Introduction

○ Showing the motivation/project context

○ Introducing the structure of your app

Discussion

○ explanation of approach & work during the project

Live-show case

○ A simple working demo of your code

Analysis of design choices and commentary

○ talking about the reasons for/comments on your design

8-12 minutes

**Motivation**:

We two are all game lovers, by any means, we would definitely choose the Online Gaming for our topic. The game suggested on the assignment sheet is tic-tac-toe, so we think we could at least elaborate on that idea and make a relevantly more complex game, that is, gobang. Gobang is a traditional game in china, and is played by a wide range of age groups. We at NYU Shanghai also has a gobang board on the second floor, and we friends would still go there and play the gobang for relaxation. Such a game with a bit of competitivity and a bit of casualness is the best for some fun and leisure time between friends. For the chat system we built in ICS, we also think that there need to be some kind of fun elements that could share between users, for right now it is only a quite limited and boring chat system. Given that, we think integrating a gobang game to the chat system would be the best. And let’s see what our Gobang game is like.

**Structure of the App**:

It is a Gobang game. First, similar to chatting, user need to type g + peer\_name to connect with peer, then the server would transmit messages to the peer, if the connection is successful, then the game window would open on both users’ window. Also, if the peer is already in game with other users, it would show that that user is unavailable. The server would inform which color the user is assigned, as well as whether he/she is on the offensive or on the defensive side. Then it would be the game between two users, and the game operates just the same with the traditional gobang game. On detecting winning, the game would display the winning or losing signal on each users’ screen. And a hint showing the user that pressing ‘Q’ would quit the game. On pressing ‘Q’, both of the user would quit and return to the main chat room, then they could do whatever they want.

**Explanation of approach**:

1. since we just learn OOP, we think it would be the best to utilize Python OOP in our game, so mainly, we divide the methods of our App class into several parts, first is the main game loop: on\_execute(), then it is on\_init() for initializing the game; then we separate the game loop running() into three parts: on\_loop() for game logic, on\_render() for updating visual elements, on\_event() for detecting user interaction; finally it’s the on\_cleanup() for ending the game.
2. For incorporating the game into the chat system, we import the game module into the client\_state\_machine, and if the users connection with the peer is successful, both of their state would change from S\_LOGGEDIN to S\_GAMING. and in the client\_state\_machine, we would call the game object, initialize it and call the main game loop: on\_execute(). since the game is running a loop and the client state machine would be jammed until the game stops. Therefore, following the calling of the game loop, we immediately write the out\_msg += “Quited” and change the client state back to S\_LOGGEDIN.
3. server receive issue, we found that if we simply add receive message method (myrecv) into the game loop, pygame will be jammed and unresponsive on its start, which we conclude that it is because the receiving loop is conflicting with the game loop. So, we decide to use the Python threading package to parallelize these two processes in the main game loop, and the result turn out to be quite effective.
4. for the main game design, since it is a gobang game, the main two tasks to handle are: detecting player interaction and synchronizing the chess movement on two screen and detecting win/lose.

1. We created a 2d matrix for the board on initialization, and every time the user put down a new chess, we modify the matrix, so that it would be easy to detect the state of winning or losing on the end of every loop; it would also make it easier for judging whether a move is valid, for there would be cases, if the user accidentally try to override the opponents chess position.

2. In order to let the game be played one move after another, we initialize the my\_turn state, which is a Boolean switch, and change it on every move made.

3. for the synchronizing, we use the server to transmit the position of the newly-added chess. To achieve this, we passes the socket of the user: self.s on initialization of the game, then we could simply call mysend() to send the user status/action to the server and transfer it to the other user. And it works the same for quitting.

**Live show case**:

**Visual Design**: We design the color palette with respect to the cyberpunk color palette. It is a popular trend that has become more popular thanks to the to-be-released game Cyberpunk 2077. A lot of games that has few connections with such a theme were put into the cyberpunk context, which some are successful and some are rather plain and far-fetched. We are trying to make an experiment on the gobang game, to see more possible applications of this cool, popular trend.

It is also making the game more visually interesting with colors of sharp contrast and mimicking the neon lights. Rather than a traditional gobang game that has simple wood color for the board and black and white for the chesses, which might seem a bit boring. And that is why we call our game Gobang 2077!

**Comments & Possible Improvements**:

Most designs of our app, from using an OOP structure, to using threading to avoid jamming, to a smooth instruction and operation are all reasonable and well-implemented.

Admittedly, we do have one issue with the multi-threading: since the thread could only be started but no direct method to pause or stop it, on exiting the game, our message transmission to the server would be jammed by the myrecv() thread, and if we try to implement a restart button to the game, the gameloop thread: running() would also jam the re-initialization of the game. And these are the two things we need to come up with a solution to.

And it is actually quite possible to add an AI into the game, so that if the user simply types g + his/her own user name, the game would also start and the user could play a game with the computer AI. The solution for such function is quite easy, the key is to let the AI consider all possible future steps and determine how many step further should the AI take into account.