```
#TTT_interface.py - Handles All Interface Calls For Tic Tac Toe
#Ari Cohen

import pygame, sys
from pygame.locals import *
from buttons import *
from TTT_state import *

pygame.init()
MAIN_SURF = pygame.display.set_mode((800, 600))
pygame.display.set_caption('Tic Tac Toe!')

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
GREEN = (0, 255, 0)
LIGHT_GREEN = (0, 100, 0)
BLUE = (0, 0, 128)
YELLOW = (255, 255, 0)
LIGHT_YELLOW = (100, 100, 0)
RED = (255, 0, 0)
LIGHT_RED = (100, 0, 0)
GRAY = (150, 150, 150)


class Board():

    def __init__(self, position, board_width, square_width):
        self.position = position
        self.total_squares = board_width * board_width
        self.board_width = board_width
        self.square_width = square_width
        self.squares = []
        for square_num in range(self.total_squares):
            new_square = Square(square_num, self)
            self.squares.append(new_square)
        width = (board_width * square_width) + (5 * (board_width - 1))
        self.surface = pygame.Surface((width, width))
        self.board_rect = pygame.Rect(position[0], position[1], width, width)

    def set_square(self, row, col, piece):
        self.squares[(self.board_width * row) + col].click_action(piece)

    def get_square_coords(self, row, col):
        (x, y) = self.squares[(self.board_width * row) + col].get_location()
        x += self.position[0]
        y += self.position[1]
        return (x,y)

    def update_board_surface(self):
        self.surface.fill(BLACK)
        for square in self.squares:
            square_surf = square.get_square_surface()
            self.surface.blit(square_surf, square.position)
        return self.surface

    def draw_board(self):
        board_surf = self.update_board_surface()
        MAIN_SURF.blit(board_surf, self.position)

    def check_for_mouse(self, dragging_piece):
        mouse_x, mouse_y = pygame.mouse.get_pos()
        if self.board_rect.collidepoint([mouse_x, mouse_y]):
            mouse_x -= self.position[0]
            mouse_y -= self.position[1]
            col = mouse_x / (self.square_width + 5)
            row = mouse_y / (self.square_width + 5)
            bad_drag = self.squares[(self.board_width * row) + col].click_action
                (dragging_piece)
            if(bad_drag):
                return [False, [row, col]]
            else:
                return [True, [row, col]]
        else:
            return [False, 0]

    def do_event_fetch(self):
        MAIN_SURF.fill(WHITE)
        self.draw_board()

        for event in pygame.event.get():
            if(event.type == QUIT):
                pygame.quit()
                sys.exit()
            elif(event.type == KEYDOWN):
                if event.key == K_ESCAPE:
                    pygame.quit()
                    sys.exit()
                key_map = pygame.key.get_pressed()
                return key_map
            elif(event.type == MOUSEBUTTONDOWN):
                return MOUSEBUTTONDOWN
            elif(event.type == MOUSEBUTTONUP):
                return MOUSEBUTTONUP

    def get_square_piece(self, row, col):
        return self.squares[(self.board_width * row) + col].get_piece()

class Square():

    EMPTY = 0

    def __init__(self, square_number, board):
        self.square_num = square_number
        self.board = board
        self.width = self.board.square_width
        row = self.square_num / self.board.board_width
        col = self.square_num % self.board.board_width
        self.position = ((self.width + 5) * col, (self.width + 5) * row)
        self.surface = pygame.Surface((self.width, self.width))
        self.surface.fill(WHITE) #Square colors
        self.piece = Square.EMPTY
    def get_val(self):
        pass

    def set_val(self):
        pass

    def get_location(self):
        return self.position

    def get_square_surface(self):
        return self.surface

    def click_action(self, piece_type):
        if(self.piece == Square.EMPTY):
            self.piece = piece_type
            draw_piece(self.surface, piece_type, (0,0), GREEN, self.board)
            return False
        else:
            return True

    def get_piece(self):
        return self.piece

def display_game_state(game_state):
    interface_state = game_state.get_interface()
    interface_state.do_event_fetch()
    pygame.display.update()

def make_move_and_display(game_state, move):
    board = game_state.interface_state
    game_state.make_move(move)
    game_state.toggle_players()
    row = move.get_row_placement()
    col = move.get_col_placement()
    if (board.get_square_piece(row, col) == Square.EMPTY):
        board.set_square(row, col, move.get_piece())
    display_game_state(game_state)

def get_human_move(game_state):
    board = game_state.get_interface()
    current_player = game_state.get_current_player()
    next_piece = game_state.get_current_piece()
    while True: #Loop until user has clicked a good square
        events = board.do_event_fetch()
        pygame.display.update()
        if (events == MOUSEBUTTONDOWN):
            [if_good, position] = board.check_for_mouse(next_piece)
            if (if_good):
                break

    move = GameMove()
    move.set_move(position[0], position[1], next_piece)
    return [move, GameStatus.PLAYING]


def get_players_information(board):
```

```
    player_radios = [RadioButtonGroup((200, 50), 2, MAIN_SURF,
                                      names=["Human", "Computer"],
                                      color=BLACK, text_size=32),
                     RadioButtonGroup((400, 50), 2, MAIN_SURF,
                                      names=["Human", "Computer"],
                                      color=BLACK, text_size=32)]
    button_font = pygame.font.Font('freesansbold.ttf', 52)
    text_surface = button_font.render("Go!", True, BLACK, GREEN)
    text_rect = text_surface.get_rect()
    text_rect.topleft = (350, 250)

    while True: #Wait for "Go!" to be pressed
        events = board.do_event_fetch()
        MAIN_SURF.fill(WHITE)
        player_radios[0].draw_surface()
        player_radios[1].draw_surface()
        MAIN_SURF.blit(text_surface, (350, 250))
        pygame.display.update()

        if (events == MOUSEBUTTONDOWN):
            for radio in player_radios:
                radio.check_for_click()
            if (text_rect.collidepoint(pygame.mouse.get_pos())):
                break
    vals = ["h", "c"]
    selected = []
    for radio in player_radios:
        selected.append(vals[radio.get_selected()])
        selected.append(3) #This would be computer strength
    return selected

def signal_end_of_game(game_status, game_state, player_1,
                       player_2, current_player):
    interface_state = game_state.interface_state
    if game_status == GameStatus.TIE:
        game_text = "It's a tie."
    elif game_status == GameStatus.WIN:
        game_text = "Player "+current_player.player_num+" wins!"
    else:
        game_text = "Game over - unknown reason"

    my_font = pygame.font.Font('freesansbold.ttf', 52)
    text_surface = my_font.render(game_text, True, BLACK, YELLOW)
    button_surface = my_font.render("Play Again!", True, BLACK, YELLOW)
    button_rect = button_surface.get_rect()
    button_rect.topleft = (300, 520)

    while True:
        events = interface_state.do_event_fetch()
        MAIN_SURF.blit(text_surface, (250, 20))
        MAIN_SURF.blit(button_surface, (300, 520))
        pygame.display.update()
        if (events == MOUSEBUTTONDOWN):
            if (button_rect.collidepoint(pygame.mouse.get_pos())):
                break
def draw_piece(surf, piece, location, color, board):
    rect = surf.get_rect()
    if (piece == Piece.O):
        for x in range(20):
            board.do_event_fetch()
            #6.283 is approximately 2*pi
            pygame.draw.arc(surf, color, rect, 0, 6.283*x/20, 4)
            pygame.display.update()
        pygame.draw.ellipse(surf, color, rect, 4)
    elif (piece == Piece.X):
        width = rect.width
        for x in range(10):
            new_spot = x*width/10
            board.do_event_fetch()
            pygame.draw.line(surf, color, rect.topleft, (new_spot, new_spot), 4)
            pygame.display.update()
        for x in range(10):
            new_spot = width - (x*width/10)
            board.do_event_fetch()
            pygame.draw.line(surf, color, rect.topright, (new_spot, x*width/10),
                3)
            pygame.display.update()
        pygame.draw.line(surf, color, rect.topleft, rect.bottomright, 4)
        pygame.draw.line(surf, color, rect.topright, rect.bottomleft, 4)
```