import os

from tkinter import \*

import tkinter.ttk as ttk

import socket, sys, threading,time,struct, tkinter.colorchooser, os, getpass

from tkinter import font,filedialog, messagebox

from PIL import Image, ImageTk

from gameobjects import \*

from chat\_gui import \*

from opponent import \*

from pygame import mixer # Load the required library

class Tetricia(Tk):

"""The main client application"""

def \_\_init\_\_(self):

Tk.\_\_init\_\_(self)

mixer.pre\_init(44100, 16, 2, 4096)

mixer.init()

default\_font = font.nametofont("TkDefaultFont")

default\_font.config(family='Comic Sans MS')

self.sounds={"lock":mixer.Sound("effects/lock.ogg"),

"rotate":mixer.Sound("effects/rotate.ogg"),

"move":mixer.Sound("effects/move.ogg"),

"clear":mixer.Sound("effects/lineclear.ogg"),

"over":mixer.Sound("music/gameover.ogg"),

"bg":mixer.Sound("music/bg.OGG"),

"bg1":mixer.Sound("music/bg1.OGG"),

"bg2":mixer.Sound("music/bg2.OGG"),

"bg3":mixer.Sound("music/bg3.OGG")

}

self.screenw=self.winfo\_screenwidth()

self.screenh=self.winfo\_screenheight()

self.protocol("WM\_DELETE\_WINDOW", self.\_destroy)

self.bind("<Escape>", self.esc)

self.bind("<F11>", self.f11)

self.chat=ChatGui(self,'aronsv.ddns.net', '64164', socket.gethostname()+'\\'+getpass.getuser())

self.chat.grid(row=2,column=0, sticky="W")

#self.chat.config(width=500, height=1000)

self.players={}

self.title("Tetrícia")

##Game difficulcity

self.level=1

##The connection socket

self.conn=None

##Button status tracker boolean

self.ready=False

##~UwU~

self.playing=False

self.trial\_start()

def esc(self, evt):

"""Escape fullscreen"""

self.attributes('-fullscreen', False)

def f11(self, evt):

"""Mode fullscreen"""

self.attributes('-fullscreen', True)

def set\_scale(self, n):

"""Scale the widgets wheter it's up to to or up to 5 players"""

##Maximum number of players

self.max=n

if n==2:

k=18

self.panel=GameDashboard(self, mixer,self.sounds,blocksize=self.screenh/20.5)

self.panel.grid(row=0,rowspan=3 ,column=0,sticky="SW")

self.chat.grid(row=2,column=1, sticky="SE")

self.yscale=0.98

elif n==5:

k=27

self.yscale=0.64

self.panel=GameDashboard(self, blocksize=self.screenh/k)

self.panel.grid(row=0,rowspan=2 ,column=0,sticky="NW")

self.panel.startButton.config(command=self.button\_send\_ready,state=NORMAL)

self.panel.online=True

def add\_player(self, name):

"""When a new player joins the server, place their dashboard"""

self.players[name]=OpponentDashboard(self,self.screenh/27\*self.yscale, name, self.level)

if self.max==2:

self.players[name].grid(row=0,column=1,sticky="NE", rowspan=3)

return

curr\_in=len(self.players)

if curr\_in>2:

self.players[name].grid(row=1,column=2-(curr\_in%2),sticky="N", rowspan=2)

else:

self.players[name].grid(row=0, column=2-(curr\_in%2), sticky="N")

def remove\_player(self, name):

"""When a player leaves the server, clear it up"""

self.players[name].destroy()

del self.players[name]

if not self.playing:

self.panel.startButton.config(state=NORMAL)

def set\_level(self,level):

"""Set the default game speed"""

self.level=level

self.panel.level=level

self.panel.set\_levels(level)

def check\_ready(self):

"""Check if everyone is ready and if so, start the game."""

for i in self.players:

if self.players[i].ready==False:

return

if self.ready:

self.playing=True

self.panel.start\_new\_game()

def check\_over(self):

"""Check if the game is over"""

ingame=0

for i in self.players:

if not self.players[i].gameOver:

ingame+=1

#This player won

if (self.panel.ingame and ingame==0):

self.panel.gameThread.won()

self.chat.write("You have won this round!")

self.reset()

elif (not self.panel.ingame and ingame==0):

self.panel.gameThread.won()

self.chat.write("Highest score wins!")

self.reset()

def reset(self):

"""Set some property to be able to start a new game"""

self.playing=False

self.panel.startButton.config(state=NORMAL)

for i in self.players:

self.players[i].gameOver=False

self.players[i].ready=False

def button\_send\_ready(self):

"""Overriden button action! Send the server a ready status, set ourselves in ready state"""

self.panel.startButton.config(state=DISABLED)

self.update\_server("#GAME#READY#0")

self.ready=True

for i in self.players:

self.players[i].defaults()

self.check\_ready()

def set\_ready(self,name):

"""Setting an opponent to READY to play state"""

self.players[name].set\_ready()

def set\_player(self, name, msg):

"""Forward the action log to the corresponding panel"""

print(name,msg)

self.players[name].log(msg)

def update\_server(self, msg):

"""Send the up-to-date information to the server"""

self.panel.netLock.acquire()

self.ready=False

msg=chr(0)+msg+"#"+self.chat.data[2].get()+chr(0)

self.conn.send(bytes(msg,'utf-8'))

self.panel.netLock.release()

def set\_connection(self, conn):

"""Set the server connection socket"""

self.conn=conn

self.attributes('-fullscreen', True)

def drop\_connection(self):

"""Set the server connection socket"""

self.panel.grid\_forget()

self.conn=None

self.panel.startButton.config(state=DISABLED)

self.panel.online=False

for i in self.players:

self.players[i].destroy()

self.players={}

self.attributes('-fullscreen', False)

def \_destroy(self):

"""Destroy event handler"""

self.chat.\_delete\_window()

if self.conn!=None:

self.panel.\_destroy()

self.after(200,self.destroy)

def opponent\_control\_test(self):

"""test opponent.py"""

self.max=1

self.player2=OpponentDashboard(self,20,"AAAA")

self.player2.grid(row=0,column=1,sticky=N)

self.player2.new\_mino("T")

time.sleep(0.5)

for y in range(19,0,-1):

self.player2.set\_coords("[(4,{0}),(5,{0}),(6,{0}),(5,{0}+1)]".format(y))

time.sleep(0.1)

self.player2.lock\_down()

self.player2.set\_eliminate("[0]")

time.sleep(0.5)

self.player2.set\_eliminate("[0]")

time.sleep(0.3)

self.player2.set\_eliminate("[0]")

self.player2.set\_statistics("6464,5,64")

time.sleep(1)

self.player2.grid\_forget()

def trial\_start(self):

"""Quick test for opponent.py game events"""

threading.Thread(target=self.opponent\_control\_test).start()

if \_\_name\_\_ == '\_\_main\_\_':

os.system('cls')

Tetricia().mainloop()