**Proiect: Joc de Table**

Realizat de:

Mercic Cristian

Roșanu Alexandru

Gr.6303

Cuprins

1.Scop

2.Regulile jocului

3.Cod

**1.Scop**

Pentru acest proiect avem ca scop realizarea unui joc în limbajul de programare python. Un joc de table cu o interfață grafica care va oferi posibilitatea de a juca atat contra calculatorului cât și contra altor jucatori. Această aplicație va ține cont de toate regulile jocului.

**2.Regulile jocului**

**01. Descriere generala si prezentarea tablei de joc**  
Table este un joc popular pentru 2 jucatori si este o combinatie de noroc si strategie. Jocul se desfasoara pe o tabla de joc compusa din 24 de triunghiuri inguste (locurile pe care se aseaza piesele). Triunghiurile sunt de culori alternante si sunt grupate in 4 zone de cate 6 triunghiuri. Zonele sunt denumite astfel: Casa jucatorului si Exterior, si Casa adversarului si Exterior. Casele si exterioarele sunt separate de o bara pe centrul tablei de joc.  
  
Fiecare jucator are 15 piese de culoare alba sau neagra. Piesele sunt asezate initial dupa cum se vede in imaginea alaturata.  
  
  
  
**02. Scopul Jocului**Scopul jocului este sa mutati toate piesele in casa proprie si sa le scoateti din casa inaintea adversarului. In imagine puteti vedea directia de mutare a pieselor negre. Piesele albe se muta in directia opusa.

**03. Mutarea pieselor**  
  
Cel care face masa incepe jocul. Dupa ce da cu zarurile (click pe butonul "da cu zarurile"), jucatorul va muta piesele sale un numar de triunghiuri (locuri) cate indica zarurile date. Piesele se muta intotdeauna inainte, pe directia de joc (dupa cum ati putut vedea in imaginea anterioara). Se aplica urmatoarele reguli:  
  
1. O piesa poate fi mutata doar la un loc liber, care sa nu fie ocupat de 2 sau mai multe piese adverse.  
  
2. Numerele celor 2 zaruri constituie mutari separate. De exemplu, daca un jucator da 3 si 5, el poate muta o piesa 5 locuri la un loc liber si alta piesa cu 3 locuri la un loc liber, sau poate muta o piesa un total de 8 locuri la un loc liber, dar doar daca locul intermediar (situat la 3 sau 5 locuri de la locul de start) este de asemenea loc liber.  
   
  
  
3. Un jucator care da o dubla joaca numerele de pe zaruri de doua ori. Un zar de 6 si 6 inseamna ca jucatorul are de facut 4 mutari de 6, si poate muta orice combinatie de piese o crede potrivita pentru a completa aceasta cerinta.  
  
4. Un jucator trebuie sa foloseasca ambele numere (de pe cele 2 zaruri) daca aceasta este posibil (sau toate 4 mutarile la o dubla). Cand doar un singur numar poate fi jucat, jucatorul trebuie sa joace acel numar (mutare). Daca doar un numar poate fi jucat dar nu amandoua, jucatorul trebuie sa joace numarul mai mare. Cand nici un numar nu poate fi jucat, jucatorul isi pierde randul (nu joaca). In cazul dublelor, cand nu se pot juca toate cele 4 mutari, jucatorul trebuie sa joace cate mutari sunt posibile.

**04. Capturarea pieselor**Un loc ocupat de o singura piesa poate fi lovita de o piesa de a adversarului, atunci cand piesa adversa ajunge pe acel loc. Piesa lovita este capturata si asezata pe bara (pe axa centrala a tablei).  
  
Oricand un jucator are una sau mai multe piese pe bara, prima sa obligatie e sa intre cu acea piesa (sau acele piese) in casa adversarului. O piesa este intrata prin mutarea sa intr-un loc liber in concordanta cu cat arata zarul.  
  
De exemplu, daca un jucator da 4 si 6, poate sa intre cu o piesa fie pe locul 4 fie pe locul 6 al adversarului, atat timp cat respectivul loc nu e ocupat de 2 sau mai multe piese ale adversarului.

Daca nici unul din locuri nu e liber, jucatorul isi pierde randul. Daca un jucator poate sa intre (cu unele) dar nu cu toate piesele sale, trebuie sa intre cu cate poate si apoi pierde restul din randul sau.  
Dupa ce ultima piesa a intrat, orice mutari nefolosite trebuie jucate, mutand fie piesa care a intrat fie alta.  
  
**05. Scoaterea pieselor - faza finala a jocului**Odata ce un jucator a adus toate cele 15 piese proprii in casa sa el poate sa inceapa sa scoata piesele. Jucatorul scoate o piesa dand cu zarurile un numar care sa corespunda cu numarul locului pe care se afla piesa si apoi o scoate de pe tabla. Astfel, dand un 6 jucatorul va scoate o piesa asezata la numarul 6.  
  
Daca nu este nici o piesa la locul indicat de zar, jucatorul trebuie sa faca o mutare corecta folosind o piesa situata pe un loc cu numar mai mare. Daca nu sunt piese situate pe locurile cu numere mai mari, jucatorului i se permite sa scoata o piesa de pe locul cu cel mai mare numar dintre cele pe care sunt piesele sale. Un jucator nu e obligat sa scoata piese daca poate sa faca alta mutare corecta.  
  
  
Un jucator trebuie sa aiba toate piesele sale active in casa sa pentru a putea sa scoata piese. Daca o piesa e lovita in timp ce piesele sunt scoase, jucatorul trebuie sa aduca piesa inapoi in casa sa inainte sa continue sa scoata piesele. Primul jucator care scoate toate cele 15 piese castiga jocul.

**3.Cod**

Joc

import pygame

pygame.init()

screen = pygame.display.set\_mode((1920, 1020))

#background

background = pygame.image.load('tablanoua2.png')

#pygame.display.update()

pygame.display.set\_caption("Table cu prietenii")

icon = pygame.image.load('backgammon.png')

pygame.display.set\_icon(icon)

#piese

#piesa1

piesa1=pygame.image.load('piesa1.png')

piesa1X=80

piesa1Y=30

def piesaA1():

screen.blit(piesa1, (piesa1X, piesa1Y))

#piesa2

piesa2=pygame.image.load('piesa1.png')

piesa2X=80

piesa2Y=120

def piesaA2():

screen.blit(piesa2, (piesa2X, piesa2Y))

#piese3

piesa3=pygame.image.load('piesa1.png')

piesa3X=80

piesa3Y=210

def piesaA3():

screen.blit(piesa3, (piesa3X, piesa3Y))

#piesa4

piesa4=pygame.image.load('piesa1.png')

piesa4X=80

piesa4Y=300

def piesaA4():

screen.blit(piesa4, (piesa4X, piesa4Y))

#piesa5

piesa5=pygame.image.load('piesa1.png')

piesa5X=80

piesa5Y=390

def piesaA5():

screen.blit(piesa1, (piesa5X, piesa5Y))

#piesa6

piesa6=pygame.image.load('piesa1.png')

piesa6X=520

piesa6Y=860

def piesaA6():

screen.blit(piesa6, (piesa6X, piesa6Y))

#piesa7

piesa7=pygame.image.load('piesa1.png')

piesa7X=520

piesa7Y=770

def piesaA7():

screen.blit(piesa7, (piesa7X, piesa7Y))

#piesa8

piesa8=pygame.image.load('piesa1.png')

piesa8X=520

piesa8Y=680

def piesaA8():

screen.blit(piesa8, (piesa8X, piesa8Y))

#piesa9

piesa9=pygame.image.load('piesa1.png')

piesa9X=820

piesa9Y=860

def piesaA9():

screen.blit(piesa9, (piesa9X, piesa9Y))

#piesa10

piesa10=pygame.image.load('piesa1.png')

piesa10X=820

piesa10Y=770

def piesaA10():

screen.blit(piesa10, (piesa10X, piesa10Y))

#piesa11

piesa11=pygame.image.load('piesa1.png')

piesa11X=820

piesa11Y=680

def piesaA11():

screen.blit(piesa11, (piesa11X, piesa11Y))

#piesa12

piesa12=pygame.image.load('piesa1.png')

piesa12X=820

piesa12Y=590

def piesaA12():

screen.blit(piesa12, (piesa12X, piesa12Y))

#piesa13

piesa13=pygame.image.load('piesa1.png')

piesa13X=820

piesa13Y=500

def piesaA13():

screen.blit(piesa13, (piesa13X, piesa13Y))

#piesa14

piesa14=pygame.image.load('piesa1.png')

piesa14X=1360

piesa14Y=30

def piesaA14():

screen.blit(piesa14, (piesa14X, piesa14Y))

#piesa15

piesa15=pygame.image.load('piesa1.png')

piesa15X=1360

piesa15Y=120

def piesaA15():

screen.blit(piesa15, (piesa15X, piesa15Y))

#game loop

open = True

while open:

screen.fill((0, 0, 0))

screen.blit(background, (0,0))

for event in pygame.event.get():

if event.type == pygame.QUIT:

open = False

piesaA1()

piesaA2()

piesaA3()

piesaA4()

piesaA5()

piesaA6()

piesaA7()

piesaA8()

piesaA9()

piesaA10()

piesaA11()

piesaA12()

piesaA13()

piesaA14()

piesaA15()

pygame.display.update()

pygame.quit()

quit()

Server

import socket

from \_thread import \*

import sys

server = "10.18.0.91"

port = 5555

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

try:

    s.bind((server, port))

except socket.error as e:

    str(e)

s.listen(2)

print("Waiting for a connection, Server Started")

def threaded\_client(conn):

    conn.send(str.encode("Connected"))

    reply = ""

    while True:

        try:

            data = conn.recv(2048)

            reply = data.decode("utf-8")

            if not data:

                print("Disconnected")

                break

            else:

                print("Received: ", reply)

                print("Sending : ", reply)

            conn.sendall(str.encode(reply))

        except:

            break

    print("Lost connection")

    conn.close()

while True:

    conn, addr = s.accept()

    print("Connected to:", addr)

    start\_new\_thread(threaded\_client, (conn,))

Network

import socket

class Network:

    def \_\_init\_\_(self):

        self.client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

        self.server = "10.18.0.91"

        self.port = 5555

        self.addr = (self.server, self.port)

        self.id = self.connect()

        print(self.id)

    def connect(self):

        try:

            self.client.connect(self.addr)

            return self.client.recv(2048).decode()

        except:

            pass

    def send(self, data):

        try:

            self.client.send(str.encode(data))

            return self.client.recv(2048).decode()

        except socket.error as e:

            print(e)

n = Network()

print(n.send("hello"))

print(n.send("working"))

Server.2

import socket

from \_thread import \*

import sys

def start\_my\_server():

    try:

        import socket

        server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

        server.bind(('127.0.0.1', 2000))

        server.listen(4)

        while True:

            print('Working...')

            client\_socket, address = server.accept()

            data = client\_socket.recv(1024).decode('utf-8')

            #print(data)

            content = load\_page\_from\_get\_request(data)

            client\_socket.send(content)

            client\_socket.shutdown(socket.SHUT\_WR)

    except KeyboardInterrupt:

        server.close()

        print('shutdown this shit...')

def load\_page\_from\_get\_request(request\_data):

    HDRS = 'HTTP/1.1 200 OK\r\nContent-Type: text/html; charset=utf-8\r\n\r\n'

    HDRS\_404 = 'HTTP/1.1 404 OK\r\nContent-Type: text/html; charset=utf-8\r\n\r\n'

    path = request\_data.split(' ')[1]

    response = ''

    try:

        with open('views' +path, 'rb') as file:

            response = file.read()

        return HDRS.encode('utf-8') + response

    except FileNotFoundError:

        return (HDRS + 'Sorry bro! No page...').encode('utf-8')

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

    start\_my\_server()