2019 Spring ICS Final Project Documentation

Yukun Jiang, Yue Shu

Original files we edited:

1. chat_client_class.py
2. chat_cmdl_client.py
3. chat_server.py
4. client_state_machine.py

Extra files and things we add:

1. interface.py
2. sign_up.py
3. k_shift_encryption.py
4. NYUSH.gif
5. snakegame.py
6. account_info.txt
7. temporary_login_name.txt
8. Wrong_Times.txt

Where we edited?

1. in chat_client_class.py:

```
class Client:
    def __init__(self, args,username):
        self.peer = ''
        self.console_input = []
        self.state = S_OFFLINE
        self.system_msg = ''
        self.local_msg = ''
        self.peer_msg = ''
        self.args = args
        self.username = username
        self.interface = interface.INTERFACE(self.username)
        self.interface.root.withdraw()
        self.chat_history = ''
```

New attributes: self.username / self.interface / self.chat_history

```
def output(self):
  if len(self.system_msg) > 0:
    # print(self.system_msg)
     #GUI.output=self.system_msg
     self.interface.get_message_print('\n'+self.system_msg)
     self.system_msg = ''
```

The way we output message on the screen. Instead of printing it on the terminal, I implement the get_message_print function to print.

```
def read_input(self):
  while True:
    #text = sys.stdin.readline()[:-1]
    if len( self.interface.text_out) > 0:
      text=self.interface.text_out[0]
      del self.interface.text_out[0]
      self.system_msg += text
      self.console_input.append(text) # no need for lock, append is thread safe
```

Instead of reading input from Terminal, we read it from the text_out box, if length of it is >=1, get it and clear the box. Otherwise do nothing, since the background is always refreshing.

```
    def run_chat(self):
        self.init_chat()
        self.interface.root.deiconify()
        self.system_msg += '\nWelcome to ICS GUI interface! '

        self.output()
        while self.login() != True:
            self.output()
        self.system_msg += 'Welcome, ' + self.get_name() + "!"
        self.output()
        while self.sm.get_state() != S_OFFLINE:
            self.proc()
            self.output()
            self.interface.root.update()
            time.sleep(CHAT_WAIT)
            # GUI update: root.update()
        self.quit()
        self.interface.root.quit()
        self.interface.root.destroy()
```

We add one line to keep updating the interface window to be consistent with the terminal.

2. chat_cmdl_client.py

```
def main():
    import argparse
    parser = argparse.ArgumentParser(description='chat client argument')
    parser.add_argument('-d', type=str, default=None, help='server IP addr')
    args = parser.parse_args()
    login_in()
    fileobject = open('temporary_login_name.txt','r')
    entrancename = fileobject.readline().strip()
    fileobject.close()
    client = Client(args,entrancename)
    client.run_chat()

main()
```

When run the program, you will first jump into “login_in()”, where you need to sign up or login in with your account. Then your “username” will be temporary stored in the file, and we will read it and use it as your username when you enter the interface.

3. chat_server.py

```
class Server:
    def __init__(self):
        self.new_clients = []  # list of new sockets of which the user id is not known
        self.logged_name2sock = {}  # dictionary mapping username to socket
        self.logged_sock2name = {}  # dict mapping socket to user name
        self.all_sockets = []
        self.group = grp.Group()
        # start server
        self.server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server.bind(SERVER)
        self.server.listen(5)
        self.all_sockets.append(self.server)
        # initialize past chat indices
        self.indices = {}
        # sonnet
        self.sonnet = indexer.PIndex("AllSonnets.txt")
        # innitialize three variables for the gaming process
        self.score = {}
        self.player = []
        self.endplayer = []
```

New attributes: self.score / self.player / self.endplayer

```
# put all the players into the self.player list
elif msg['action'] == 'snakegaming':
    from_name = self.logged_sock2name[from_sock]
    self.player.append(from_name)

# collect the scores
# check whether the game procedure is finished for every player
elif msg['action'] == 'snakegamed':
    from_name = self.logged_sock2name[from_sock]
    # put all the players who finished the game into the self.endplayer list
    # check the gaming process by checking whether the number of players in the self.player and self.endplayer list
    try:
        self.endplayer.append(from_name)
        self.score[from_name] = msg['score']
        print(self.score)
    except:
        pass
    print(self.player)
    print(self.endplayer)

    the_guys = self.group.list_me(from_name)
    for g in the_guys[1:]:
        to_sock = self.logged_name2sock[g]
        if len(self.player) == len(self.endplayer):
            mysend(to_sock, json.dumps({"action": "over", "msg": "Score Uploaded."}))
        else:
            mysend(to_sock, json.dumps({"action": "snakegamed", "msg": "Score Uploaded."}))
    if len(self.player) == len(self.endplayer):
        mysend(from_sock, json.dumps({"action": "over", "msg": "Score Uploaded."}))

# send all the scores back to the players
# send the game results back to the players
elif msg['action'] == 'w':
    from_name = self.logged_sock2name[from_sock]
    for i in self.score.values():
        t = (i == max(self.score.values()))

        if t == False:
            if self.score[from_name] == max(self.score.values()):
                s = '--Congratulations! You win!--\n'
            else:
                s = '--You lose--\n'
            break
    if t == True:
        s = '--Draw--\n'
    mysend(from_sock, json.dumps({'msg':'\n--Game Finished--\n' + str(self.score) + '\n' + s}))
```

Add 3 ‘action’ states in handle_msg(self, from_sock) function to handle the snake game process. (See detail in comments)

4. client_state_machine.py

```
elif self.state == S_CHATTING:
    if len(my_msg) > 0:     # my stuff going out
        mysend(self.s, json.dumps({"action":"exchange", "from":"[" + self.me + "]", "message":my_msg}))
        if my_msg == 'bye':
            self.disconnect()
            self.state = S_LOGGEDIN
            self.peer = ''
    if len(peer_msg) > 0:  # peer's stuff, coming in
        # ----------your code here------#
        peer_msg = json.loads(peer_msg)

        if 'action' not in peer_msg:
            pass

        elif peer_msg["action"] == "connect":
            self.out_msg += "(" + peer_msg["from"] + " joined)\n"
        elif peer_msg["action"] == "disconnect":
            self.out_msg += peer_msg["message"]
            self.state = S_LOGGEDIN
        else:
            self.out_msg += peer_msg["from"] + peer_msg["message"]
            # plugging in snakegame
            if peer_msg['message'].lower() == 'snakegame?':
                self.out_msg +="\n--Game Request: Snake--\nPlease respond 'Yes' or 'No'."
                self.state = 'S_SNAKEWAITING'
            elif peer_msg['message'] == '--Game Request Accepted--':
                self.state = 'S_SNAKEGAMING'

        # ----------end of your code----#
    if self.state == S_LOGGEDIN:
        # Display the menu again
        self.out_msg += menu
```

To guarantee the program can run in the newly added code without ‘action’. And add an access to the snake game process.

```
# inviter waits other peers to respond
elif self.state == 'S_SNAKEWAITING':
    if my_msg == 'No':
        # back to chatting status
        self.state = S_CHATTING
        mysend(self.s, json.dumps({"action":"exchange", "from":"[" + self.me + "]", "message":'--Game Request Refused--'}))
    elif my_msg == 'Yes':
        # respond to the peer and server, change the status to gaming
        mysend(self.s, json.dumps({"action":"exchange", "from":"[" + self.me + "]", "message":'--Game Request Accepted--'}))
        self.state = 'S_SNAKEGAMING'

# gaming status
# send the score back to the server
elif self.state == 'S_SNAKEGAMING':
    a = 0
    mysend(self.s, json.dumps({"action":"snakegaming", "from":"[" + self.me + "]", "message":'--Game finished--', 'score': a}))
    score = gamesnake.main()
    mysend(self.s, json.dumps({"action":"snakegamed", "from":"[" + self.me + "]", "message":'--Game finished--', 'score': score}))
    if len(peer_msg) > 0:
        peer_msg = json.loads(peer_msg)
        if peer_msg["action"] == "over":
            self.state = 'S_SNAKERESULT'
    else:
        self.state = 'S_SNAKEREWAITING'

# wait for other players to finish the game
elif self.state == 'S_SNAKEREWAITING':
    if len(peer_msg) > 0:
        peer_msg = json.loads(peer_msg)
        if peer_msg["action"] == "over":
            self.state = 'S_SNAKERESULT'
        else:
            pass

# all the players have finished the game
# server returns the score of all players back to them
# switch back to chatting status
elif self.state == 'S_SNAKERESULT':
    mysend(self.s, json.dumps({"action":"w", "from":"[" + self.me + "]", "message":'--Game finished--'}))
    if len(peer_msg) > 0:
        peer_msg = json.loads(peer_msg)
        self.out_msg += peer_msg['msg']
        self.state = S_CHATTING
```

Add 4 states in proc(self, my_msg, peer_msg) to handle the gaming procedure from game request, acceptance/refuse, gaming, return individual scores, to receive all the scores and the game result from the server. (See detail in comments)

Simple explanation for 8 extra files we added (there are detailed comments in the files you can read for more information)

1. interface.py
This is the main GUI construction file. It's inserted and cooperating with the background cha.system.

2. sign_up.py
This is the sign-up interface. You need to go through this interface then go to the interface

3. k_shift_encryption.py
This is designed for store the account information. When you sign up, your information will be encrypted and stored. When I read your information again, it will be decrypted.

4. NYUSH.gif
This Is the background picture for sign up window

5. snakegame.py
(Reference: http://inventwithpython.com/pygame/chapter6.html)
This file is the game program file. I cited from the link above. Here are the places I changed,
main():

```
    showStartScreen()
    runGame()

    try:
        showGameOverScreen(a)
        terminate()
    except:
        pass
    return a
```

Change the terminate way in order to close the pygame window only without quit python

showGameOverScreen(a):

```
def showGameOverScreen(a):
    gameOverFont = pygame.font.Font('freesansbold.ttf', 80)
    gameSurf = gameOverFont.render('Game Over', True, WHITE)
    overSurf = pygame.font.Font('freesansbold.ttf', 40).render('Your Score: ' + str(a), True, WHITE)
    gameRect = gameSurf.get_rect()
    overRect = overSurf.get_rect()
    gameRect.midtop = (WINDOWWIDTH / 2, 150)
    overRect.midtop = (WINDOWWIDTH / 2, 250)


    DISPLAYSURF.blit(gameSurf, gameRect)
    DISPLAYSURF.blit(overSurf, overRect)
    drawPressKeyMsg2()

    #quit the game automatically or by click the 'x' box
    run = True
    clock = pygame.time.Clock()
    dt = 0
    while run:
        dt += clock.tick()
        for e in pygame.event.get():
            if e.type == pygame.QUIT:
                run = False
        if dt >= 1000:
            run = False
        pygame.display.update()
```

Add the score onto the final screen, and a way to close the window automatically.

6. account_info.txt

This is the txt file where I store the encrypted account information.

7. temporary_login_name.txt

This is the temporary txt file, when you login in, I will store your username here, and when open the interface, it will be loaded and used as your username.

8. Wrong_Times.txt

The place where if you type your password wrong, I will record you. If wrong for 3 times, you will be locked.

Works Cited

1. https://docs.python.org/3/library/tkinter.html
   Tkinter official documentation

2. https://www.bilibili.com/video/av16942112?from=search&seid=18001351313575636593
   bilibili <Tkinter 做简单的窗口视界 (GUI 莫烦 Python 教程) >

3. https://www.python-course.eu/tkinter_events_binds.php
   《Tkinter Events and Binds》

4. https://www.runoob.com/python/python-gui-tkinter.html
   《Python GUI 编程(Tkinter) | 菜鸟教程 》

5. http://inventwithpython.com/pygame/chapter6.html
   Source Code to Wormy