Alex Steven Bonilla Castaño
Juan Pablo González Casado

Scrabble

Introduction:

In this project, the goal is to implement a typical game for social networks Scrabble in which players take turns networked to form words with letters that the server will provide it randomly.

The ultimate goal of the game is to get the maximum number of points depending on the words created on the board. Players also have the posibility to exchange messages between them.

To create this game, we used the Python programming language in conjuntion with the Pygame library for graphics and game Mastermind library for client-server network.

Rules:

1. The player can not form more than one word in a single turn.
2. When the player form more than one word with the same token, the game will take as valid the horizontal word, excluding the vertical.
3. If the player sends a word that does not exist, it will lose the tokens sent and the turn.
4. Words only can be formed horizontally with west to east and vertical north-south direction.
5. On each turn, the player has the posibiity to request as many tokens as the player wants, with a limit of 8.

Implementation:

The game is based on a client-server on which the server controls the entire game, movements, score and messages between the clients.

The client is responsible for creating a comfortable and simple interface for the player, can send messages and word combinations depending on the state of the game.

Alex Steven Bonilla Castaño
Juan Pablo González Casado

Server

The server has been developed with the Mastermind library, responsible for creating TCP connections between players. The code is based on the examples provided by the author of the library, adding more some features to use in this game.

The function responsible for managing messages from clients is as follows:

```
def callback_client_handle    (self, connection_object, data
):
```

In this feature we can see how the server treats differently each message that arrives, performing a function and returning a client data if necessary.

```
if cmd == "introduce":
    self.currentgame.add_player(data[1])
    self.add_message("Server: "+data[1]+" has joined.")
elif cmd == "add":
    self.add_message(data[1])
elif cmd == "update":
    pass
elif cmd == "gettoken":
    fichas = self.currentgame.addTokenToPlayer(data[1])
elif cmd == "getstate":
    state_player = self.currentgame.getPlayerState(data[1])
elif cmd == "removetoken":
    fichas = self.currentgame.removeToken(data[1],data[2])
elif cmd == "gettokenlist":
    fichas = self.currentgame.getTokenListByPlayer(data[1])
elif cmd == "sendgrid":
    grid =
self.currentgame.compareGrids(self.currentgame.grid,data[2],data[1])
elif cmd == "leave":
    if self.currentgame.del_player(data[1]):
        self.add_message("Server: "+data[1]+" has left.")

//Callback
if fichas != None:
    self.callback_client_send(connection_object, fichas)
elif state_player != None:
    self.callback_client_send(connection_object, state_player)
elif grid != None:
    for c in self._mm_connections.items():
      self.callback_client_send(c.__getitem__(1), grid)
else:
    self.callback_client_send(connection_object, self.chat)
```

Alex Steven Bonilla Castaño
Juan Pablo González Casado

Class game:

In this class is all game information, contains variables which indicate the status, a list of players and is responsible for checking the words that the client sent.

```
def __init__(self):
    self.player_list = []
    self.queue = deque([])
    self.data = "NO DATA"
    # MATRIZ LOGICA PARA TABLERO
    self.grid = [[-1 for i in xrange(TAMANO)] for i in xrange(TAMANO)]
    self.scrabble = scrabble.Scrabble()
    self.current_player = ""
```

Class player:

This class represents the data that each player handles. Score, tokens, ID and Name. It is used by the class 'game' to create a player every time that someone connects and add it to the list of players.

```
class Player(object):
    def __init__(self,id,name,n_tokens,scrabble_ins):
        self.ID = id
        self.Name = name
        self.Tokens = []
        self.getTokens(n_tokens,scrabble_ins)
        self.Points = 0
```

Class scrabble:

The 'Scrabble' class has a list of all the letters available, both its value as the number of tokens available for each letter.

The class 'game' makes tokens requests to this class, returning random tokens in the case still remain available tabs.

```
A = ["A",1,16]
B = ["B", 3, 4]
C = ["C", 3, 6]
D = ["D", 2, 8]
E = ["E", 1, 24]
F = ["F", 4, 4]
G = ["G", 2, 5]
H = ["H", 4, 5]
I = ["I", 1, 13]
J = ["J", 8, 2]
```

Alex Steven Bonilla Castaño
Juan Pablo González Casado

```
K = ["K", 5, 2]
L = ["L", 1, 7]
M = ["M", 3, 6]
N = ["N", 1, 13]
O = ["O", 1, 15]
P = ["P", 3, 4]
Q = ["Q", 10, 2]
R = ["R", 1, 13]
S = ["S", 1, 10]
T = ["T", 1, 15]
U = ["U", 1, 7]
V = ["V", 4, 3]
W = ["W", 4, 4]
X = ["X", 8, 2]
Y = ["Y", 4, 4]
Z = ["Z", 10, 2]
letras = [A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z]


class Scrabble(object):
    def __init__(self):
        self.totals_letters= 196
        self.count_letters=0
```

Main

The 'main' class develops all the creation and management of user interface, as well as connections from the client to the server.

To send messages to the server, the following function is used:

```
to_send.append(["message", player_name])
```

The 'draw map ()' function is responsible for creating the game board.

```
def draw_map(self):
```

In the function 'send next blocking' the data sent by the server is managed, receiving and updating the game state.

```
def send_next_blocking(self):
```

'Draw_chat' and 'draw_text' painted mainly text. In the case of 'draw_chat' refers to the area where players can send messages between them and in the case of 'draw_text' refers to any text in a specific position we want to paint on the screen, such as score or shift.

```
def draw_chat(self):
def draw_text(self, text,pos_x,pos_y):
```

Alex Steven Bonilla Castaño
Juan Pablo González Casado

'Handler_click' is the function responsible for managing the clicks in each area of the screen. That is, if we are clicking on the tokens of a player, this will shade the tab to display currently selected, if later an empty square on the board is clicked, this token will be introduced in the square on the board. All these features interaction with the graphical interface are discussed in this function.

```
def handler_click(self,event):
      if event.type == pygame.MOUSEBUTTONDOWN:
        pos = pygame.mouse.get_pos()
```

INTERFACE