

Journal De Bord - Projet Scrabble

Jules / Charles / Isabella

JANUARY 12 2023

Modelisation/ functionality thoughts:

- Empty table displayed (later- need to find a way to color the multiplier boxes -- maybe pandas/matplotlib)
- User receives 7 random letters
 - Maybe have a list of all the letters possible (can be generated based on the list provided by Mr. Lorget of available letters per word)? And then gradually pop them from the list as they're used?
 - **TASK** → write code to convert the dictionary of num of available letters to a list with all the letters and their respective amounts
- Board is displayed (via PrettyTable -- make a list then pass it into a PrettyTable object)
- User is prompted to enter a word, a start coordinate (have it indexed by 1 for more understandability) and an end coordinate (or direction - horizontal/vertical)

```
for coordinate, letter in enumerate(list of coordinates, word):
    if its a letter:
        check if it's in the users inventory
        check if coordinate is empty
    else:
        check if coordinate has a letter
```

- Place word on board, re update the display (clear output + reprint prettytable)
 - **TASK** → write function to place word
 - **TASK** → write function to clear output and update the display
- prompt user to place next word (or prompt second player-- maybe start w/ one user and then add a second player)

Different functions to write:

- Generate all letters
 - Ran ONCE at the beginning of the game

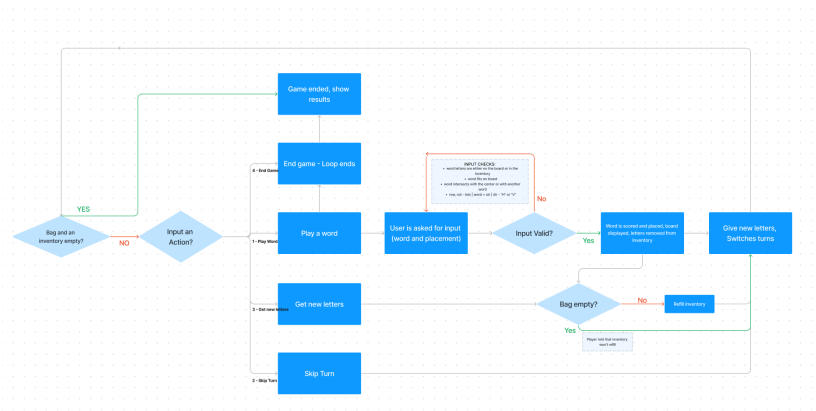
- Turns the letter: available dictionary into one list where the elements are just all the letters * their respective amounts
- No return value
- Add to constant called BAG
- Generate random letters
 - Will have to check available letters
 - Take into account current amount of letters user has (user should always have 7)
 - Remove them from the giant list of all letters until BAG has no letters left
 - No return value - update global inventory variables
- Get Input
 - No args
 - Returns the desired input - if any input is false, returns -1 for row and col which will trigger False in validate
- Validate
 - Three args: word, direction (horizontal/vertical), start coordinate (row + col)
 - Only triggered when get_input is True
 - Generates coordinates
 - Will have to iterate through each letter - if the c
 - call function word_played()
- Print display
 - Appends the letter to a string w/ appropriate color based on whether or not it's on a multiplier
 - Inventory should be printed after
- Inventory display, invprint function that takes player inventory and player number to put in a legible way the inventory and who it's for
- Board init
 - Initializes board to a bunch of blank spaces, places a star in the middle for aesthetic purposes
- Score
 - The word is valid so scores based on multipliers + whether or not the word is a joker

FIGJAM - USER FLOW

(with extra details related to how the program will handle it):

<https://www.figma.com/file/aKWqNQgLB51WGTX3orUxsa/User-Flow---Scrabble-NSI?node-id=0%3A1&t=jaP26T2SZyOXJ46x-1>

(thumbnail below)



```
main.py × +
main.py

1 # using pretty table to format the 2D array nicely in the console
2 from prettytable import PrettyTable
3
4 # constant for horizontal rule argument
5 from prettytable import ALL
6
7 JETONS_PTS = {"A":1,"B":3,"C":3,"D":2,"E":1,"F":4,"G":2,"H":4,"I":1,
8              "J":8,"K":10,"L":1,"M":2,"N":1,"O":1,"P":3,"Q":8,"R":1,
9              "S":1,"T":1,"U":1,"V":4,"W":10,"X":10,"Y":10,"Z":10,
10             "joker":0}
11
12 MULTIS = {(0,0):('m',3), (0,7):('m',3), (0,14):('m',3), (7,0):('m',3),
13          (7,14):('m',3), (14,0):('m',3), (14,7):('m',3),
14          (14,14):('m',3), (1,1):('m',2), (1,13):('m',2), (2,2):('m',2),
15          (2,12):('m',2), (3,3):('m',2), (3,11):('m',2), (4,4):('m',2),
16          (4,10):('m',2), (7,7):('m',2), (10,4):('m',2), (10,10):('m',2),
17          (11,3):('m',2), (11,11):('m',2), (12,2):('m',2), (12,12):('m',2),
18          (13,1):('m',2), (13,13):('m',2), (1,5):('l',3), (1,9):('l',3),
19          (5,1):('l',3), (5,5):('l',3), (5,9):('l',3), (5,13):('l',3),
20          (9,1):('l',3), (9,5):('l',3), (9,9):('l',3), (9,13):('l',3),
21          (13,5):('l',3), (13,9):('l',3), (0,3):('l',2), (0,11):('l',2),
22          (2,6):('l',2), (2,8):('l',2), (3,0):('l',2), (3,7):('l',2),
23          (3,14):('l',2), (6,2):('l',2), (6,6):('l',2), (6,8):('l',2),
24          (6,12):('l',2), (7,3):('l',2), (7,11):('l',2), (8,2):('l',2),
25          (8,6):('l',2), (8,8):('l',2), (8,12):('l',2), (11,0):('l',2),
26          (11,7):('l',2), (11,14):('l',2), (12,6):('l',2), (12,8):('l',2),
27          (14,3):('l',2), (14,11):('l',2)}
28
29 def score_word(word, coord):
30     pass
31
```

- Wrote score function

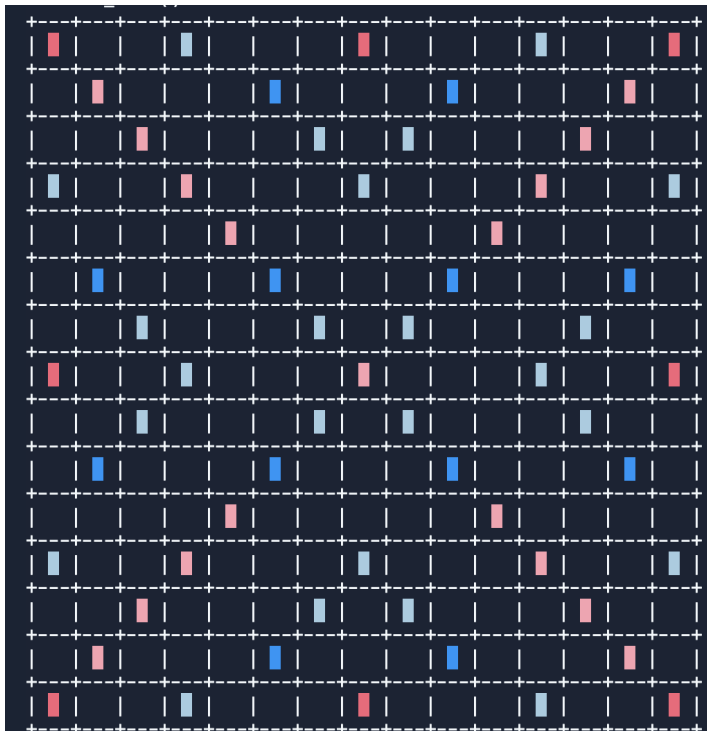
```


29 def score(word, coord_x, coord_y, dir):
30     score = 0
31     multiply_word = 1
32     word = word.upper()
33     letters = [l for l in word]
34
35     if dir == 'V':
36         coordinates = [(coord_x, coord_y + y) for y in range(len(word))]
37     else:
38         coordinates = [(coord_x + x, coord_y) for x in range(len(word))]
39     for array in coordinates:
40         for pos in array:
41             if pos > 14:
42                 raise "This is not a valid placement. Please try again."
43
44     letter_placement = dict(zip(coordinates, letters))
45     for place, let in letter_placement.items():
46         if place in MULTIS.keys():
47             #case 1: it's on a multiplier
48             type, num = MULTIS[place]
49             if type == 'l':
50                 # multiplies the letter by the amount necessary
51                 score += JETONS_PTS[let]*num
52             else:
53                 # it adds to the word multiplier, adds regular amount to score
54                 multiply_word *= num
55                 score += JETONS_PTS[let]
56         else:
57             # it's not on a multiplier - add regular amount
58             score += JETONS_PTS[let]
59     return score*multiply_word, multiply_word
60


```


JANUARY 14 2023

- Fixed colors (ANSI escape codes → <https://stackoverflow.com/questions/4842424/list-of-ansi-color-escape-sequences>)



 = double word

 = triple word

 = double letter

 = triple letter

- Upon entering a word, console is cleared (via os module)
- Starbuck assisted-- professional programmer (specializes in concatenation)



JANUARY 15 2023

- Modified place_letters() so that it makes sure the word intersects with another word or is the first word placed
- Created the loop of asking for the word and necessary information for placement (using the score function, available letters, and player number)

JANUARY 17 2023

- Modified the letteruse variable check to include doubles of a letter, by temporarily removing them from inventory (temp_letters variable for temporarily removed letters), and made it so that one letter not being in inventory disqualified the word (bug made it so that if the last letter worked the word worked)
 - Debugged! There was an indentation error and temp_letters needed to be added back into the inventory the moment a letter was wrong

(old main function:

```
def main_2():
    game = True
    letteruse = False
    temp_letters = []

    b = 0 #used as a break to not get out of all loops
    while letteruse == False:
        b = 0
        temp_letters = []
        for elt in word:
            if endgame == "1":
                if elt not in inventory1 and elt != '-':
```

```

print(f"You do not have the letter: {elt}")
print(temp_letters)
inventory1.extend(temp_letters)
letteruse = False
b = 1 # your input is wrong - letteruse cannot be set to true
elif b == 0:
    inventory1.remove(elt)
    print(elt + " was removed")
    print(inventory1)
    temp_letters.append(elt)
    letteruse = True
elif endgame == "2":
    if elt not in inventory2 and elt != '-':
        letteruse = False
        b = 1
    elif b == 0:
        inventory2.remove(elt)
        temp_letters.append(elt)
        letteruse = True
if endgame == 1:
    for elt in temp_letters:
        inventory1.append(elt)
        print(inventory1)
if endgame == 2:
    for elt in temp_letters:
        inventory2.append(elt)
temp_letters = []
)

```

- Reorganizing input into a separate function

JANUARY 18 2023

- Made the give_letter() fonction, it gives a letter to a player from the list of letter BAG and removes it from BAG. It can give any number of letters to any of the players.
 - Debugged it and made it so it can be used to set the letters at the start of the game and for whenever a word is made from the inventory of a player

```

def give_letters(num,player):
    letter = ''
    for i in range(num):
        letter = random.choice(BAG)
        BAG.remove(letter)
    if player == 1:
        inventory1.append(letter)
        return(inventory1)
    else:
        inventory2.append(letter)
        return(inventory2)

```

JANUARY 19 2023

- Added use of _ as a blank letter, replacement on board, in word, and in inventory by asking for an input of the letter wanted then replacing everywhere

JANUARY 20 2023

- Debugged give_letters() + modified so that only one argument is required

JANUARY 21 2023

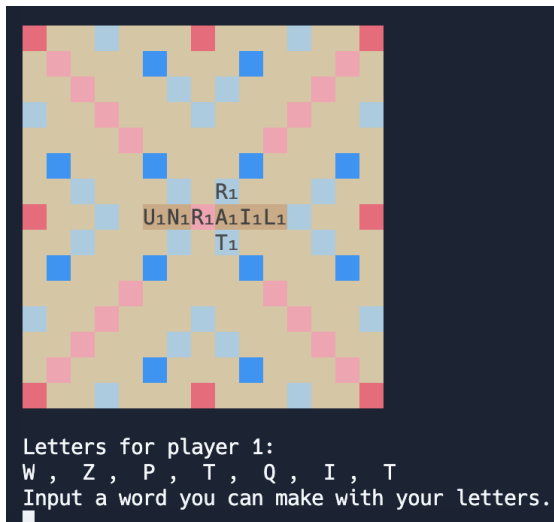
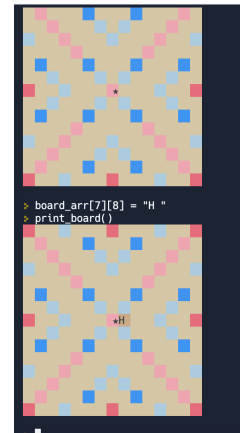
- Debugged place letters and scoring
 - Before: couldn't intersect words, board wouldn't update

JANUARY 21 2023

- Debugged the END function that would not work
 - A part of the code was missing

JANUARY 25 2023

- Changed format for board-- ANSI escape codes were annoying in booleans-- wrote function to print out the board by appending characters to a string
- Made a printable inventory function `invprint(inventory, player number)` separate from the actual inventories
- Changed display formatting so that the points per letter show - + = 10
- Wrote validate function + fixed `get_input` so that getting the information from the user and validating it were in two separate functions
- Fixed display formatting to color tiles before multipliers



JANUARY 26 2023

- Added unicode numbers for numbering the grid and changed positioning of center star (issue is that each box is 2/2, so symbols had to be in the bottom left corner and therefore only be one character)
- Fixed validate
- Fixed issue with double letters counting as one (temp_letters not being used anymore)
- Made sure turns only switch when a word is put on the board (other than skip and shuffle words), not when you make a mistake with word placement or have an incorrect input

JANUARY 28 2022

- Fixed turn switching (didn't switch if user messed up input)
- Restructured returns so that they were consistent no matter the error (or lack thereof)
- Fixed intersection problems
- Integrated joker replacement code after scoring

JANUARY 30 2022

- Fixed joker display bug (it would replace the score with the letter's score before)



JANUARY 31 2022

- Warning when bag is empty and letters won't be refilled + reorganized give_letters() to account for letters running out (a bag that didn't have enough letters to completely fill someone's inventory again now stops once it's out)

- Edited the bag empty part to not immediately give score and winner and end game
- Option each round to end game, says who won (people can choose to stop playing when they don't have any words, not when bag is empty)
- Word with an incorrect direction no longer default to vertical but rather prompts a resubmission

OTHER CREDITS:

- Using two places to display each tile -- idea taken from Samy/Chris/Esme
- One character ASCII two digit numbers found by Samy

Il précisera à la fin

si certaines fonctionnalités initialement prévues et analysées n'ont pas pu être réalisées (et pourquoi).

CHANGEMENTS AU COURS DU PROJET:

- PrettyTable → String qui montre le board
 - C'était difficile de comparer les valeurs avec les ANSI escape codes-- plus facile d'imprimer un string qui montre les valeurs dans board_arr (qui contient seulement les lettres placés + leurs scores)
- Structure du code
 - Le code est assez similaire, on a simplement réorganisé pour que tout qui était lié à la validation du saisie soit dans un endroit -- plus facile à lire + utiliser dans le main function
- Score du joker
 - Les jokers apportent 0 points, comme dans le jeu Scrabble - avant, ça comptait pour la lettre choisie