

Project Com Pro 1

หัวข้อ “Wordle”

จัดทำโดย

รหัส 6504062620159 นายโมกซ์ มาอาจ ลำดับที่ 16

รหัส 6504062620175 นายอชิชา เล็กสรรเสริญ ลำดับที่ 17

รายวิชา 040613201 COMPUTER PROGRAMMING I

ปีการศึกษา 2565

บทที่ 1

1. วัตถุประสงค์ของโปรแกรม

- 1.1 ใช้เป็นเกมแก้ปริศนาคำว่างได้
- 1.2 เป็นเกมบริหารสมอง เนื่องด้วยต้องใช้ทักษะ วิเคราะห์ และ recall
- 1.3 ฝึกคำศัพท์ภาษาอังกฤษ
- 1.4 เป็น project สำหรับการนำความรู้ด้าน coding มาประยุกต์

2. หลักการทำงานของโปรแกรม

- 2.1 โปรแกรมจะถูกทำงาน เมื่อเรารัน execute
- 2.2 ทำการสร้างหน้า GUI
 - 2.2.1 หน้าแสดงผล
 - 2.2.2 กล่องตอบ
 - 2.2.3 กราฟิกคีย์บอร์ด ที่สามารถคลิก แล้วแสดงผลแบบคีย์บอร์ด
 - 2.2.4 ปุ่มไปหน้า stats window
- 2.3 สุ่มคำจากไฟล์ คำศัพท์
- 2.4 เมื่อหน้าแสดงผลขึ้นมาแล้วจึงกรอก คำศัพท์ที่มีความหมาย และมี 5 ตัวอักษรลงไป
 - 2.4.1 หากคำศัพท์ไม่มีความหมายให้สั่งให้ กรอกใหม่ แต่คงคำศัพท์ไว้
- 2.5 ทำการเช็คตัวอักษรในคำศัพท์ ตรงในเฉลยไหม
 - 2.5.1 หากตัวอักษรตรงกันแบบ ตำแหน่งเดียวกันให้ หน้าแสดงผล เป็นสีเขียว
 - 2.5.2 หากตัวอักษรอยู่ในเฉลย แต่ไม่ได้อยู่ตรงตำแหน่ง หน้าแสดงผล เป็นสีเหลือง
 - 2.5.3 ทำการเปลี่ยนสีคีย์บอร์ด ตามอักษรของคำตอบ
- 2.5 ทำแบบนี้ 6 รอบ
- 2.6 อัปเดต history ไปสู่ไฟล์ csv
- 2.7 หากกดหน้า stat window ให้ใช้สถิติการเล่น

3. การเล่น

เราจะสุ่มคำมา 5 ตัวอักษรซ่อนไว้ จากนั้นให้ผู้เล่นกรอกคำศัพท์ที่มีความหมาย และมี 5 ตัวอักษร ให้ถูกภายใน 6 ครั้ง มาโดยโปรแกรมจะตรวจสอบว่า ในคำนั้นมีตัวอักษรที่ตรงกับคำที่ซ่อนไว้ไหม และตรงกับตำแหน่งนั้น ๆ เลยหรือเปล่า เป็น 3 สีได้แก่



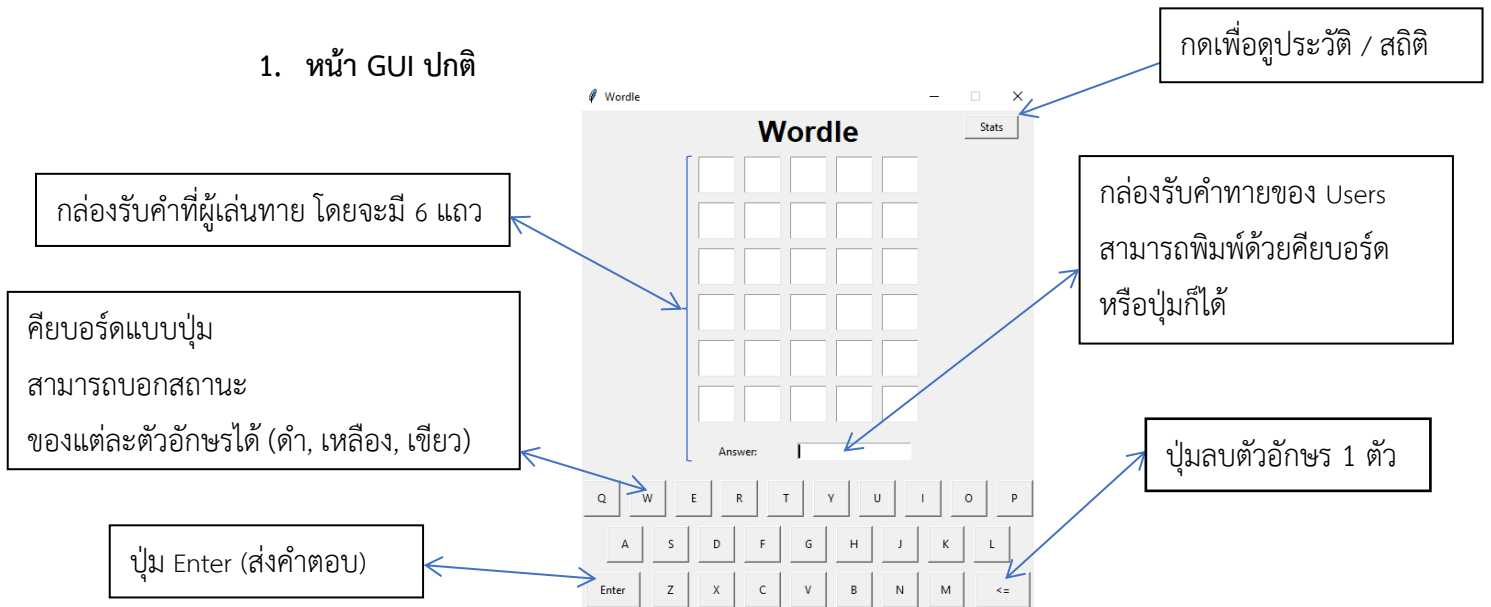
รูป 1.1 เปรียบเทียบสัญลักษณ์สีในเกม “wordle”

- สีเขียว : อักษรที่ทายลงไปในนั้น ถูกทั้งตัวและตำแหน่ง
- สีเหลือง : อักษรที่ทายลงไปในนั้น ถูกตัวแล้ว แต่ไม่ถูกตำแหน่ง
- สีดำ : อักษรที่ทายลงไปในนั้น ไม่ถูกทั้งตัวอักษร และตำแหน่ง

บทที่ 2

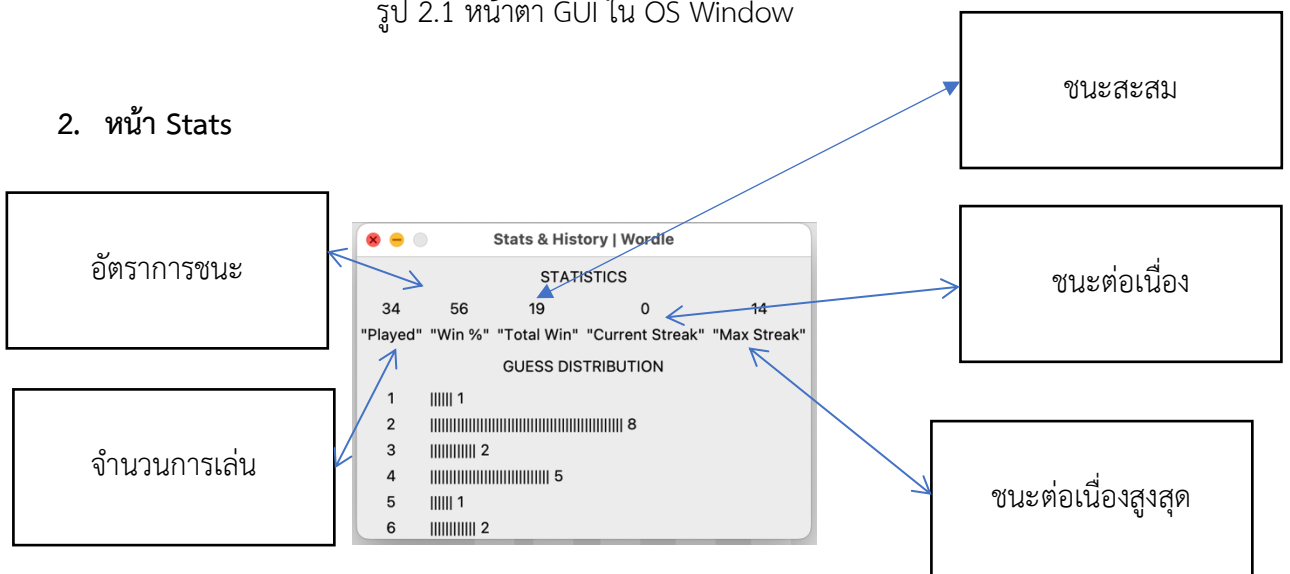
User interface

1. หน้า GUI ปกติ



รูป 2.1 หน้าตา GUI ใน OS Window

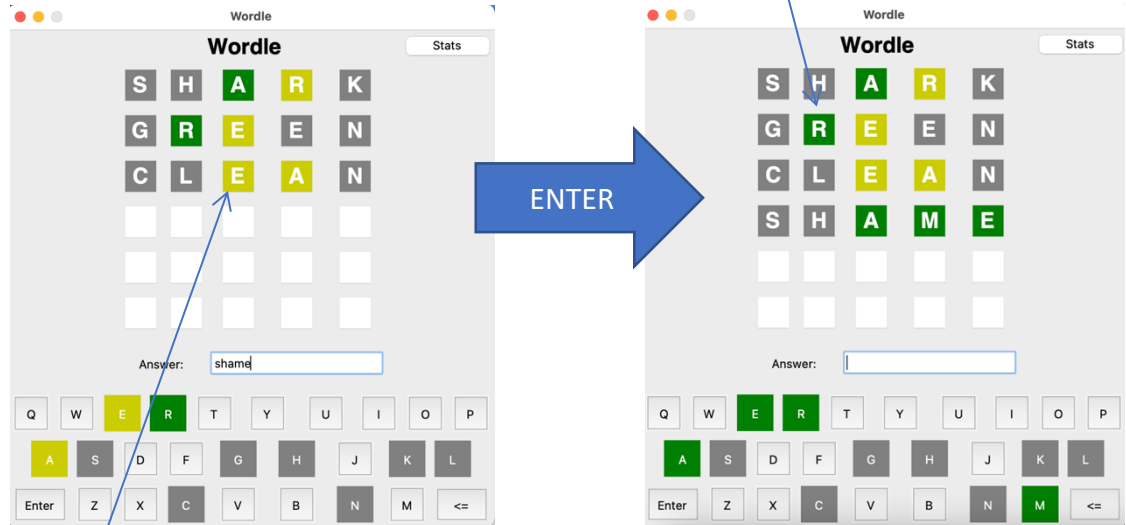
2. หน้า Stats



รูป 2.2 หน้าตาของหน้าต่าง Stats & History | Wordle

3. เมื่อกรอกคำศัพท์ไปแล้ว

หากเรากด ตัวอักษรของคำตอบเรา ตรงกับ
เฉลยที่ตำแหน่งนั้น ๆ จะเป็นคำใบ้สีเขียว

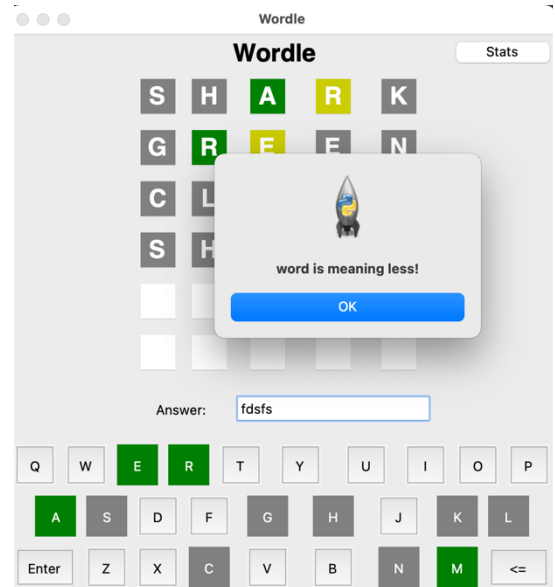
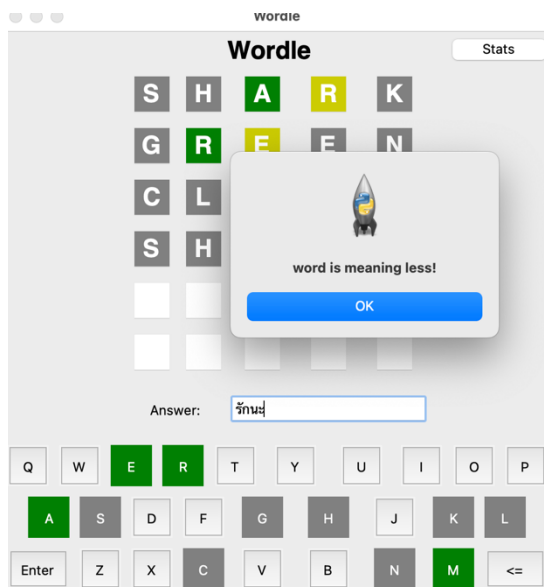


รูป 2.3 รูปเปรียบเทียบการเปลี่ยนสีของ ตาราง และคีย์บอร์ด

หากในคำตอบ ตัวอักษรของคำตอบเรา อยู่ในเฉลย
แต่ไม่ตรงตำแหน่งจะเป็นคำใบ้สีเหลือง

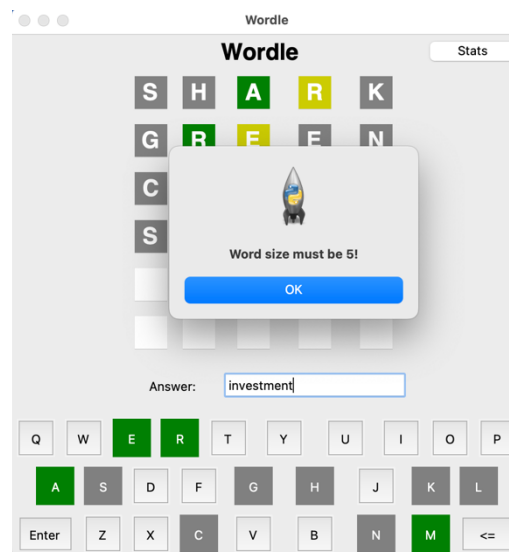
โดยเฉลยคือ “ frame ” ที่ “AME” ของ “shame” เป็นสีเขียว เพราะตัวอักษรตรงตำแหน่งกันทั้งเฉลย และ คำตอบ

4. เมื่อกรอกคำศัพท์ที่ไร้ความหมายหรือ ไม่ใช่ภาษาอังกฤษ จะขึ้นแจ้งเตือนว่า “word is meaning less!”



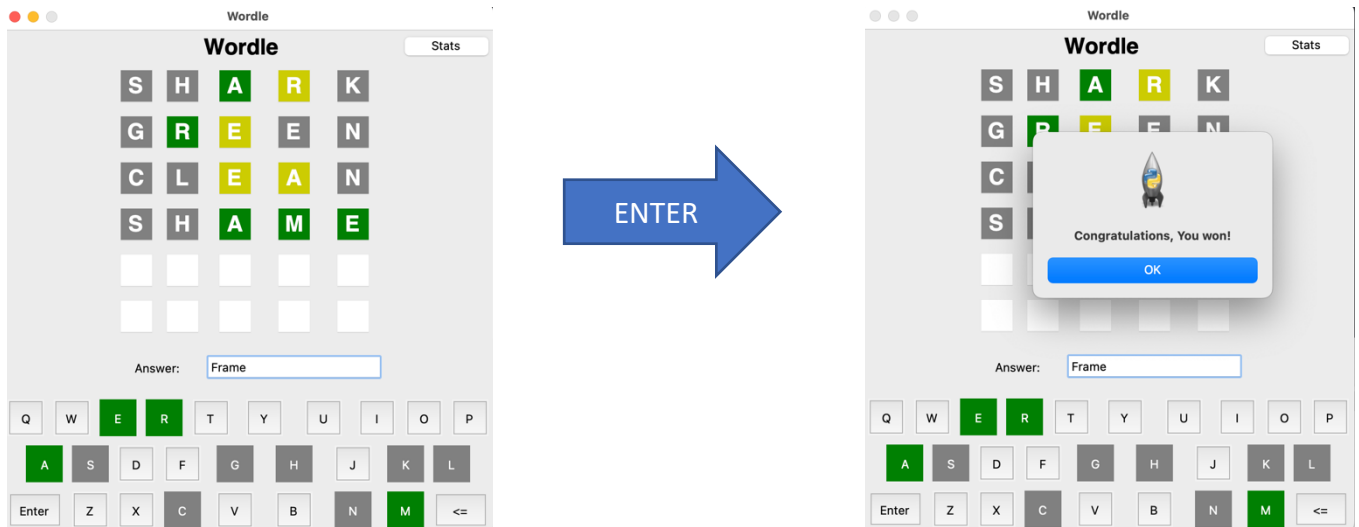
รูป 2.4 การแสดงผล เมื่อกรอกคำศัพท์ที่ไร้ความหมาย และไม่ใช่ภาษาอังกฤษ

5. เมื่อกรอกคำศัพท์ตัวอักษรไม่ถึง 5 และ น้อยกว่า 5 จะขึ้นว่า “word size must be 5!”



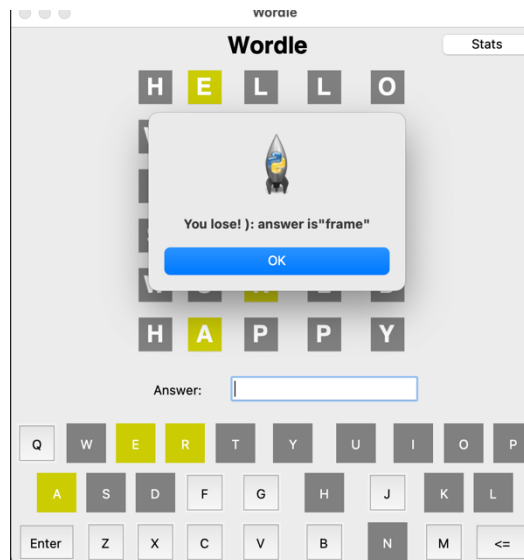
รูป 2.5 การแสดงผล เมื่อกรอกคำศัพท์ที่มีอักขรมากกว่า หรือ น้อยกว่า 5

6. เมื่อชนะ



รูป 2.5 การแสดงผล เมื่อคำศัพท์ที่กรอก ตรงกับเฉลย

7. เมื่อกรอกคำศัพท์ครบ 6 แถวแต่ยังไม่ถูก หรือแพ้



รูป 2.6 การแสดงผล เมื่อแพ้

บทที่ 3

Input and Output

1. Input

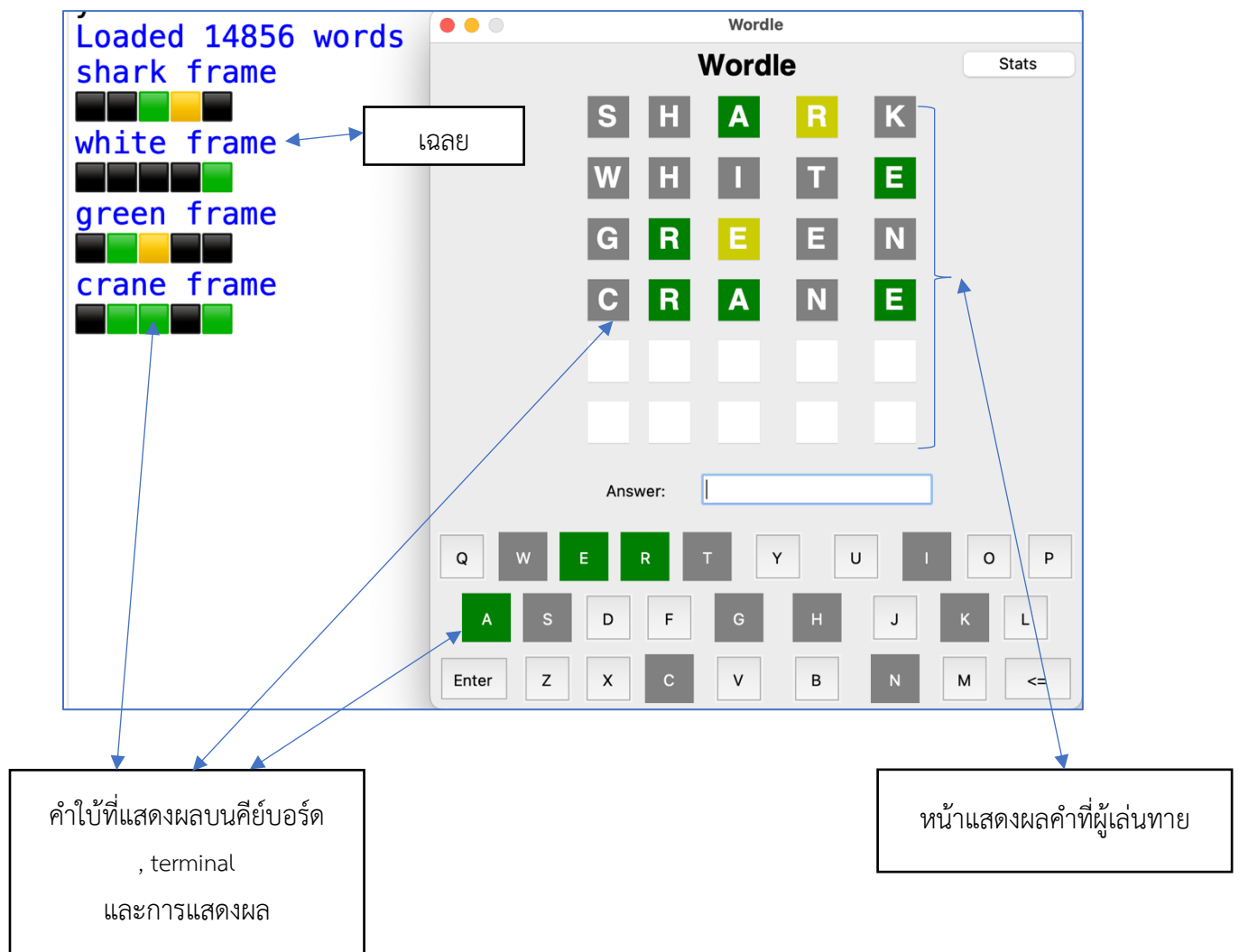
Input จะเป็นคำศัพท์ที่มี 5 ตัวอักษรสามารถส่งได้ 6 ครั้งในแต่ละเกม

2. Output

2.1 หน้าจอแสดงผลคำศัพท์ที่ผู้เล่นทาย

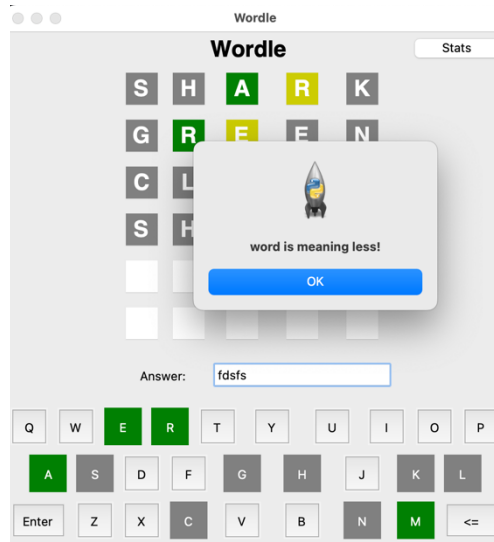
2.2 การแสดงคำใบ้ เหลือง หรือเขียว บนช่องแสดงผล และ คีย์บอร์ด

2.3 เฉลย, คำที่เรากรอก และ คำใบ้ในช่อง Terminal



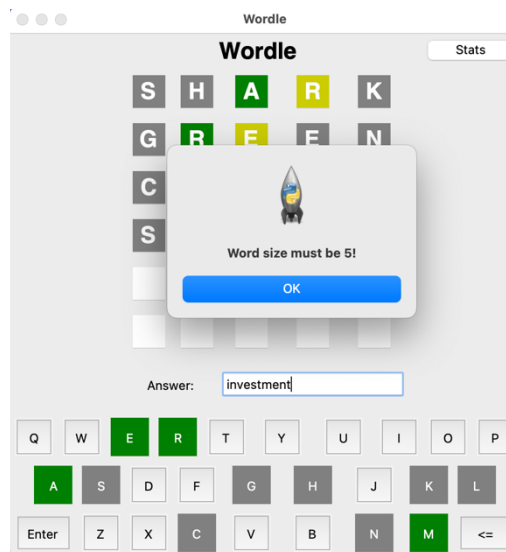
รูป 3.1 การแสดงผล

2.4 หน้าจอเตือน เมื่อกรอกคำศัพท์ที่ไม่มีความหมาย หรือภาษาอื่น



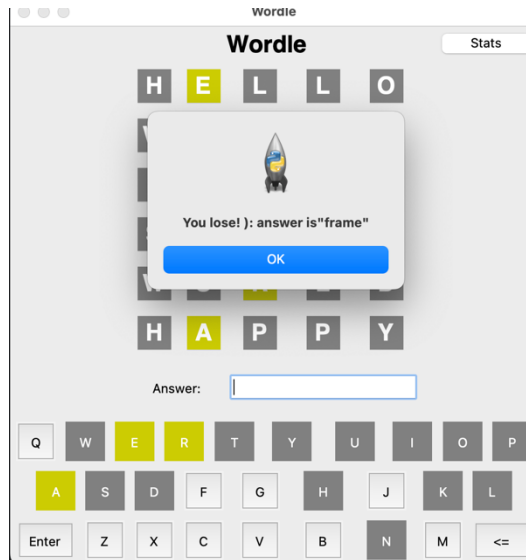
รูป 3.2 กรอกศัพท์ที่ไม่มีความหมาย

2.5 หน้าจอเตือน เมื่อกรอกคำศัพท์อักขรที่มาก หรือน้อยกว่า 5



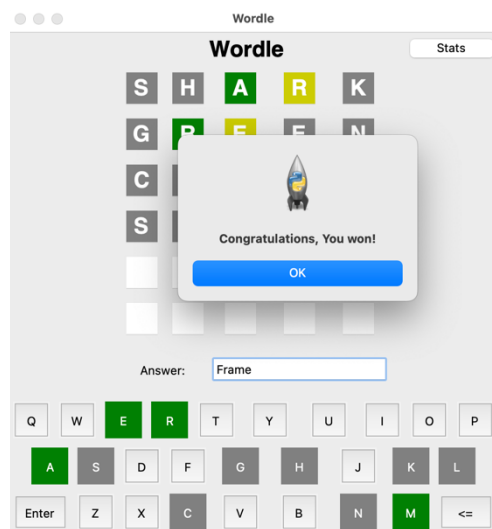
รูป 3.3 กรอกคำศัพท์ตัวอักษรน้อยกว่า หรือมากกว่า 5

2.6 หน้าจอเตือน เมื่อแพ้



รูป 3.4 หน้าจอเตือนว่าแพ้

2.7 หน้าจอเตือน เมื่อชนะ

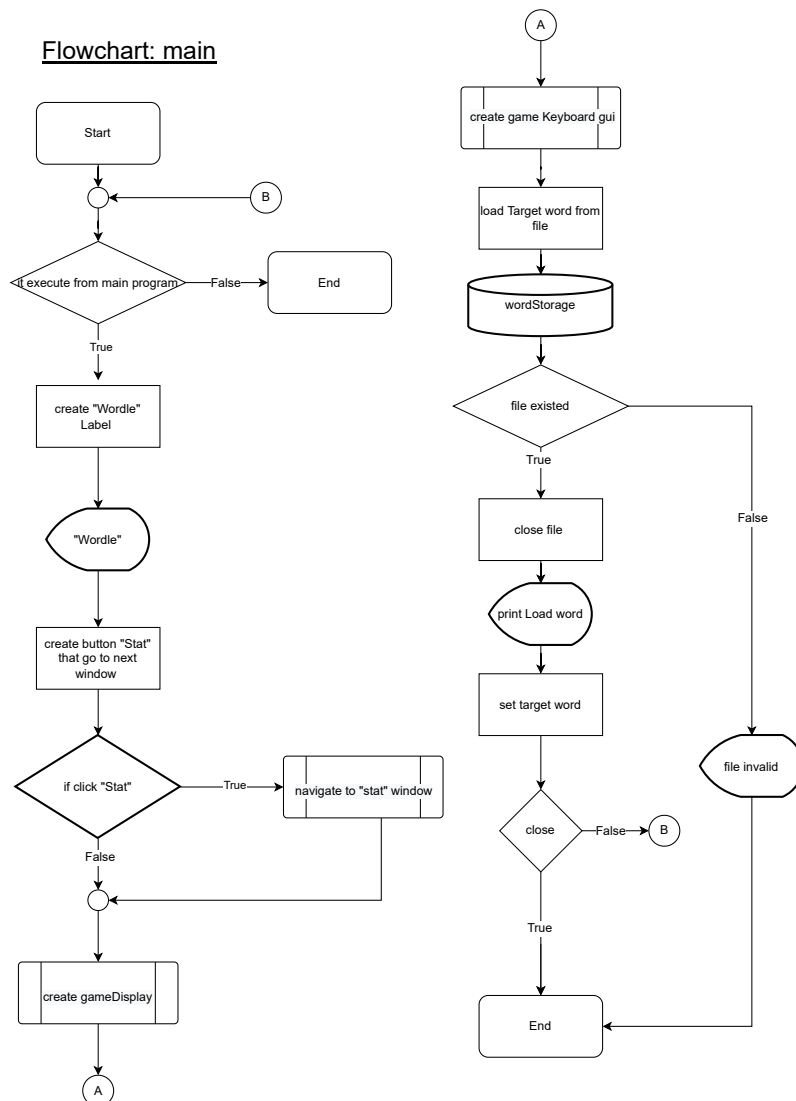


รูป 3.5 หน้าจอเตือนว่าชนะ

บทที่ 4

Flow chart

1. Flowchart : การทำงานรวม

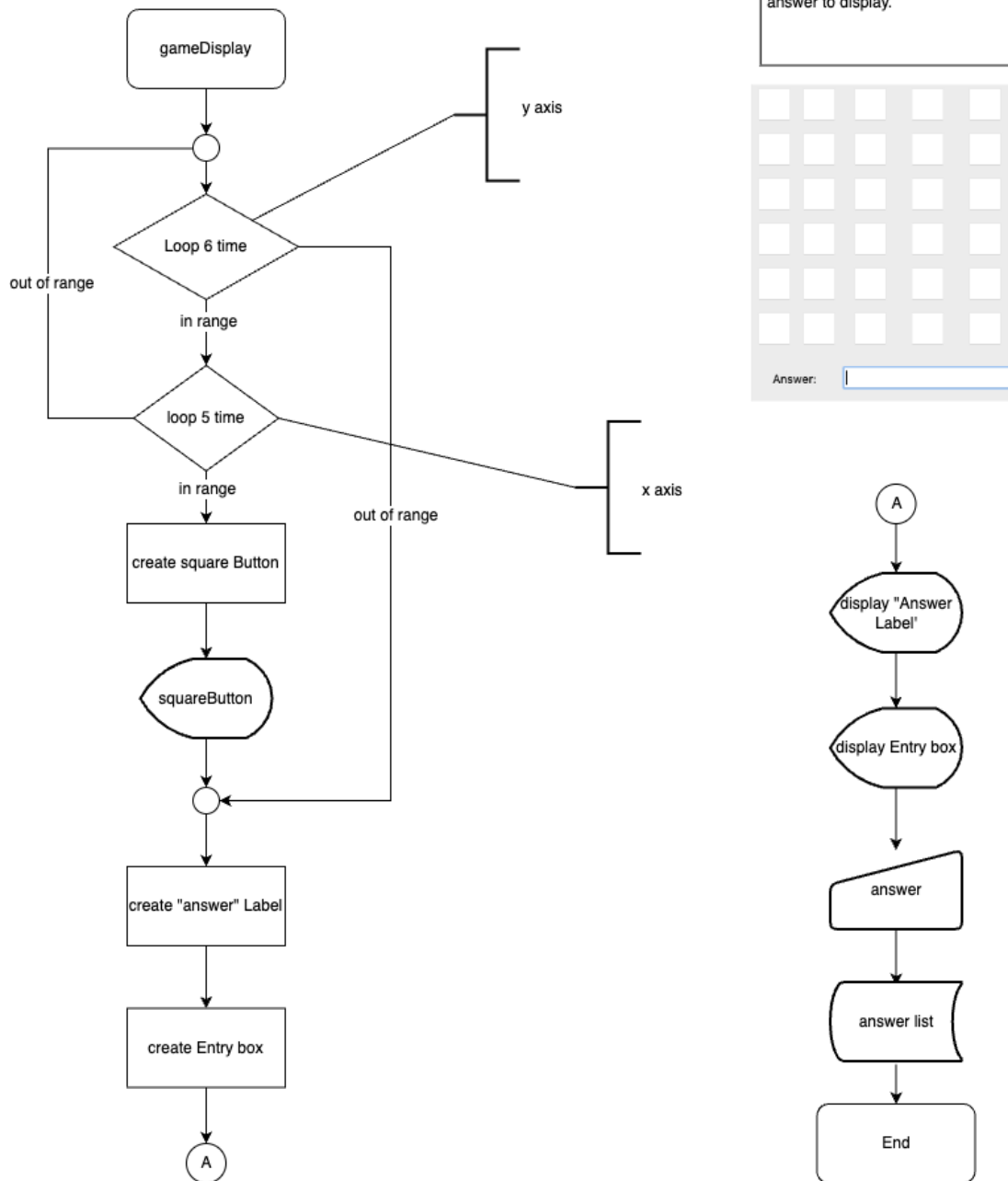


รูปที่ 4.1 Flowchart ของการทำงานรวม

2. Flowchart : การทำงานของส่วนหลัก

2.1 Flowchart : การสร้าง Graphic ส่วนกล่องแสดงผล และกล่องรับคำศัพท์

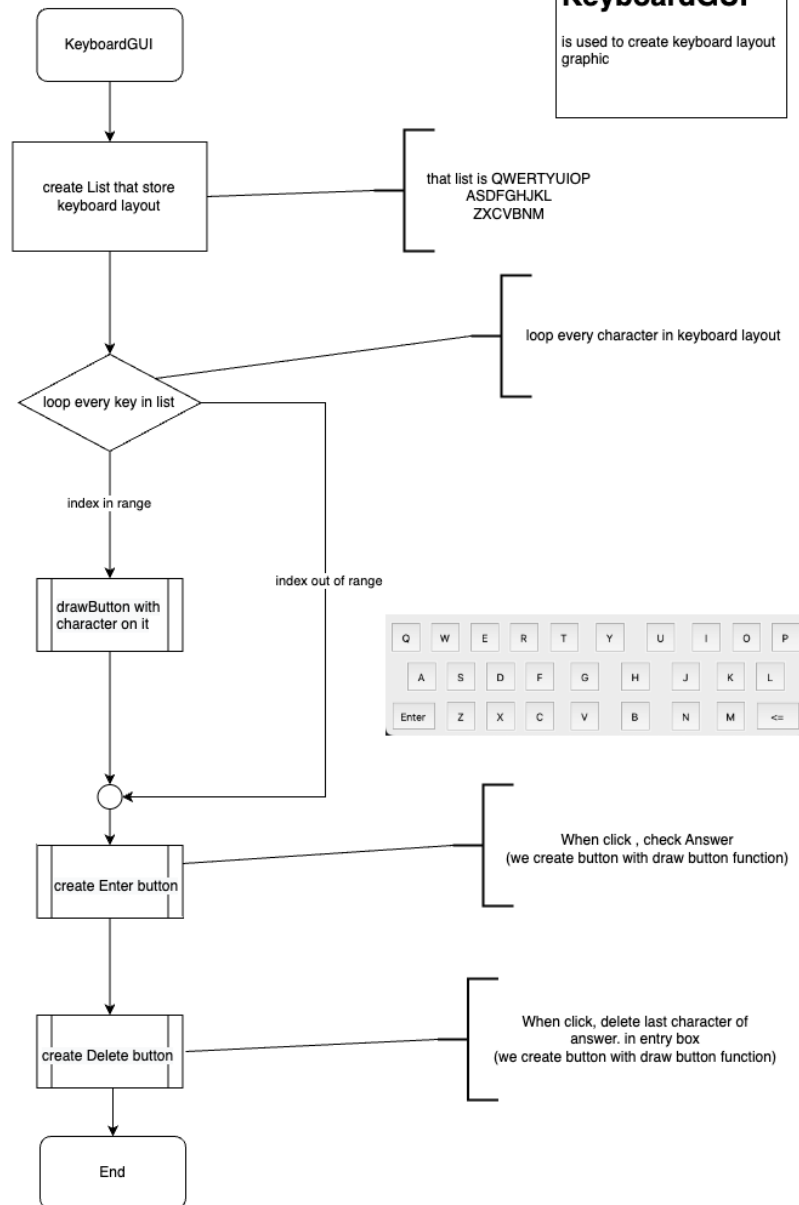
Flowchart: gameDisplay



รูป 4.2 Flow chart ของการสร้าง GUI สำหรับแสดงคำใบ้ และกล่องคำตอบ

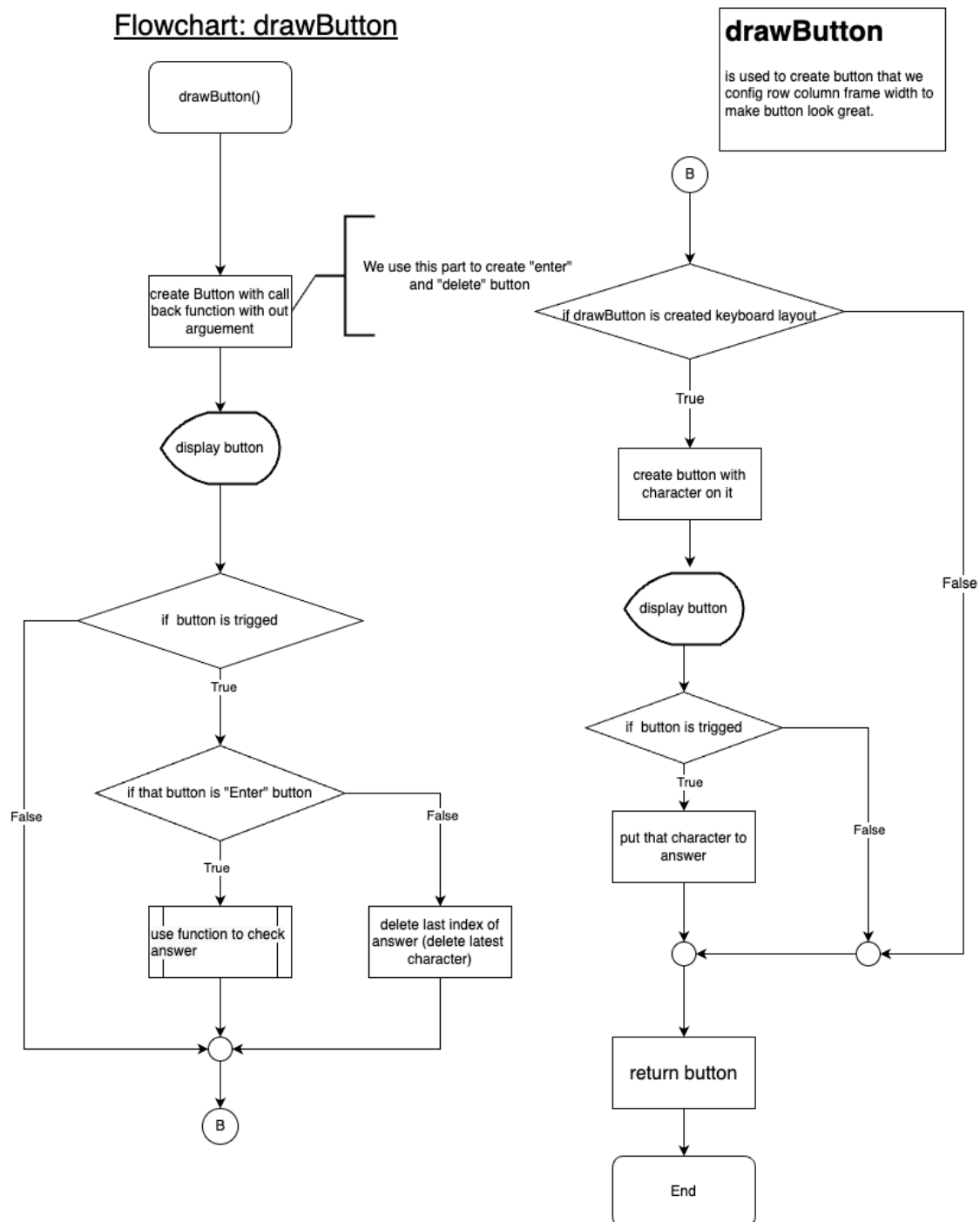
2.2 Flowchart : การสร้าง Graphic ส่วนของคีย์บอร์ด

Flowchart: KeyboardGUI



รูป 4.3 Flowchart ของการสร้าง keyboard graphic

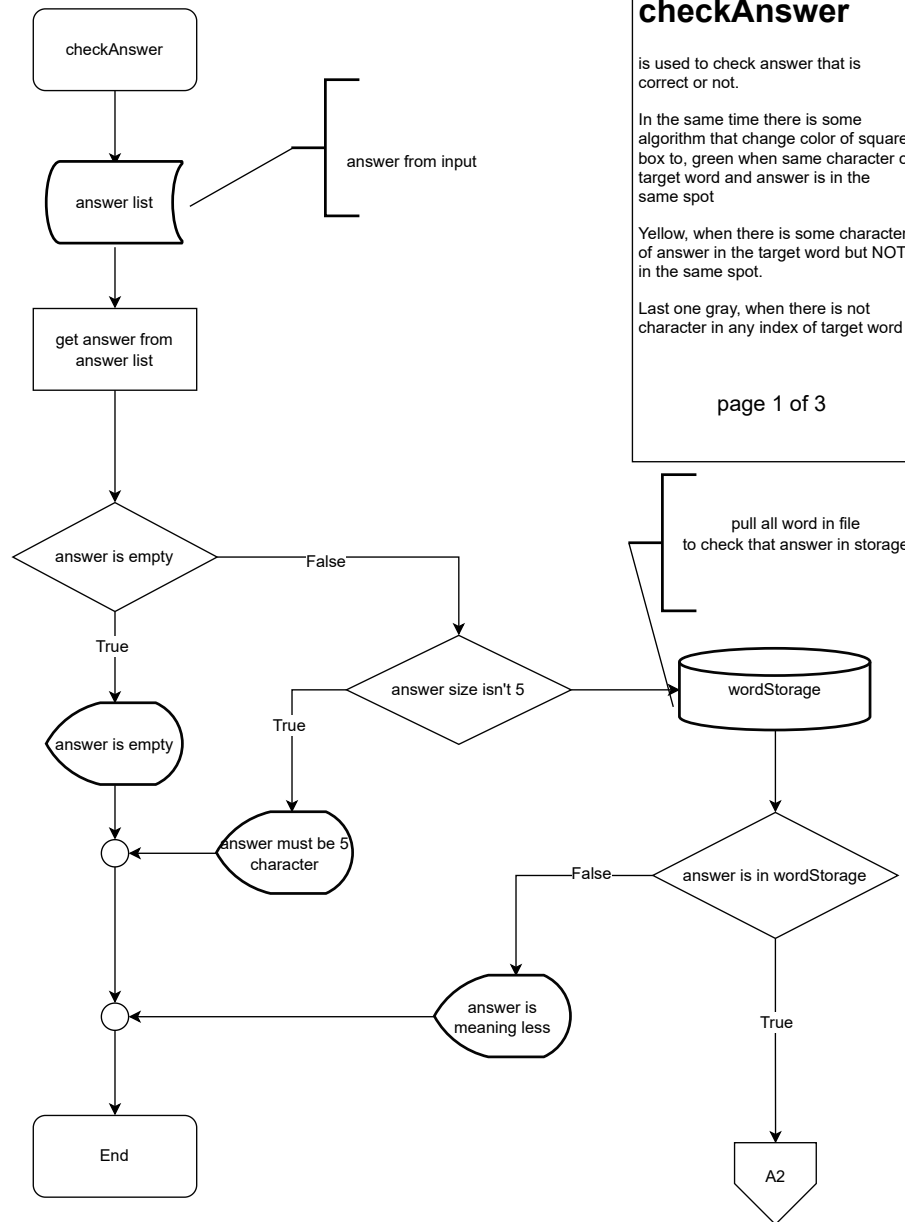
2.3 Flowchart : ฟังก์ชัน drawButton เป็นฟังก์ชันการสร้าง button ที่มีคุณสมบัติที่จะสามารถเปลี่ยนสีคำไปได้ ถูกใช้ใน graphic keyboard เพื่อสร้างปุ่มต่าง ๆ ที่มีตัวอักษรบนนั้น



รูป 4.4 Flowchart drawButton

2.4 Flowchart : ฟังก์ชัน checkAnswer ใช้ในการตรวจสอบคำศัพท์ เปลี่ยนสีคำใบ้ และใช้งานฟังก์ชัน บันทึกข้อมูล ถูกใช้ในฟังก์ชัน drawButton

Flowchart: checkAnswer



Description

checkAnswer

is used to check answer that is correct or not.

In the same time there is some algorithm that change color of square box to, green when same character of target word and answer is in the same spot

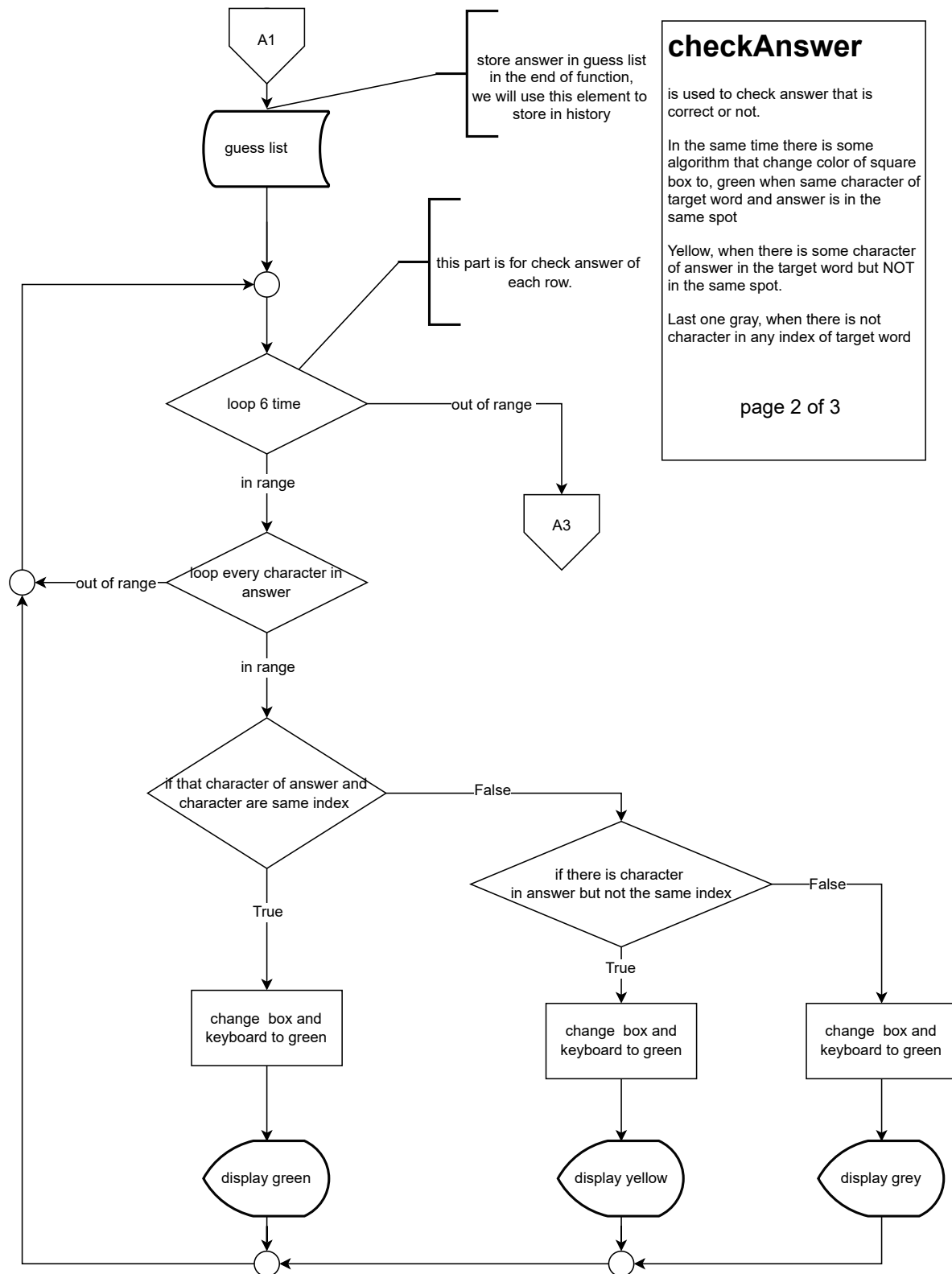
Yellow, when there is some character of answer in the target word but NOT in the same spot.

Last one gray, when there is not character in any index of target word

page 1 of 3

รูป 4.5 Flowchart ฟังก์ชัน checkAnswer page 1

Flowchart: checkAnswer



รูป 4.6 Flowchart ฟังก์ชัน checkAnswer page 2

Flowchart: checkAnswer

Description

checkAnswer

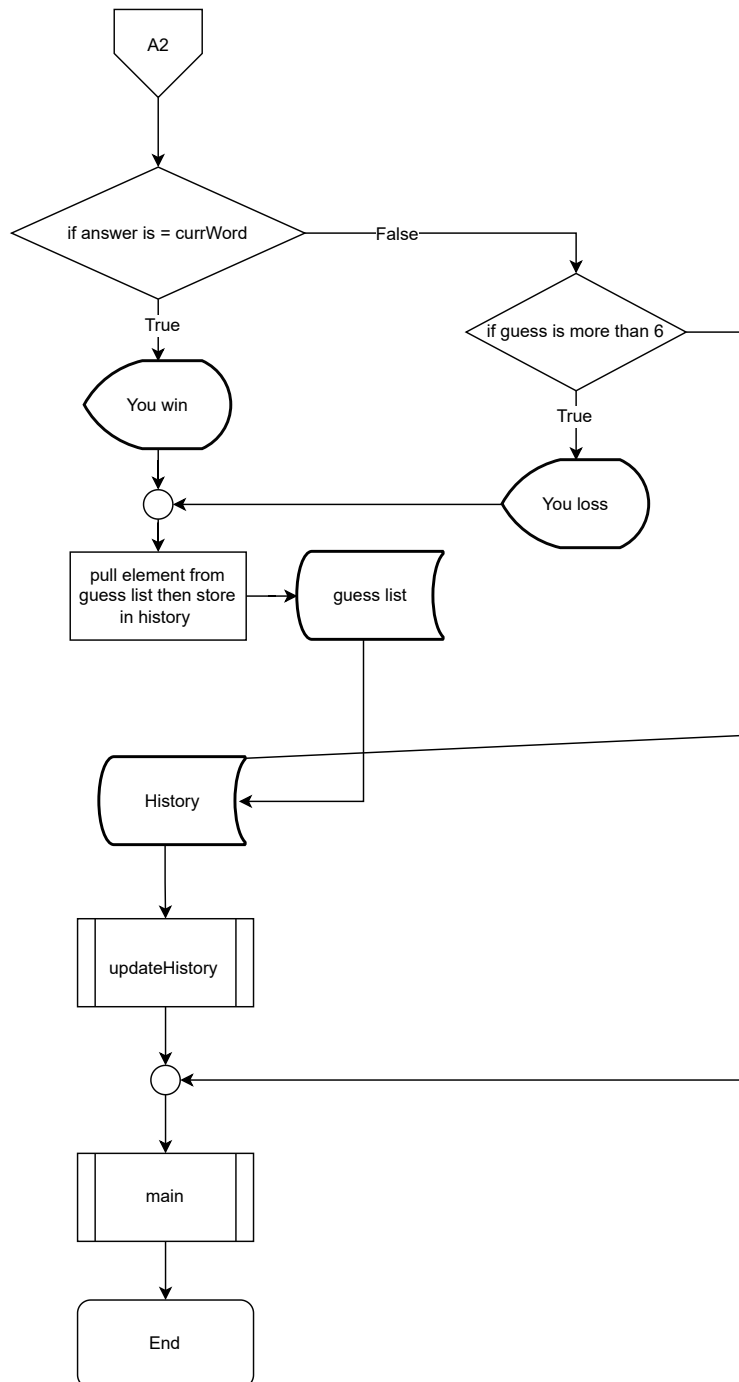
is used to check answer that is correct or not.

In the same time there is some algorithm that change color of square box to, green when same character of target word and answer is in the same spot

Yellow, when there is some character of answer in the target word but NOT in the same spot.

Last one gray, when there is not character in any index of target word

page 3 of 3



store history in data such as,
time targetWord guess word

รูป 4.7 Flowchart ฟังก์ชัน checkAnswer page 3

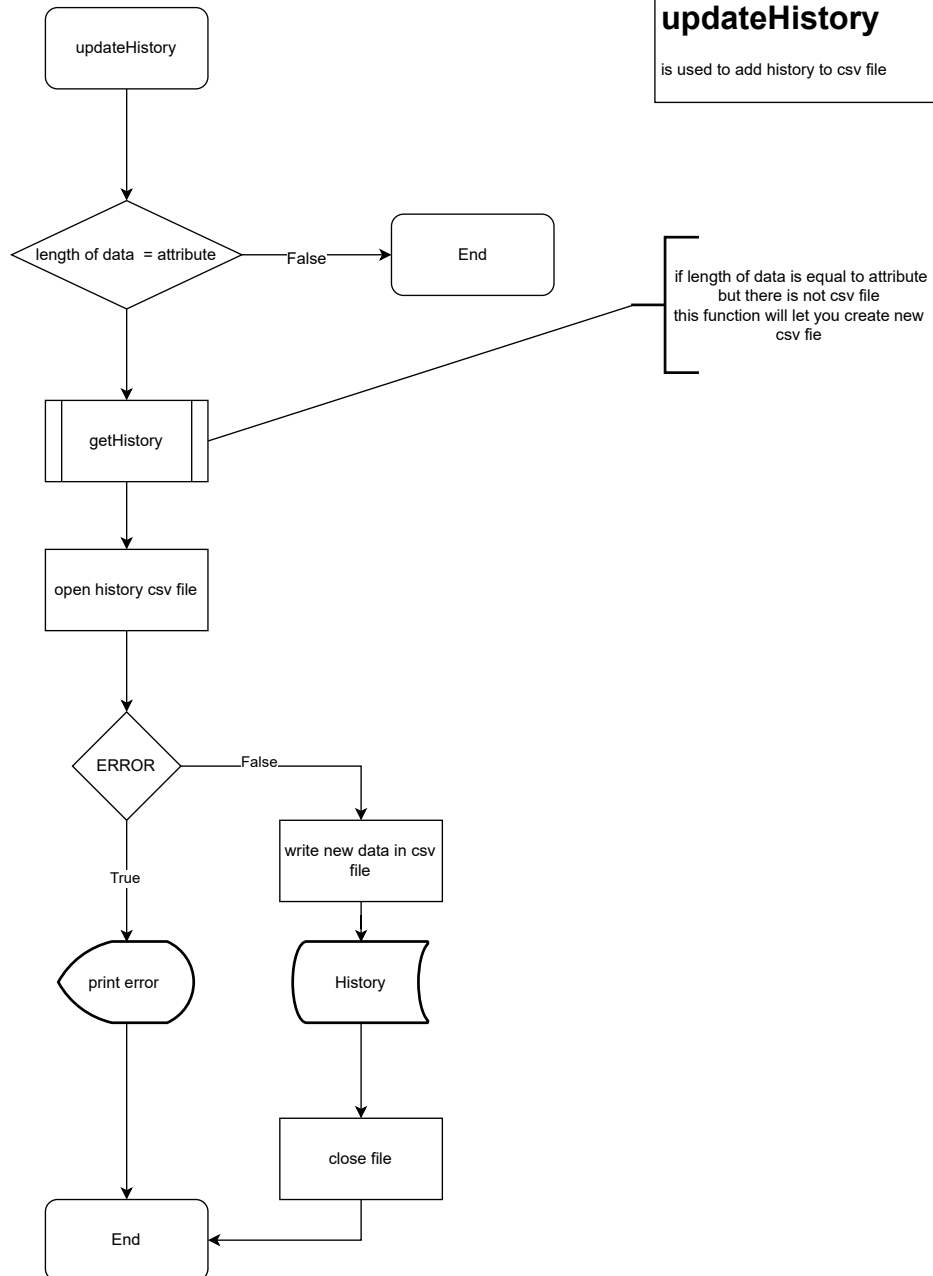
2.5 Flowchart : ฟังก์ชันในการอัปเดตข้อมูลไปยังไฟล์ csv ถูกใช้ใน ฟังก์ชัน check word

Flowchart: updateHistory

Description

updateHistory

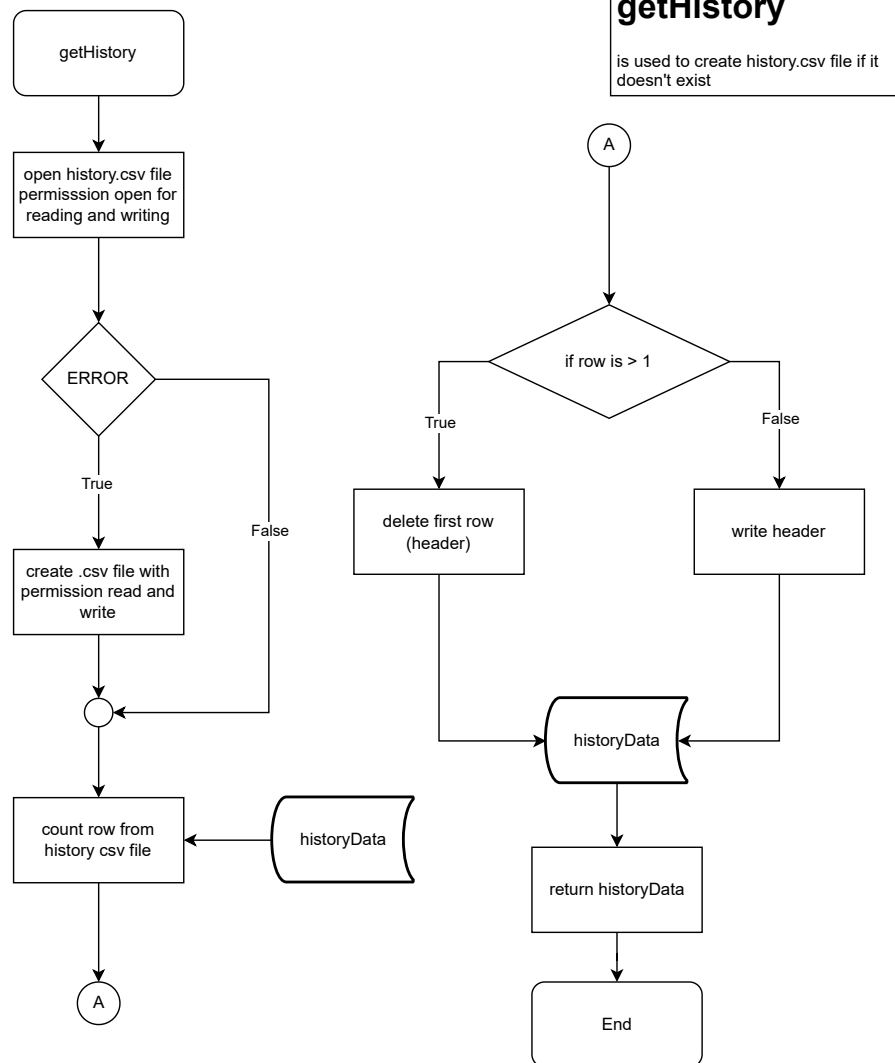
is used to add history to csv file



รูป 4.8 Flowchart ฟังก์ชัน updateHistory

2.6 Flowchart : ฟังก์ชัน getHistory เป็นฟังก์ชันที่เอาไว้สร้างไฟล์ csv หากไม่มี และเพิ่มส่วน header หากไฟล์ csv มีอยู่แล้ว เราจะลบ header ออก

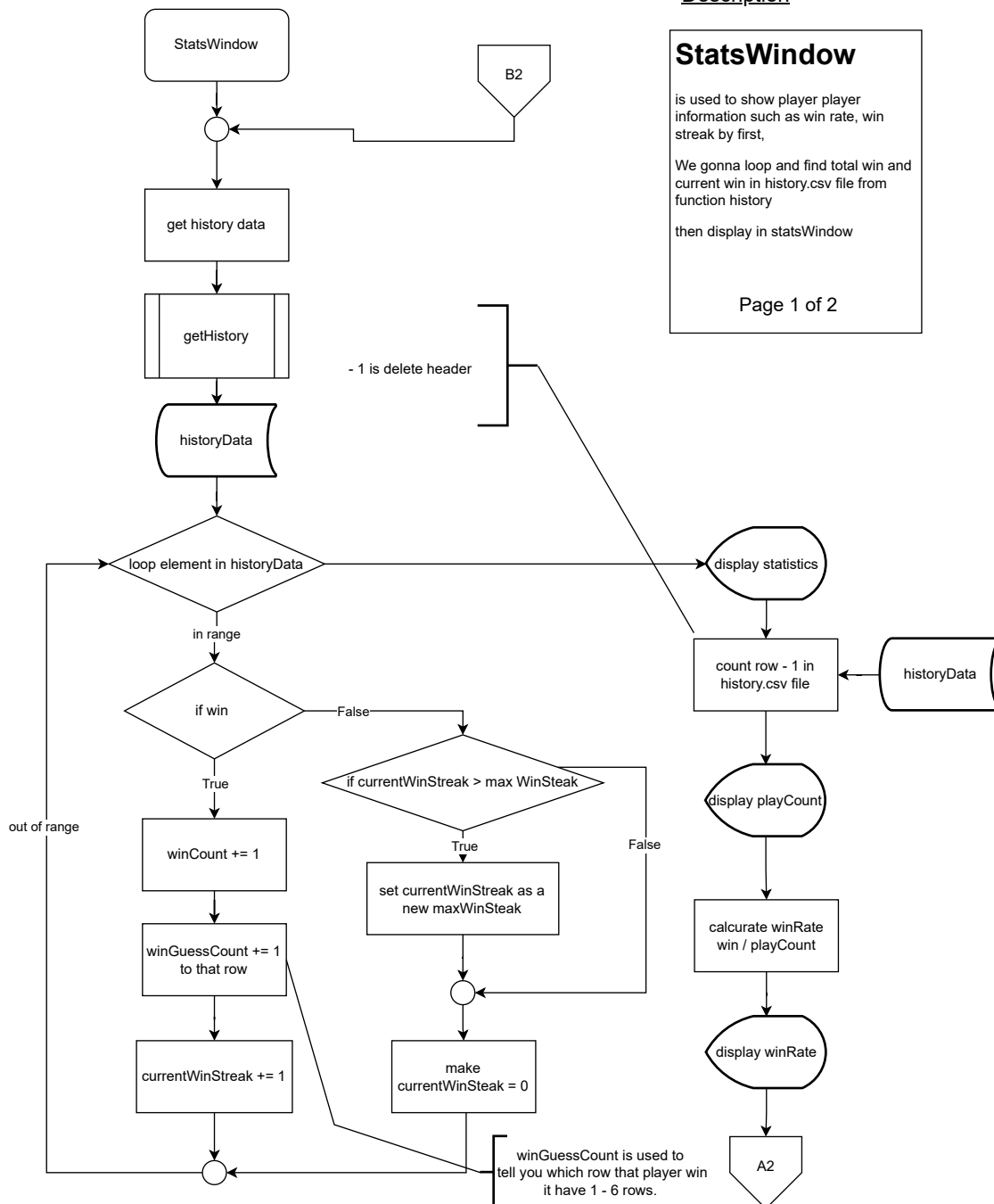
Flowchart: getHistory.



รูป 4.9 Flowchart ฟังก์ชัน getHistory

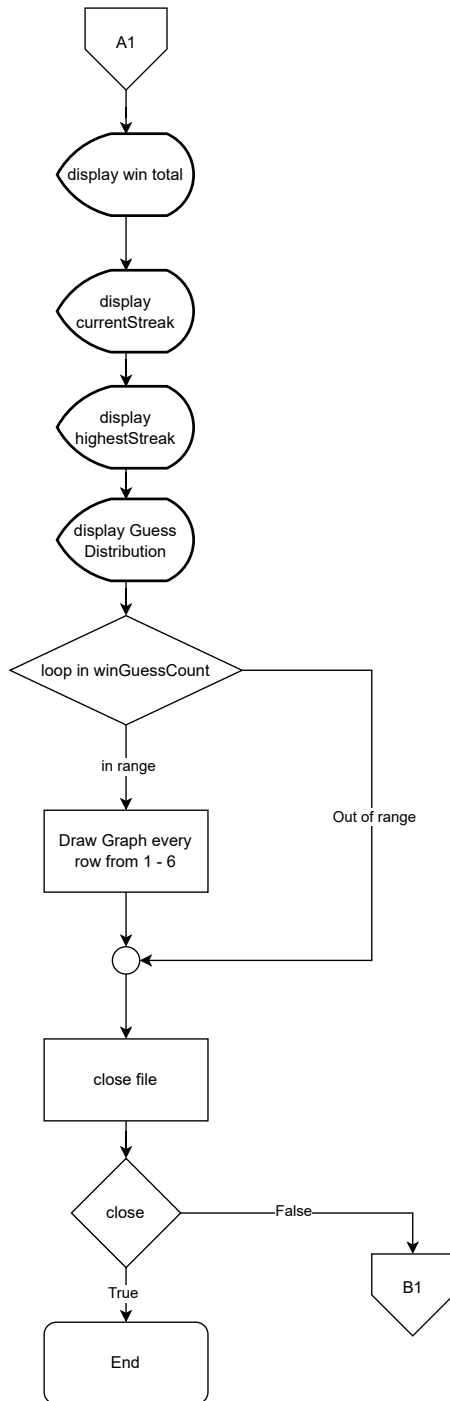
2.7 Flowchart : การสร้าง graphic หน้าต่าง สถิติข้อมูลของผู้เล่น

Flowchart: StatsWindow



รูป 4.10 Flowchart หน้าต่างสถิติ page 1

Flowchart: StatsWindow



Description

StatsWindow

is used to show player player information such as win rate, win streak by first,

We gonna loop and find total win and current win in history.csv file from function history

then display in statsWindow

Page 2 of 2

รูป 4.11 Flowchart หน้าต่างสถิติ page 2

บทที่ 5

Data file

1. ไฟล์ words สำหรับเป็นคลังศัพท์

ComPro1Project > words_			
1	shark		
2	green	14848	femal
3	white	14849	frame
4	world		
5	annoy	14850	abuse
6	space		
7	squad	14851	penny
8	crane		
9	apply	14852	stirp
10	aware	14853	input
11	angel		
12	light	14854	queen
13	speed		
14	apple	14855	change
15	array		
16	polio		

รูป 5.1 ตัวอย่างคำในคลังศัพท์

2. ไฟล์ csv สำหรับเก็บประวัติการเล่น โดยมี header ดังนี้

- 1.1 date : เวลาที่เกมนั้นจบลง (Epoch time)
- 1.2 target_word : คำตอบของเกมนั้น
- 1.3 guess_count : จำนวนที่ผู้เล่นตอบ (-1 ถ้าแพ้)
- 1.4 guess_word1 - guess_word6 : คำที่ผู้เล่นเดา

ComPro1Project > history.csv

```
1 date,target_word,guess_count,guess_word1,guess_word2,guess_word3,guess_word4,guess_word5,guess_word6
2 1665935033.9171999,frame,-1,hello,world,green,white,abuse,fetal
3 1665935210.8469172,adore,2,adorn,adore,,,,
4 1665935274.549983,bloom,-1,carry,catch,world,shark,bunch,blood
5 1665935335.549576,cache,6,input,cliff,candy,cheat,camel,cache
6 1665935378.6963441,email,5,world,erase,equip,embed,email,
7 1665935424.504321,grade,6,white,green,great,gloat,grace,grade
8 1665935509.846447,first,6,white,ghost,green,world,final,first
9 1665935566.3636632,latte,-1,least,leapt,label,layer,lapse
10 |
```

history

date	target_word	guess_count	guess_word1	guess_word2	guess_word3	guess_word4	guess_word5	guess_word6
1665935033.9172000	frame	-1	hello	world	green	white	abuse	fetal
1665935210.8469200	adore	2	adorn	adore				
1665935274.549980	bloom	-1	carry	catch	world	shark	bunch	blood
1665935335.549580	cache	6	input	cliff	candy	cheat	camel	cache
1665935378.6963400	email	5	world	erase	equip	embed	email	
1665935424.504320	grade	6	white	green	great	gloat	grace	grade
1665935509.846450	first	6	white	ghost	green	world	final	first
1665935566.3636600	latte	-1	least	leapt	label	later	layer	lapse

รูป 5.2 ตัวอย่างข้อมูลสถิติจากไฟล์ history.csv

บทที่ 6

1. **entryList**: ไว้เก็บ Entry ที่เป็นตารางการเล่น (5 x 6) สำหรับแก้ไขสี

ข้อมูลตัวอย่าง: entryList = [
 [<tkinter.Entry object .!frame29.!entry>, ...], # ขนาด 5
 [<tkinter.Entry object .!frame29.!entry>, ...], # ขนาด 5
 ...
] # ขนาด 6

2. **buttonList**: ไว้เก็บ Button ที่เป็นคีย์บอร์ดสำหรับแก้ไขสี (เก็บตามตัวอักษร)

ข้อมูลตัวอย่าง: buttonList = {
 'q': <tkinter.Button object .!frame.!button2>,
 'w': <tkinter.Button object .!frame2.!button2>,
 ...
}

3. **textVariableList**: เหมือนกับ entryList แต่เก็บ textVariable แทนไว้ใช้เปลี่ยนค่าที่แสดงอยู่

4. **wordsList**: ไว้เก็บคำศัพท์ของ Wordle ทั้งหมด (เปิดมาจากไฟล์)

5. **guessList**: ไว้เก็บคำที่ผู้เล่นเดา ในเกมนั้นๆ (รีเซ็ตทุกๆ เกม)

ข้อมูลตัวอย่าง: guessList = ['crane', 'plaza', 'quash', 'staff', 'toast', 'staid']

6. **targetWordCount**: จำนวนของแต่ละตัวอักษรของคำตอบ (รีเซ็ตทุกๆ เกม)

ข้อมูลตัวอย่าง: ถ้าคำตอบ = 'fells', targetWordCount = {
 'f': 1, 'e': 1, 'l': 2, 's': 1
}

7. **currWordState**: ไว้เก็บข้อมูล (ว่าเป็นสีเหลือง หรือเขียว) ของคำที่ผู้เล่นเดา (รีเซ็ตทุกๆ ครั้งที่ผู้เล่นพิมพ์)

ข้อมูลตัวอย่าง: ถ้าคำตอบ = 'wonts', ผู้เล่นเดาว่า 'pains', จะได้ currWordState = {
 0: {'char': 'p', 'color': 'gray'},
 1: {'char': 'a', 'color': 'gray'},
 2: {'char': 'l', 'color': 'gray'},
 3: {'char': 'n', 'color': 'yellow'},
 4: {'char': 's', 'color': 'green'}
}

บทที่ 7

Code

```
from math import floor
from tkinter import *
from tkinter import messagebox

import random
import time
import csv

# Global Game Variable
entryList = []           # Store all Entry
buttonList = {}          # Store all Keyboard's Button
textVariableList = []    # Store all Entry's TextVariable
wordsList = []           # Wordle's words list
answerEntry = None       # Answer Entry box
answerVariable = None     # Answer Entry box's TextVariable
guessList = []           # Current Game's guess words
targetWord = ''          # Current Game's target word
currRow = 0              # Current Game's playing row

# Constants
HISTORY_HEADER = [
    'date',
    'target_word',
    'guess_count',
    'guess_word1',
    'guess_word2',
    'guess_word3',
    'guess_word4',
    'guess_word5',
    'guess_word6'
]

def getHistory():
    try:
        f = open('history.csv', 'r+', newline='')
    except:
        try:
            # File does not existed
            f = open('history.csv', 'w+', newline='')
        except:
            print('Error can\'t create history.csv File, getHistory()')
```

```

        return []

    csvReader = csv.reader(f)
    historyData = [row for row in csvReader]

    # File existed, but it's empty
    if len(historyData) == 0:
        csvWriter = csv.writer(f)
        csvWriter.writerow(HISTORY_HEADER)
    else:
        # Removed first row (header row)
        historyData = historyData[1:]

    f.close()
    return historyData

def updateHistory(data):
    if(len(data) != len(HISTORY_HEADER)):
        print('Invalid History data!, (updateHistory(data))')
        return False

    # Call for header checking ...
    getHistory()

    try:
        f = open('history.csv', 'a', newline='')
    except:
        print('Error can\'t open history.csv File')
        return False

    csvWriter = csv.writer(f)
    csvWriter.writerow(data)
    f.close()

    return True

def drawButton(text='', row=0, rowspan=1, column=0, columnspan=1, width=100,
height=100, command=None, keyboard=False):
    frame = Frame(root, width=width, height=height)
    button = Button(frame, text=text, command=command)
    if(keyboard):
        button = Button(frame, text=text,
                        command=lambda: onKeyboardClick(text))

    frame.grid_propagate(False)          # Disables resizing of frame
    frame.columnconfigure(0, weight=1)   # Enables button to fill frame

```

```

frame.rowconfigure(0, weight=1)

frame.grid(row=row, column=column, rowspan=rowspan,
           columnspan=columnspan, padx=5, pady=5)
button.grid(sticky='wens')
return button

def drawSquareEntry(textvariable, row=0, rowspan=1, column=0, columnspan=1, width=100,
height=100):
    frame = Frame(root, width=width, height=height)
    entry = Entry(frame, textvariable=textvariable, justify='center',
                  foreground='white', font='Helvetica 24 bold')

    frame.grid_propagate(False)          # Disables resizing of frame
    frame.columnconfigure(0, weight=1)    # Enables button to fill frame
    frame.rowconfigure(0, weight=1)

    frame.grid(row=row, column=column, rowspan=rowspan,
              columnspan=columnspan, padx=5, pady=5)
    entry.grid(sticky='wens')
    return entry

def initKeyboardGUI():
    keyboardLayout = [
        'QWERTYUIOP',
        'ASDFGHJKL',
        'ZXCVBNM'
    ]

    offset = [0, 1, 3]
    startRow, startColumn = 19, 1

    # Draw Keyboard Key
    for inxRow, row in enumerate(keyboardLayout):
        placeRow = startRow + (2 * inxRow)
        for inxCol, text in enumerate(list(row)):
            placeColumn = startColumn + (2 * inxCol) + offset[inxRow]

            btn = drawButton(text, width=40, height=40,
                            row=placeRow, rowspan=2,
                            column=placeColumn, columnspan=2, keyboard=True
                            )
            buttonList[text.lower()] = btn

    # Enter Button
    drawButton('Enter', row=23, rowspan=2, column=1,

```

```

        colspan=3, width=40 / 2 * 3, height=40, command=checkWord)

# Return Button
drawButton('<=', row=23, rowspan=2, column=18,
          colspan=3, width=40 / 2 * 3, height=40, command=onReturn)

def initDisplay():
    startRow, startColumn = 4, 6

    # Display 6 x 5
    for inxRow in range(6):
        #* y axis
        placeRow = startRow + (2 * inxRow) #* 4 6 8 10 12 14
        textVariableRow = []
        entryRow = []
        for inxCol in range(5):
            #* x axis
            placeColumn = startColumn + (2 * inxCol) #* 6 8 10 12

            str = StringVar()
            textVariableRow.append(str)

            entry = drawSquareEntry(str, width=40, height=40,
                                    row=placeRow, rowspan=2,
                                    column=placeColumn, colspan=2
                                    )

            entry['state'] = DISABLED
            entry['disabledbackground'] = 'white'
            entry['disabledforeground'] = 'white'
            entryRow.append(entry)
            textVariableList.append(textVariableRow)

        entryList.append(entryRow)

# Answer Box
Label(root, text='Answer: ').grid(
    row=17, column=6, colspan=4, pady=15)

global answerVariable
answerVariable = StringVar()

entryAnswer = Entry(root, textvariable=answerVariable)
entryAnswer.grid(row=17, column=10, colspan=6, pady=15)
entryAnswer.bind('<Return>', checkWord)
entryAnswer.focus()

global answerEntry
answerEntry = entryAnswer

```

```

def onReturn():
    currWord = answerVariable.get()
    currWord = currWord[:-1] # Remove last element

    answerVariable.set(currWord)
    answerEntry.icursor(len(currWord))

def onKeyboardClick(key):
    currWord = answerVariable.get()
    currWord += key.lower()

    answerVariable.set(currWord)
    answerEntry.icursor(len(currWord))

def checkWord(event=None):
    currWord = answerVariable.get().strip().lower()

    # Is word empty
    if(len(currWord) == 0):
        messagebox.showinfo('Please enter again', 'Word can\'t be emptied!')
        return

    # Is word wrong size
    if(len(currWord) != 5):
        messagebox.showinfo('Please enter again', 'Word size must be 5!')
        return

    # Is word a word
    if(currWord not in wordsList):
        messagebox.showinfo('Please enter again', 'Word is meaningless!')
        return

    print(currWord, targetWord)
    guessList.append(currWord)

    # Create dict of each letter count of Target Word
    targetWordCount = {}
    for c in targetWord:
        if(c in targetWordCount):
            targetWordCount[c] += 1
        else:
            targetWordCount[c] = 1

    currWordState = {}
    # Check for exact match
    for idx, char in enumerate(currWord):

```

```

if(char == targetWord[idx]):
    # Remove Exact Match from Target Word's letter count
    targetWordCount[char] -= 1

    # Exact Match, green color
    currWordState[idx] = {
        'char': char,
        'color': 'green'
    }
else:
    # Not Exact Match, can be yellow, or gray
    currWordState[idx] = {
        'char': char,
        'color': 'gray'
    }

for idx, char in enumerate(currWord):
    # Is there is any char in Target Word
    if(char in targetWord):
        if(targetWordCount[char] != 0):
            # If not Exact Match but exist in word, yellow color
            if(currWordState[idx]['color'] != 'green'):
                currWordState[idx]['color'] = 'yellow'

            targetWordCount[char] -= 1
            # No more words left, gray color
            elif(targetWordCount[char] < 1):
                currWordState[idx]['color'] = 'gray'

global currRow
# Set Color, and Char
for idx in currWordState:
    color = currWordState[idx]['color']
    if(color == 'green'):
        print('■', end='')
    elif(color == 'yellow'):
        print('■', end='')
    elif(color == 'gray'):
        print('■', end='')

    # Special Yellow Color
    if(color == 'yellow'):
        color = '#CCCC00'

    entryList[currRow][idx]['disabledbackground'] = color
    buttonList[currWordState[idx]['char']]['background'] = color

# For Mac OS

```

```

        buttonList[currWordState[idx]['char']]['highlightbackground'] = color
        buttonList[currWordState[idx]['char']]['highlightthickness'] = 30

        textVariableList[currRow][idx].set(currWordState[idx]['char'].upper())
        entryList[currRow][idx]['state'] = DISABLED

        buttonList[currWordState[idx]['char']]['foreground'] = 'white'
    print('')

    answerVariable.set('')
    currRow += 1

    # For Mac OS, manually update the windows
    root.update_idletasks()
    root.update()

    # Answer is correct
    if(currWord == targetWord):
        messagebox.showinfo('You won!', 'Congratulations, You won!')
        history = [
            time.time(), # date
            targetWord, # target_word
            currRow, # guess_count
        ]

        # guess_word1 - guess_word6
        for i in range(6):
            if(i >= len(guessList)):
                history.append('')
            else:
                history.append(guessList[i])

        updateHistory(history)
        gameCycle()
        return

    # You lose ):
    if(currRow == 6):
        messagebox.showinfo('You lose!', 'You lose! ):\nAnswer was "' +
targetWord.title() + '"')
        history = [
            time.time(), # date
            targetWord, # target_word
            -1, # guess_count
            guessList[0], # guess_word1
            guessList[1], # guess_word2
            guessList[2], # guess_word3
            guessList[3], # guess_word4

```

```

        guessList[4], # guess_word5
        guessList[5], # guess_word6
    ]

    updateHistory(history)
    gameCycle()
    return

def gameCycle():
    # Pick random words
    global targetWord
    targetWord = random.choice(wordsList)

    global guessList
    guessList = []

    # Reset Counter
    global currRow
    currRow = 0
    for idxRow, row in enumerate(textVariableList):
        for idxCol, textVar in enumerate(row):
            textVar.set('')

            entryList[idxRow][idxCol]['disabledbackground'] = 'white'

    for btn in buttonList:
        # Default Button Color
        buttonList[btn]['background'] = 'SystemButtonFace'
        buttonList[btn]['foreground'] = 'black'

        # For Mac OS
        buttonList[btn]['highlightbackground'] = 'SystemButtonFace'
        buttonList[btn]['highlightthickness'] = 0

def StatsWindow():
    root2 = Tk()
    root2.title('Stats & History | Wordle')

    # Make root2 (Stats Windows) not resizable
    root2.resizable(False, False)

    historyData = getHistory()

    # One loop calculate all
    winCount = 0
    winGuessCount = {

```



```

        '1': 0,
        '2': 0,
        '3': 0,
        '4': 0,
        '5': 0,
        '6': 0
    }
    highestStreak = 0
    currStreak = 0
    for game in historyData:
        guessCount = int(game[2])

        # Winning
        if(guessCount != -1):
            winGuessCount[str(guessCount)] += 1
            winCount += 1

            currStreak += 1
        else: # Losing
            if(currStreak > highestStreak):
                highestStreak = currStreak

            currStreak = 0

#Draw 'STATISTICS' Label
Label(root2, text='STATISTICS', font='Helvetica 10 bold').grid(row=0, column=0,
columnspan=5, pady=5)

# Draw Play count
playCount = len(historyData)
Label(root2, text=playCount).grid(row=1, column=0)
Label(root2, text='Played').grid(row=2, column=0)

# Draw Win rate
winRate = winCount / playCount
Label(root2, text=str(round(winRate * 100))).grid(row=1, column=1)
Label(root2, text='Win %').grid(row=2, column=1)

# Draw Total Win
Label(root2, text=str(winCount)).grid(row=1, column=2)
Label(root2, text='Total Win').grid(row=2, column=2)

# Draw Current Streak
Label(root2, text=str(currStreak)).grid(row=1, column=3)
Label(root2, text='Current Streak').grid(row=2, column=3)

# Draw Max Streak
Label(root2, text=str(highestStreak)).grid(row=1, column=4)

```

```

Label(root2, text='Max Streak').grid(row=2, column=4)

# Draw 'GUESS DISTRIBUTION' Label
Label(root2, text='GUESS DISTRIBUTION', font='Helvetica 10 bold').grid(
    row=3, column=0, columnspan=5, pady=5)
maxLength = max([winGuessCount[key] for key in winGuessCount])
charType, charMaxSize = '|', 50
for i in winGuessCount:
    graphBar = charType * floor(charMaxSize * winGuessCount[i] / maxLength)
    idx = int(i) - 1

    # Guess Count
    Label(root2, text=i).grid(row=4+idx, column=0)
    # Bar Graph
    Label(root2, text=f'{graphBar} {winGuessCount[i]}', fg='green').grid(
        row=4+idx, column=1, columnspan=3, sticky='W')

f.close()
root2.mainloop()

if __name__ == '__main__':
    root = Tk()
    root.title('Wordle')

    # Make root not resizable
    root.resizable(False, False)

    root.rowconfigure(tuple(range(26)), weight=1, minsize=1)
    root.columnconfigure(tuple(range(22)), weight=1, minsize=1)

    # Draw Title
    Label(root, text='Wordle', font='Helvetica 24 bold').grid(
        row=1, column=8, rowspan=2, columnspan=6)

    # Stats Button
    Button(root, text='Stats', width=7, command=StatsWindow).grid(
        row=1, column=17, columnspan=4)

    # Draw Components
    initKeyboardGUI()
    initDisplay()

    # Load Words List
    try:
        f = open('words', 'r')
        wordsList = f.read().split('\n')
        f.close()

```

```
except:
    print('Can\'t Find Words list File, exiting...')
    exit()

print(f'Loaded {len(wordsList)} words')

# Main Game Cycle
gameCycle()

root.mainloop()
```