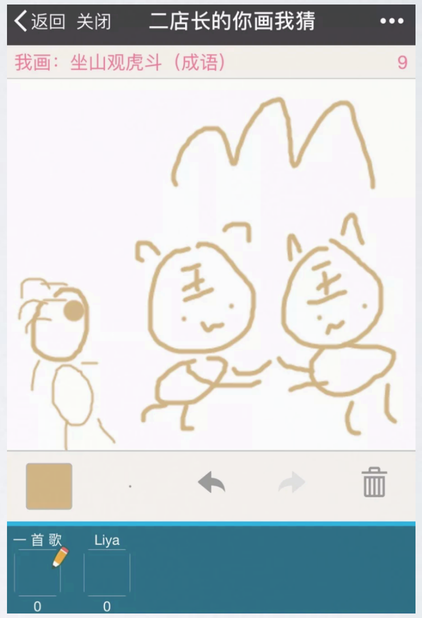
DRAW AND GUESS

A Game Based on the Chat System

Zixin Wang & Chuyi Zhang

INTRODUCTION

- Context: A game going viral on Wechat
- Game Play: two users, take turns, realtime
- GameGUI, weChat mini program



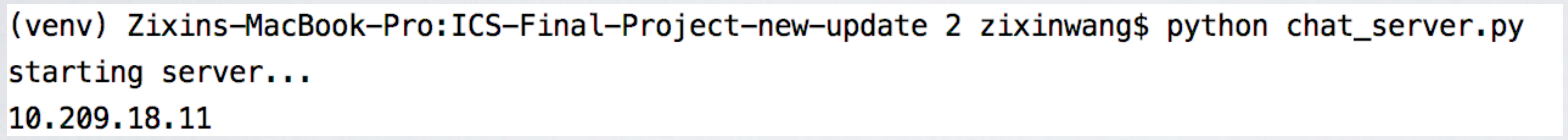
DEMO

PROGRESS

- Embed Game Grouping into the Chat System
- Connect GUI to the Chat System
- GUI Development
- Game Systems Development
- Real Time Images Transmission and Buffering
- System Optimisation (GUI Styling)

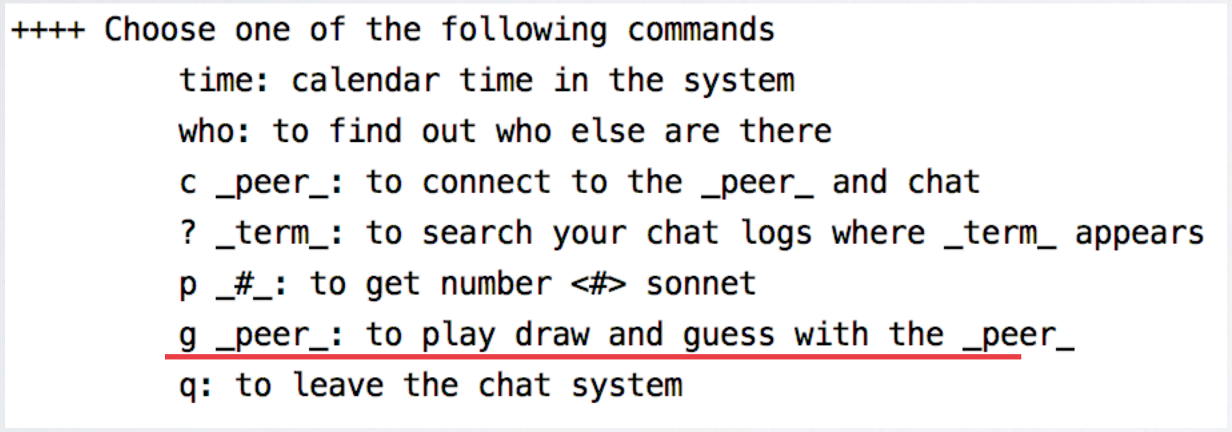
Embed Game Grouping into the Chat System

Enable using chat system on different computers

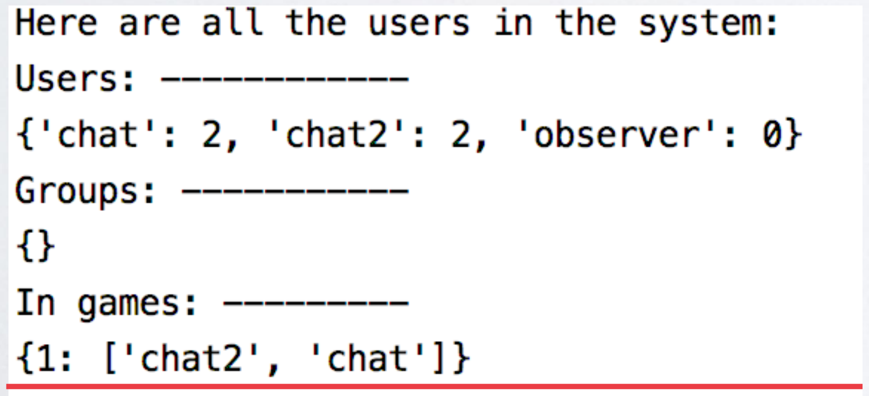


```
(venv) Zixins-MacBook-Pro:ICS-Final-Project-new-update 2 zixinwang$ python chat_server.py
starting server...
10.209.18.11
```

Create game group management



```
++++ Choose one of the following commands
         time: calendar time in the system
         who: to find out who else are there
         c _peer_: to connect to the _peer_ and chat
         ? _term_: to search your chat logs where _term_ appears
         p _#_: to get number <#> sonnet
         g _peer_: to play draw and guess with the _peer_
         q: to leave the chat system
```

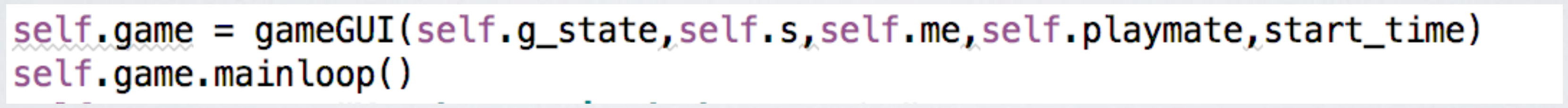


```
Here are all the users in the system:
Users: ------------
{'chat': 2, 'chat2': 2, 'observer': 0}
Groups: -----------
{}
In games: ---------
{1: ['chat2', 'chat']}
```

Connect GUI to the chat system

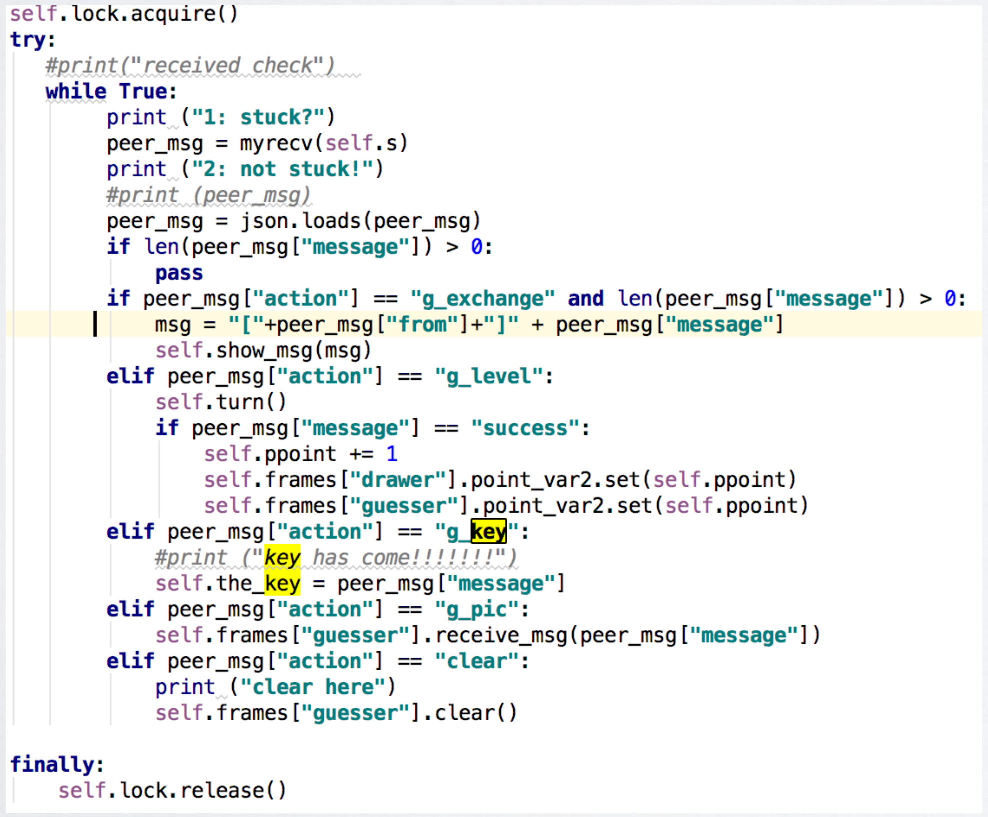
- Use threading to actively receive messages from the server

Problem: The tkinter module is not completely thread safe



```
self.game = gameGUI(self.g_state,self.s,self.me,self.playmate,start_time)
self.game.mainloop()
```

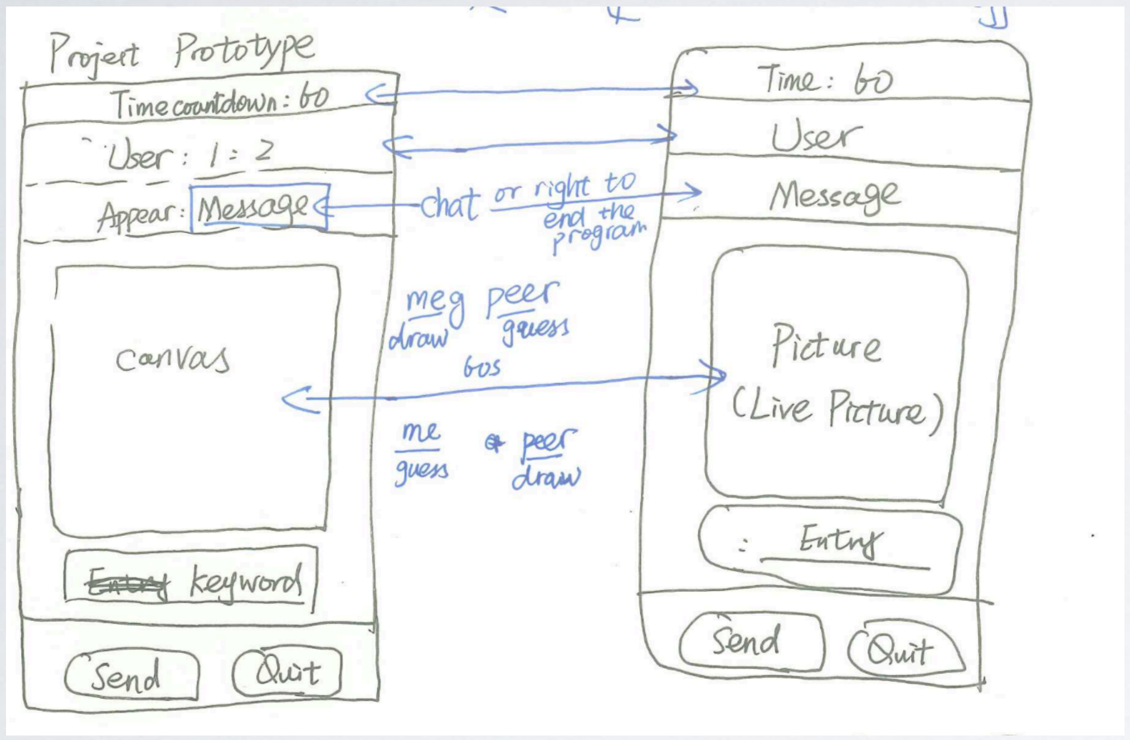
Solution: Put GUI operations outside of the threading, and use the new thread mainly receive messages (with lock())



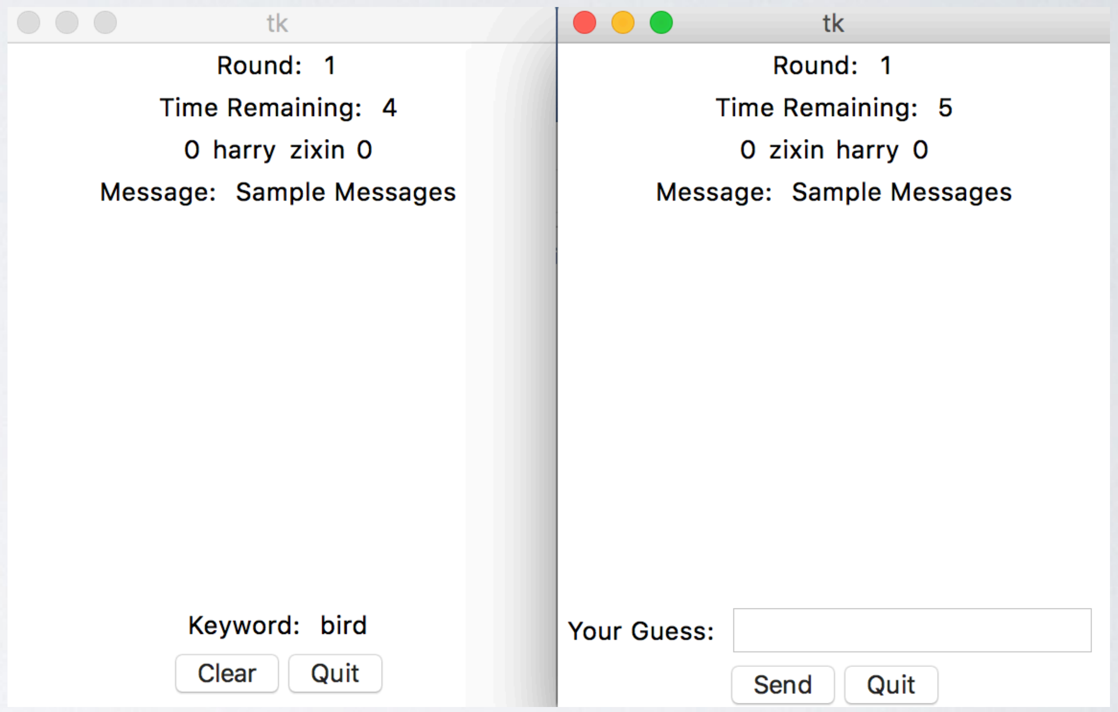
```
self.lock.acquire()
try:
   #print("received check")
   while True:
       print ("1: stuck?")
       peer_msg = myrecv(self.s)
       print ("2: not stuck!")
       #print (peer_msg)
       peer_msg = json.loads(peer_msg)
       if len(peer_msg["message"]) > 0:
           pass
       if peer_msg["action"] == "g_exchange" and len(peer_msg["message"]) > 0:
           msg = "["+peer_msg["from"]+"]" + peer_msg["message"]
           self.show_msg(msg)
       elif peer_msg["action"] == "g_level":
           self.turn()
           if peer_msg["message"] == "success":
               self.ppoint += 1
               self.frames["drawer"].point_var2.set(self.ppoint)
               self.frames["guesser"].point_var2.set(self.ppoint)
       elif peer_msg["action"] == "g_key":
           #print ("key has come!!!!!!")
           self.the_key = peer_msg["message"]
       elif peer_msg["action"] == "g_pic":
           self.frames["guesser"].receive_msg(peer_msg["message"])
       elif peer_msg["action"] == "clear":
           print ("clear here")
           self.frames["guesser"].clear()

finally:
    self.lock.release()
```

GUI Development

Sketch



GUI



tk

Round: 1

Time Remaining: 4

0 harry zixin 0

Message: Sample Messages

Keyword: bird

tk

Round: 1

Time Remaining: 5

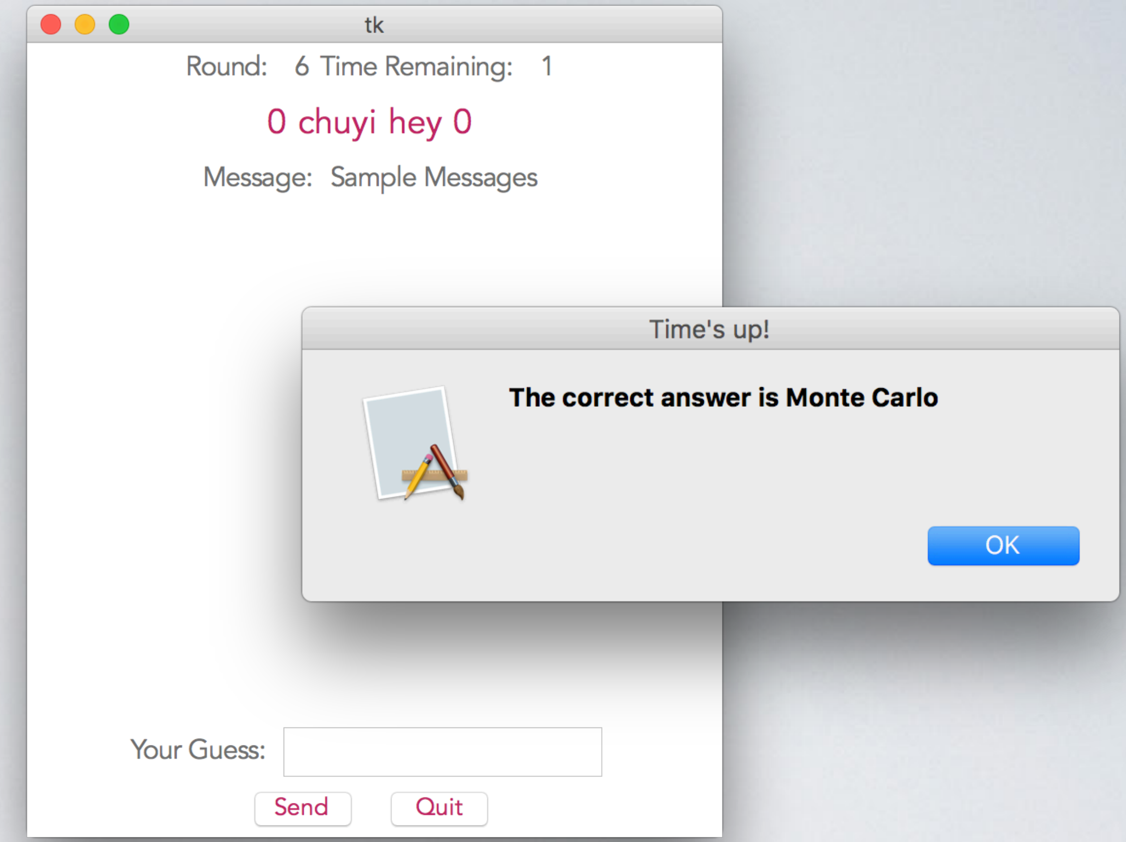
0 zixin harry 0

Message: Sample Messages

Your Guess:

Game System

- Timing
- Validation
- Scoring System
- Information Distribution
- Pop-up Message-box
- Roles Switching


Round: 6 Time Remaining: 1

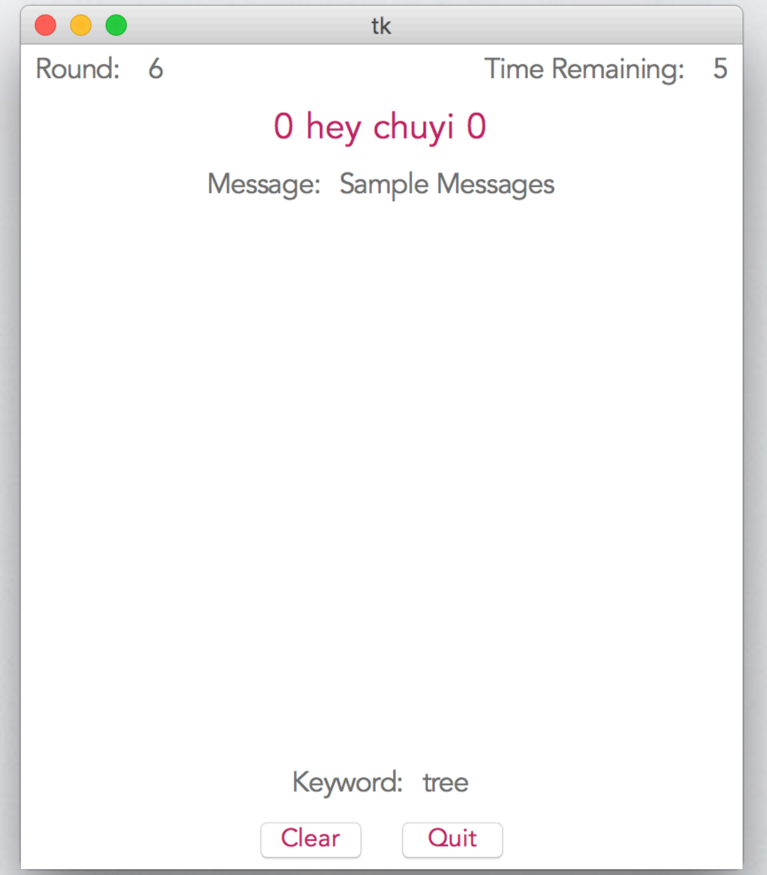
0 chuyi hey 0

Message: Sample Messages

Time's up!

The correct answer is Monte Carlo

Your Guess:


Round: 6 Time Remaining: 5

0 hey chuyi 0

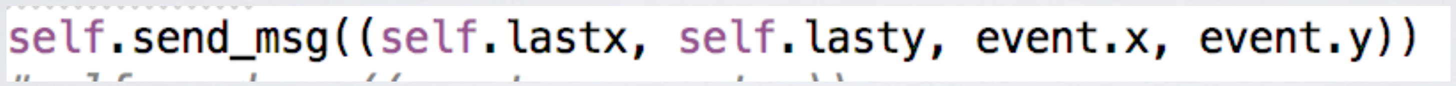
Message: Sample Messages

Keyword: tree

Real-time Image Transmission and Buffering

Problem: The socket transmission speed is **so slow**

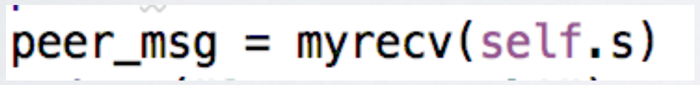
Send while drawing:



```
self.send_msg((self.lastx, self.lasty, event.x, event.y))
```

# takes around 0.1* e^-5 s per "send"

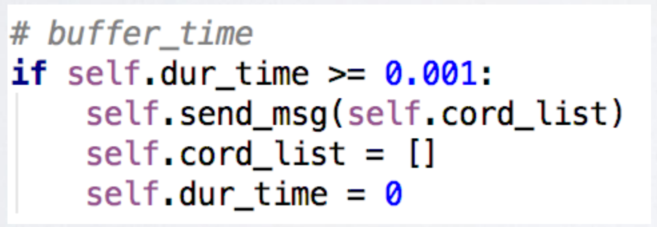
Receiving:



```
peer_msg = myrecv(self.s)
```

# takes around 0.2s per "recv"

Solution: Buffering



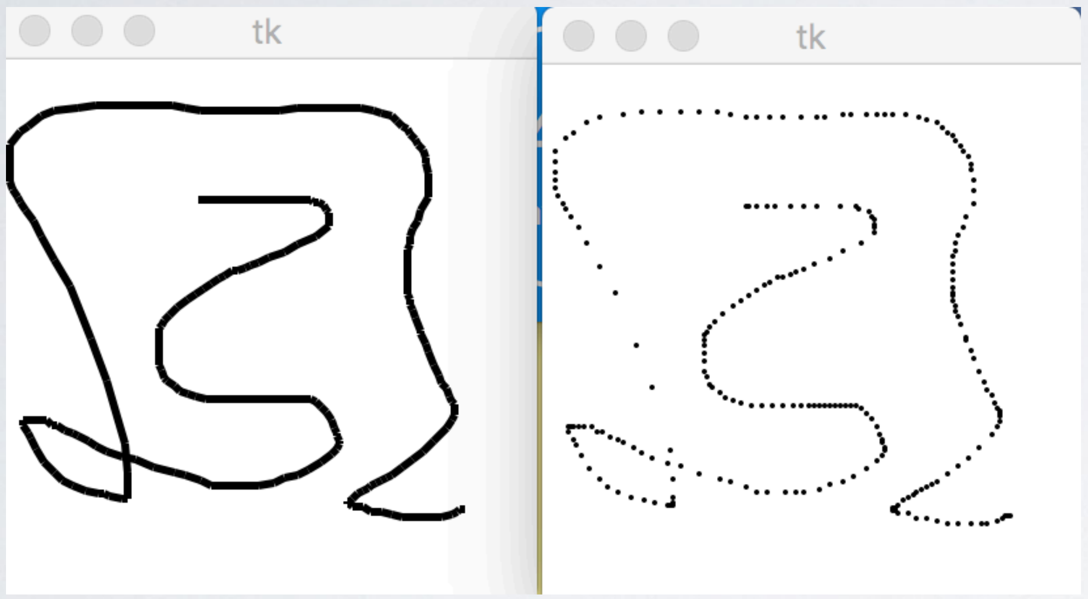
```
# buffer_time
if self.dur_time >= 0.001:
    self.send_msg(self.cord_list)
    self.cord_list = []
    self.dur_time = 0
```

# set the timer, store the data, send the data all at once when timer expires

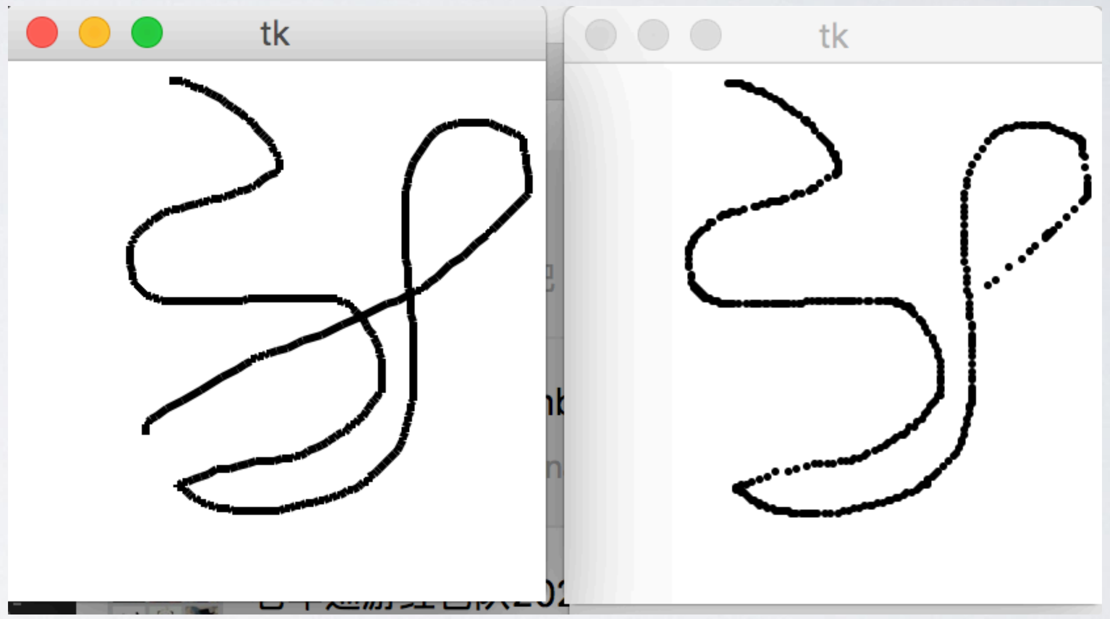
Tradeoff: The the drawing would be not exactly "real time", according to our research, this problem exists for many current drawing games

Real-time Image Transmission and Buffering

Version I (with out buffering)



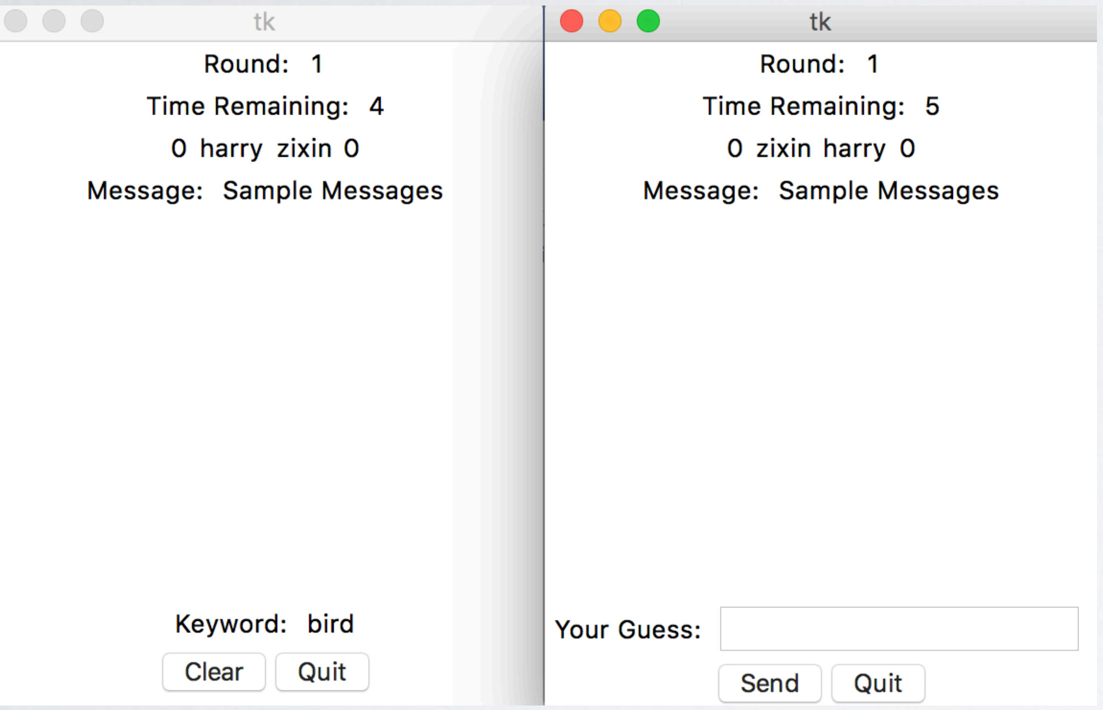
Version II (with buffering)



Still, it's not the perfect solution

System Optimisation(GUI Styling)

Previous


tk

Round: 1

Time Remaining: 4

0 harry zixin 0

Message: Sample Messages

Keyword: bird

Clear Quit

tk

Round: 1

Time Remaining: 5

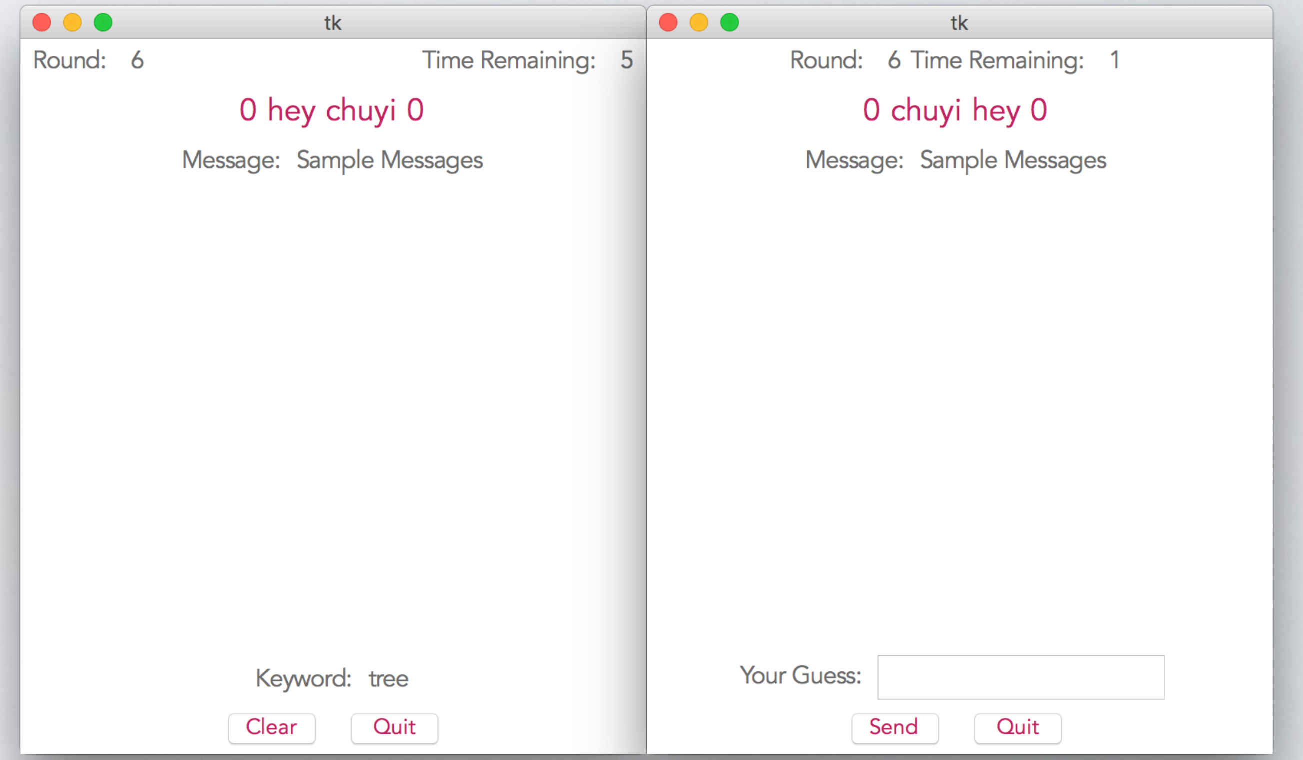
0 zixin harry 0

Message: Sample Messages

Your Guess:

Send Quit

Now


tk

Round: 6 Time Remaining: 5

0 hey chuyi 0

Message: Sample Messages

Keyword: tree

Clear Quit

tk

Round: 6 Time Remaining: 1

0 chuyi hey 0

Message: Sample Messages

Your Guess:

Send Quit

DESIGN CHOICES

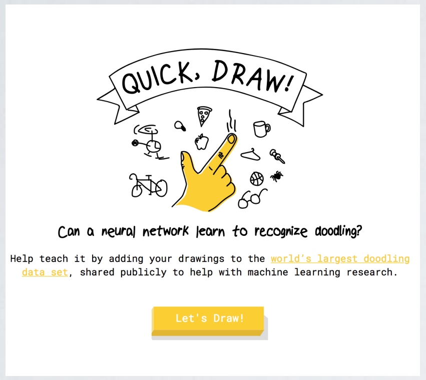
- Why using GUI?
- Why real time?

TO BE DEVELOPED

- A Multi-user Game
- Better Transmission Algorithm
- Game Animation
- Add AI into the Game

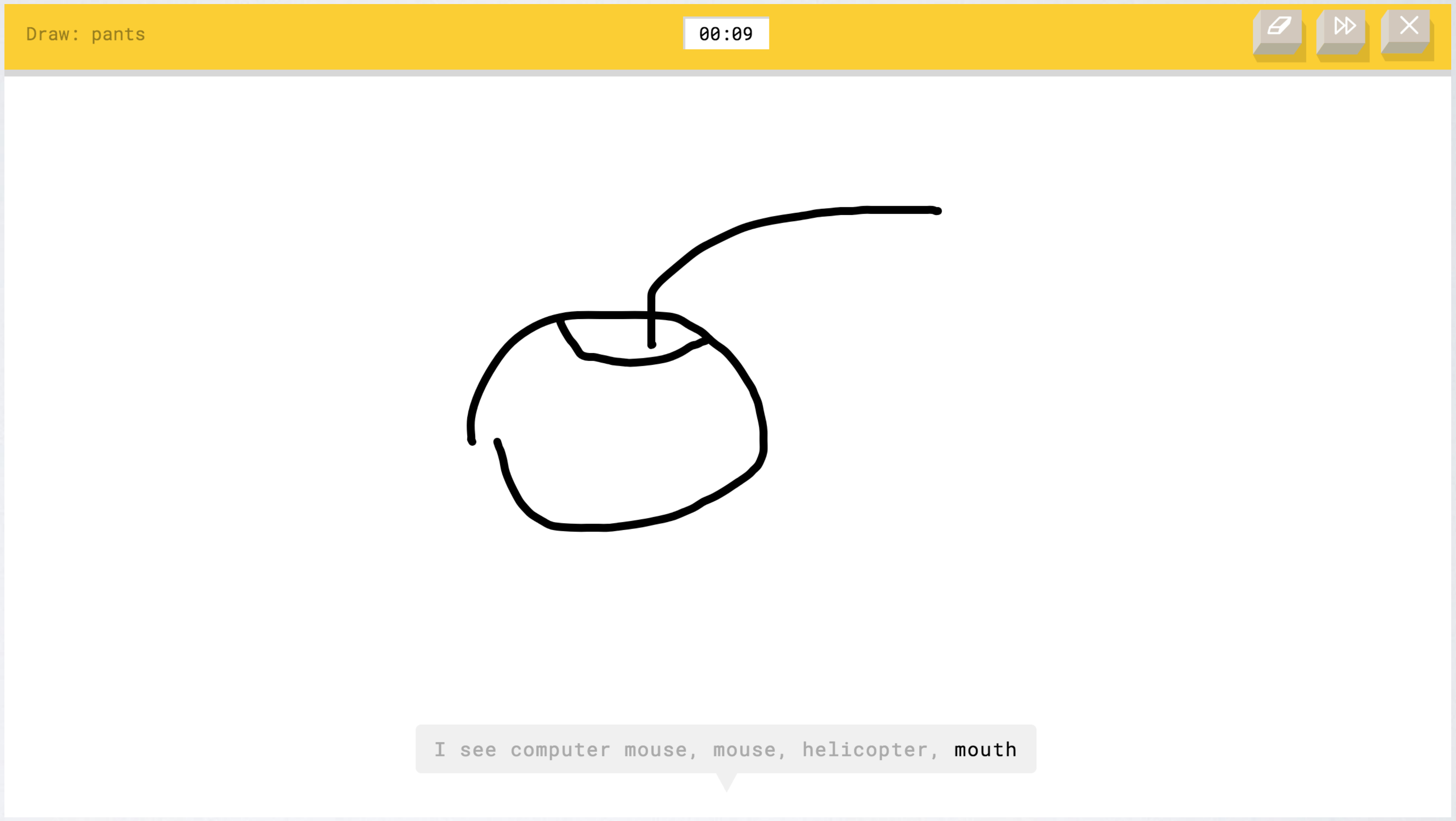
DRAWING GAME AND A.I.

Google Quick Draw and Deep Neural Network



Can a neural network learn to recognize doodling?

Help teach it by adding your drawings to the world's largest doodling data set, shared publicly to help with machine learning research.



I see computer mouse, mouse, helicopter, mouth