draw the level text
    levelSurf = BASICFONT.render('Level: %s' % level, True, TEXTCOLOR)
    levelRect = levelSurf.get_rect()
    levelRect.topleft = (WINDOWWIDTH - 150, 50)
    DISPLAYSURF.blit(levelSurf, levelRect)
```

플레이 시간과 점수, 현재 레벨을 스크린에 텍스트로 출력해주는 함수이다. 기본 코드에 시간 경과를 알려주는 코드를 추가하여 수정해주었다.

drawPiece()

```
def drawPiece(piece, pixelx=None, pixely=None):
    shapeToDraw = PIECES[piece['shape']][piece['rotation']]
    if pixelx == None and pixely == None:
        # if pixelx & pixely hasn't been specified, use the location stored in the piece data structure
        pixelx, pixely = convertToPixelCoords(piece['x'], piece['y'])

    # draw each of the boxes that make up the piece
    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))
```

떨어지는 블록이나 다음에 나올 블록을 그릴 때 사용하는 함수이다. 받은 피스의 값에 해당하는 모양, 회전 정도를 가진 블록을 주어진 좌표에 그린다. For 문을 사용해 drawBox() 함수를 호출하여 모양에 맞게 그려낸다. 만약, 좌표가 주어지지 않았다면 화면 중앙에 그린다.

drawNextPiece()

```
def drawNextPiece(piece):
    # draw the "next" text
    nextSurf = BASICFONT.render('Next:', True, TEXTCOLOR)
    nextRect = nextSurf.get_rect()
    nextRect.topleft = (WINDOWWIDTH - 120, 80)
    DISPLAYSURF.blit(nextSurf, nextRect)
    # draw the "next" piece
    drawPiece(piece, pixelx=WINDOWWIDTH-120, pixely=100)
```

스크린의 오른쪽에 다음에 떨어질 블록을 나타내는 함수이다. Next 텍스트와 함께, 다음에 떨어질 블록을 drawPiece() 함수를 통해 그려낸다.

2. 함수 호출 순서 및 호출 조건

함수 호출 순서

mian() 함수를 호출 → main() 함수에서 runGame() 함수 호출

→ runGame() 함수에서 getBlankBoard() 함수를 호출하여 빈 보드를 생성 (초기화)

→ runGame() 함수에서 getNewPiece() 함수를 두번 호출하여 각각, 이번에 떨어질 블록(fallingPiece)과 다음에 떨어질 블록(nextPiece)으로 결정

→ runGame() 함수의 게임 작동을 위한 while 반복문 시작 [게임 루프]

→ fallingPiece가 없으면 nextPiece가 위에서 떨어지도록 하고 nextPiece에 getNewPiece() 함수를 호출해 새로운 블록 할당

→ checkForQuit() 함수를 호출해 종료 이벤트 발생 여부 확인 for 반복문 시작 [이벤트 처리 루프]

→ 플레이어가 p키를 클릭하면 게임 중단

→ showTextScreen() 함수를 호출해 정지되었다는 텍스트 출력

→ 블록 움직임 관련 변수가 입력되면 isValidPosition() 함수 호출

→ isValidPosition() 함수는 반복적으로 호출되며 블록이 겹치지 않도록 도움

→ 블록이 바닥에 닿으면 addToBoard() 함수를 통해 바닥에 쌓인 모습을 보드 판에 추가

→ 게임이 진행되다가 한 줄이 블럭으로 가득 차게 되면 removeCompleteLine() 함수를 호출하여 채워진 줄을 지우고, 지워진 줄 수를 점수에 더함

→ 점수가 10점까지 쌓이면 calculateLevelAndFallFreq() 함수 호출

→ 레벨을 1씩 올리며, 블록 위에 떨어지는 속도 빠르게 조정

→ fallingPiece가 비어있지 않다면 drawPiece() 함수를 호출해 fallingPiece에 저장된 블록을 보드에 그림

→ 이벤트 감지를 위한 for문 종료

→ drawboard() 함수 호출해 바뀐 게임 화면 그림

→ drawStatus() 함수 호출해 next 블록을 오른쪽 화면에 그림

→ 다시 runGame() 함수의 while 반복문의 시작으로 돌아가서 반복

→ 게임 도중 블럭이 끝까지 쌓이거나 플레이어가 esc키를 누르는 경우 checkForQuit() 함수 호출

→ terminate() 함수가 호출되어 게임 종료

함수 호출 조건

makeTextObjs() : showTextScreen() 함수에서 출력할 텍스트를 만들기 위해 호출

terminate() : checkForQuit() 함수에서 종료 이벤트를 감지하면 호출

isOnBoard() : isValidPosition() 함수에서 블록이 보드 밖에 있는지 확인하기 위해 호출

isCompleteLine() : removeCompleteLine() 함수에서 채워진 줄이 있는지 확인하기 위해 호출

drawBox() : drawboard()와 drawPiece() 함수에서 블록을 이루는 상자들을 그리기 위해 호출