脑电波模块代码

using System;

using System.Diagnostics;

using System.Text;

using System.Threading;

using System.Reflection;

using System.Net;

using System.Net.Sockets;

using NeuroSky.ThinkGear;

using System.Windows.Forms;

namespace testprogram

{

class Program {

static byte[] buffer = new byte[16];

static Socket tcp1;

static Socket tcp2;

static int flag = 0;

static int flag1 = 0;

static int y = 0;

static int x = 0;

static int tag = 0;

static Connector connector;

static int num=0;

static bool golfZoneDemo = false;

static double task\_famil\_baseline, task\_famil\_cur, task\_famil\_change;

static bool task\_famil\_first;

static double mental\_eff\_baseline, mental\_eff\_cur, mental\_eff\_change;

static bool mental\_eff\_first;

public static void Main(string[] args) {

Assembly assembly = System.Reflection.Assembly.GetExecutingAssembly();

if (assembly != null)

{

object[] customAttribute1 = assembly.GetCustomAttributes(typeof(AssemblyTitleAttribute), false);

if ((customAttribute1 != null) && (customAttribute1.Length > 0))

Console.WriteLine(((AssemblyTitleAttribute)customAttribute1[0]).Title);

object[] customAttribute2 = assembly.GetCustomAttributes(typeof(AssemblyCompanyAttribute), false);

if ((customAttribute2 != null) && (customAttribute2.Length > 0))

Console.WriteLine(((AssemblyCompanyAttribute)customAttribute2[0]).Company);

Console.WriteLine(assembly.GetName().Version.ToString());

}

AppDomain MyDomain = AppDomain.CurrentDomain;

Assembly[] AssembliesLoaded = MyDomain.GetAssemblies();

foreach (Assembly MyAssembly in AssembliesLoaded)

{

if (MyAssembly.FullName.Contains("ThinkGear"))

Console.WriteLine(MyAssembly.FullName);

}

Console.WriteLine("----------");

if (golfZoneDemo) Console.WriteLine("Hello Golfer!");

else Console.WriteLine("Hello EEG!");

Console.WriteLine("----------");

// Initialize a new Connector and add event handlers

Thread t1 = new Thread(new ThreadStart(snake));

t1.Start();

connector = new Connector();

connector.DeviceConnected += new EventHandler(OnDeviceConnected);

connector.DeviceConnectFail += new EventHandler(OnDeviceFail);

connector.DeviceValidating += new EventHandler(OnDeviceValidating);

connector.ConnectScan("COM17");

//start the mental effort and task familiarity calculations

if (golfZoneDemo) {

connector.setMentalEffortEnable(false);

connector.setTaskFamiliarityEnable(false);

connector.setBlinkDetectionEnabled(false);

}

else {

connector.enableTaskDifficulty(); //depricated

connector.enableTaskFamiliarity(); //depricated

connector.setMentalEffortRunContinuous(true);

connector.setMentalEffortEnable(true);

connector.setTaskFamiliarityRunContinuous(true);

connector.setTaskFamiliarityEnable(true);

connector.setBlinkDetectionEnabled(true);

}

task\_famil\_baseline = task\_famil\_cur = task\_famil\_change = 0.0;

task\_famil\_first = true;

mental\_eff\_baseline = mental\_eff\_cur = mental\_eff\_change = 0.0;

mental\_eff\_first = true;

Thread.Sleep(80 \* 60 \* 1000); // time to live for this program (8 min \* 60 sec \* 1000 ms)

Console.WriteLine("----------");

if (golfZoneDemo) Console.WriteLine("Time is up. Goodbye from Golf Zone sample program!");

else Console.WriteLine("Time is up. Goodbye from EEG sample program!");

Console.WriteLine("----------");

// Close all open connections

connector.Close();

Thread.Sleep(10 \* 1000); // delay long enough for a human to see the message (10 sec \* 1000 ms)

Environment.Exit(0);

}

/\*\*

\* Called when a device is connected

\*/

static void OnDeviceConnected(object sender, EventArgs e) {

Console.WriteLine("wwwww");

//Thread t1 = new Thread(new ThreadStart(snake));

// t1.Start();

Thread.Sleep(10000);//睡眠500毫秒，也就是0.5秒

IPAddress ip = IPAddress.Parse("127.0.0.1");

tcp1 = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

tcp2 = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

tcp1.Connect(new IPEndPoint(ip, 9999));

tcp2.Connect(new IPEndPoint(ip, 9999));

Thread t2 = new Thread(new ThreadStart(tcp2\_feed));

t2.Start();

buffer[0]= 90;

buffer[1] = 8;

Connector.DeviceEventArgs de = (Connector.DeviceEventArgs)e;

Console.WriteLine("Device found on: " + de.Device.PortName);

de.Device.DataReceived += new EventHandler(OnDataReceived);

Console.WriteLine("ppppppp");

}

static void tcp2\_feed()

{

while(true)

{

tcp2.Receive(buffer);

Console.WriteLine(buffer[0]+buffer[1]);

}

}

static void snake()

{

Process p = Process.Start("snake.exe");

p.WaitForExit();

}

static void Form1\_KeyDown()

{

            //Ctrl +F4关闭

           //f (Console.ReadKey() == ConsoleKey.F10)

// Console.Write("F10 ");

}

/\*\*

\* Called when scanning fails

\*/

static void OnDeviceFail(object sender, EventArgs e) {

Console.WriteLine("No devices found! :(");

}

/\*\*

\* Called when each port is being validated

\*/

static void OnDeviceValidating(object sender, EventArgs e) {

Console.WriteLine("Validating: ");

}

static byte rcv\_poorSignal\_last = 255; // start with impossible value

static byte rcv\_poorSignal;

static byte rcv\_poorSig\_cnt = 0;

/\*\*

\* Called when data is received from a device

\*/

static void OnDataReceived(object sender, EventArgs e) {

//Device d = (Device)sender;

Device.DataEventArgs de = (Device.DataEventArgs)e;

DataRow[] tempDataRowArray = de.DataRowArray;

string mybyte;

TGParser tgParser = new TGParser();

tgParser.Read(de.DataRowArray);

//Console.WriteLine(buffer[0] + buffer[1]);

//if (buffer=='1')

//{

// Console.WriteLine("wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww");

//}

/\* Loop through new parsed data \*/

for (int i = 0; i < tgParser.ParsedData.Length; i++)

{

if (tgParser.ParsedData[i].ContainsKey("MSG\_MODEL\_IDENTIFIED"))

{

Console.WriteLine("Model Identified");

connector.setMentalEffortRunContinuous(true);

connector.setMentalEffortEnable(true);

connector.setTaskFamiliarityRunContinuous(true);

connector.setTaskFamiliarityEnable(true);

connector.setPositivityEnable(false);

//

// the following are included to demonstrate the overide messages

//

connector.setRespirationRateEnable(true); // not allowed with EEG

connector.setPositivityEnable(true);// not allowed when famil/diff are enabled

}

if (tgParser.ParsedData[i].ContainsKey("PoorSignal"))

{

// NOTE: this doesn't work well with BMD sensors Dual Headband or CardioChip

rcv\_poorSignal = (byte)tgParser.ParsedData[i]["PoorSignal"];

if (rcv\_poorSignal != rcv\_poorSignal\_last || rcv\_poorSig\_cnt >= 30)

{

// when there is a change of state OR every 30 reports

rcv\_poorSig\_cnt = 0; // reset counter

rcv\_poorSignal\_last = rcv\_poorSignal;

if (rcv\_poorSignal == 0)

{

// signal is good, we are connected to a subject

Console.WriteLine("SIGNAL: we have good contact with the subject");

mybyte = "9";

tcp2.Send(Encoding.UTF8.GetBytes(mybyte));

//tcpClient1.Send(Encoding.UTF8.GetBytes(mybyte), mybyte.Length, ipEndPoint);

}

else

{

Console.WriteLine("SIGNAL: is POOR: " + rcv\_poorSignal);

mybyte ="8";

tcp2.Send(Encoding.UTF8.GetBytes(mybyte));

//udpClient.Send(Encoding.UTF8.GetBytes(mybyte), mybyte.Length, ipEndPoint);

}

}

else rcv\_poorSig\_cnt++;

}

if (tgParser.ParsedData[i].ContainsKey("Meditation"))

{

if (tgParser.ParsedData[i]["Meditation"] != 0)

{

Console.WriteLine("Meditation: " + tgParser.ParsedData[i]["Meditation"]);

if (tgParser.ParsedData[i]["Meditation"]>100)

{

tag = 4;

}

if (tgParser.ParsedData[i]["Meditation"] < 20)

{

tag = 6;

}

y = (int)tgParser.ParsedData[i]["Meditation"];

}

}

if (tgParser.ParsedData[i].ContainsKey("Attention"))

{

if (tgParser.ParsedData[i]["Attention"] != 0)

{

Console.WriteLine("Attention : " + tgParser.ParsedData[i]["Attention"]);

flag = 0;

flag1 = 0;

num++;

if (tgParser.ParsedData[i]["Attention"] > 90)

{

tag = 5;

}

x = (int)tgParser.ParsedData[i]["Attention"];

}

}

if (!golfZoneDemo) // turn this off for the Golf Zone Demo

{

if (tgParser.ParsedData[i].ContainsKey("BlinkStrength"))

{

if (tgParser.ParsedData[i]["BlinkStrength"]> 55)

{

Console.WriteLine("\t\tBlinkStrength: " + tgParser.ParsedData[i]["BlinkStrength"]);

if (num>=2)

{

flag++;

num = 0;

}

flag1++;

if (buffer[0]+buffer[1]==97)

{

if (flag1 == 1)

{

Console.WriteLine("KKKKKKKKKK");

mybyte = "1";

tcp2.Send(Encoding.UTF8.GetBytes(mybyte));

//udpClient.Send(Encoding.UTF8.GetBytes(mybyte), mybyte.Length, ipEndPoint2);

}

}

else if(buffer[0] + buffer[1] == 98)

{

if (flag == 1 && flag1 == 1)

{

Console.WriteLine("KKKKKKKKKK");

mybyte = "1";

tcp2.Send(Encoding.UTF8.GetBytes(mybyte));

//udpClient.Send(Encoding.UTF8.GetBytes(mybyte), mybyte.Length, ipEndPoint2);

}

}

}

}

}

if ((x > 0 && y > 0) || tag>0)

{

mybyte = x \* 10000 + y\*10+tag + "";

tcp1.Send(Encoding.UTF8.GetBytes(mybyte));

//udpClient.Send(Encoding.UTF8.GetBytes(mybyte), mybyte.Length, ipEndPoint);

x = 0;

y = 0;

tag = 0;

}

}

}

static double calcPercentChange(double baseline, double current)

{

double change;

if (baseline == 0.0) baseline = 1.0; //don't allow divide by zero

/\*

\* calculate the percentage change

\*/

change = current - baseline;

change = (change / baseline) \* 1000.0 + 0.5;

change = Math.Floor(change) / 10.0;

return (change);

}

}

}

贪吃蛇模块代码

# coding=UTF-8

#!/usr/bin/env python

from \_\_future\_\_ import print\_function

import tensorflow as tf

import cv2

import sys

sys.path.append("game/")

#from tkinter import \*

import socket

import win32api,win32con

import play

import random

import pygame

import numpy as np

import threading

import time

GAME = 'snake' # the name of the game being played for log files

ACTIONS = 4 # number of valid actions

EXPLORE = 2000000. # frames over which to anneal epsilon

FRAME\_PER\_ACTION = 1

INITIAL\_EPSILON = 0.05

FINAL\_EPSILON = 0.05

mutex2 = threading.Lock()

mutex = threading.Lock()

port=8082

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

def weight\_variable(shape):

initial = tf.truncated\_normal(shape, stddev = 0.01)

return tf.Variable(initial)

def bias\_variable(shape):

initial = tf.constant(0.01, shape = shape)

return tf.Variable(initial)

def conv2d(x, W, stride):

return tf.nn.conv2d(x, W, strides = [1, stride, stride, 1], padding = "SAME")

def max\_pool\_2x2(x):

return tf.nn.max\_pool(x, ksize = [1, 2, 2, 1], strides = [1, 2, 2, 1], padding = "SAME")

class createNetwork():

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

self.W\_conv1 = weight\_variable([8, 8, 4, 32])

self.b\_conv1 = bias\_variable([32])

self.W\_conv2 = weight\_variable([4, 4, 32, 64])

self.b\_conv2 = bias\_variable([64])

self.W\_conv3 = weight\_variable([3, 3, 64, 64])

self.b\_conv3 = bias\_variable([64])

self.W\_fc1 = weight\_variable([1600, 512])

self.b\_fc1 = bias\_variable([512])

self.W\_fc2 = weight\_variable([512, ACTIONS])

self.b\_fc2 = bias\_variable([ACTIONS])

self.s = tf.placeholder("float", [None, 80, 80, 4])

self.h\_conv1 = tf.nn.relu(conv2d(self.s, self.W\_conv1, 4) + self.b\_conv1)

self.h\_pool1 = max\_pool\_2x2(self.h\_conv1)

self.h\_conv2 = tf.nn.relu(conv2d(self.h\_pool1, self.W\_conv2, 2) + self.b\_conv2)

self.h\_conv3 = tf.nn.relu(conv2d(self.h\_conv2, self.W\_conv3, 1) + self.b\_conv3)

self.h\_conv3\_flat = tf.reshape(self.h\_conv3, [-1, 1600])

self.h\_fc1 = tf.nn.relu(tf.matmul(self.h\_conv3\_flat, self.W\_fc1) + self.b\_fc1)

# readout layer

self.readout = tf.matmul(self.h\_fc1, self.W\_fc2) + self.b\_fc2

self.predict = tf.argmax(self.readout, 1)

self.a = tf.placeholder("float", [None, ACTIONS])

self.y = tf.placeholder("float", [None])

self.readout\_action = tf.reduce\_sum(tf.multiply(self.readout, self.a), reduction\_indices=1)

self.cost = tf.reduce\_mean(tf.square(self.y - self.readout\_action))

self.train\_step = tf.train.AdamOptimizer(1e-6).minimize(self.cost)

def trainNetwork(current\_q,sess):

global sock1,addr1,sock2,addr2

global data

global data\_2

data='10'

data\_2='11'

t1=threading.Thread(target=tcp,args=(sock1, addr1))

t2=threading.Thread(target=tcp2,args=(sock2, addr2))

t1.start()

t2.start()

pygame.mixer.init()

pygame.mixer.Sound('audio/7301.wav').play()

game\_state = play.game(0,0,0)

win32api.MessageBox(0,"白色代表人工智能(强化学习)\n蓝色代表人类\n操作电脑上下左右键，z表示蓝蛇加速\n按键1,2,3,4分别对应4种模式\n和人工智能比赛吧","规则",win32con.MB\_OK)

do\_nothing = np.zeros(ACTIONS)

do\_nothing[random.randrange(ACTIONS)] = 1

s\_t,terminal,\_= game\_state.frame\_step(do\_nothing,data,data\_2,mutex,mutex2)

s\_t = cv2.cvtColor(cv2.resize(s\_t.astype(np.uint8), (80, 80)), cv2.COLOR\_BGR2GRAY)

\_, s\_t = cv2.threshold(s\_t,1,255,cv2.THRESH\_BINARY)

s\_1 = np.stack((s\_t, s\_t, s\_t, s\_t), axis=2)

sess.run(tf.global\_variables\_initializer())

checkpoint = tf.train.get\_checkpoint\_state("net\_data")

saver = tf.train.Saver()

if checkpoint and checkpoint.model\_checkpoint\_path:

saver.restore(sess, checkpoint.model\_checkpoint\_path)

print("Successfully loaded:", checkpoint.model\_checkpoint\_path)

else:

print("Could not find old network weights")

epsilon = INITIAL\_EPSILON

feed\_back=0

while (True):

# choose an action epsilon greedily

readout\_t = current\_q.readout.eval(feed\_dict={current\_q.s : [s\_1]})[0]

a\_t = np.zeros([ACTIONS])

action\_index = 0

if random.random() <= epsilon:

action\_index = random.randrange(ACTIONS)

a\_t[random.randrange(ACTIONS)] = 1

else:

action\_index = np.argmax(readout\_t)

a\_t[action\_index] = 1

if terminal:

a\_t = np.zeros([ACTIONS])

a\_t[random.randrange(ACTIONS)] = 1

if epsilon > FINAL\_EPSILON:

epsilon -= (INITIAL\_EPSILON - FINAL\_EPSILON) / EXPLORE

# run the selected action and observe next state and reward

s\_t2,terminal,feed\_back= game\_state.frame\_step(a\_t,data,data\_2,mutex,mutex2)

s\_t2 = cv2.cvtColor(cv2.resize(s\_t2.astype(np.uint8), (80, 80)), cv2.COLOR\_BGR2GRAY)

ret, s\_t2 = cv2.threshold(s\_t2, 1, 255, cv2.THRESH\_BINARY)

s\_t2 = np.reshape(s\_t2, (80, 80, 1))

s\_2 = np.append(s\_t2, s\_1[:, :, :3], axis=2)

s\_1=s\_2

if feed\_back==1:

sock2.send(b'10')

elif feed\_back==2:

sock2.send(b'11')

def playGame():

current\_q= createNetwork()

sess = tf.InteractiveSession()

trainNetwork(current\_q,sess)

def music():

pygame.mixer.init()

time.sleep(5)

while(True):

pygame.mixer.Sound('audio/7895.wav').play()

time.sleep(96)

def tcp(sock,addr):

global data

while(True):

mutex.acquire()

data='10'

data,addr=sock.recvfrom(16)

data=data.decode()

def tcp2(sock,addr):

global data\_2

while(True):

mutex2.acquire()

data\_2='11'

data\_2,addr2=sock.recvfrom(16)

data\_2=data\_2.decode()

def main():

global sock1,addr1,sock2,addr2

s.bind(('127.0.0.1',9999))

s.listen(2)

sock1,addr1=s.accept()

sock2,addr2=s.accept()

# t1=threading.Thread(target=tcp,args=(sock1, addr1))

# t2=threading.Thread(target=tcp2,args=(sock2, addr2))

t4 = threading.Thread(target=playGame)

t3=threading.Thread(target=music)

# t1.start()

# t2.start()

t3.start()

t4.start()

#playGame()

if \_\_name\_\_ == "\_\_main\_\_":

main()

# coding=UTF-8

#!/usr/bin/env python

import pygame,sys,time,random

from pygame.locals import \*

import numpy as np

import win32api,win32con

import copy

import socket

import threading

blackColour = pygame.Color(0,0,0)

whiteColour = pygame.Color(255,255,255)

redColour = pygame.Color(255,0,0)

blueColour= pygame.Color(0,0,255)

yellowColour= pygame.Color(0,255,0)

bakColour=pygame.Color(50,50,50)

class game:

def \_\_init\_\_(self,num,flag,tag):

pygame.init()

pygame.mixer.init()

self.fpsClock = pygame.time.Clock()

pygame.display.set\_caption('脑电波对战人工智能')

if flag==0:

self.snakePosition\_m = [140,240]

self.snakeSegments\_m = [[140,240]]

a=random.randint(0,3)

if a==0:

self.direction\_m = 'right'

if a==1:

self.direction\_m = 'left'

if a==2:

self.direction\_m = 'up'

if a==3:

self.direction\_m = 'down'

self.changeDirection\_m = self.direction\_m

self.snakePosition = [140,240]

self.snakeSegments = [[140,240]]

a=random.randint(0,3)

if a==0:

self.direction = 'right'

if a==1:

self.direction = 'left'

if a==2:

self.direction = 'up'

if a==3:

self.direction = 'down'

self.changeDirection = self.direction

x = random.randrange(0,15)

y = random.randrange(0,25)

self.raspberryPosition = [int(x\*20),int(y\*20)]

self.raspberrySpawned = 1

self.fps=1

self.temp=0

self.con=8

if num==1 and flag==1:

self.snakePosition\_m = [140,240]

self.snakeSegments\_m = [[140,240]]

a=random.randint(0,3)

if a==0:

self.direction\_m = 'right'

if a==1:

self.direction\_m = 'left'

if a==2:

self.direction\_m = 'up'

if a==3:

self.direction\_m = 'down'

self.changeDirection\_m = self.direction\_m

elif num==0 and flag==1:

self.snakePosition = [140,240]

self.snakeSegments = [[140,240]]

a=random.randint(0,3)

if a==0:

self.direction = 'right'

if a==1:

self.direction = 'left'

if a==2:

self.direction = 'up'

if a==3:

self.direction = 'down'

self.changeDirection = self.direction

if tag==0:

x = random.randrange(0,15)

y = random.randrange(0,25)

self.raspberryPosition = [int(x\*20),int(y\*20)]

self.raspberrySpawned = 1

self.playSurface = pygame.display.set\_mode((500,500))

# background = pygame.image.load('image/8.jpg').convert\_alpha ()

# width,height = background.get\_size()

# background = pygame.transform.smoothscale(background,(width//3,height//2))

# self.playSurface.blit(background, (300,-20))

def frame\_step(self,input\_actions,data,data\_2,mutex,mutex2):

#self.playSurface.set\_clip(0,0, 300, 500)

q=0

acce=0

terminal=False

con=1

image\_data=np.zeros([300,500,3])

feed\_back=0

#pygame.mixer.Sound('audio/swoosh.wav').play()

#self.playSurface.set\_clip(0,0, 300, 500)

playSurface=self.playSurface.subsurface(pygame.rect.Rect((0,0), (300, 500)))

lenth=len(data)

if lenth==7:

data1=data[6]

data2=data[3]+data[4]+data[5]

data3=data[0]+data[1]+data[2]

elif lenth==6:

data1=data[5]

data2=data[2]+data[3]+data[4]

data3=data[0]+data[1]

elif lenth==1:

data1=data[0]

data2='0'

data3='0'

else:

data1='0'

data2='0'

data3='0'

for event in pygame.event.get():

if event.type == QUIT:

pygame.quit()

sys.exit()

elif event.type == KEYDOWN:

# 判断键盘事件

if event.key == ord('t'):

win32api.MessageBox(0,"切换到人工智能模式","提示",win32con.MB\_OK)

self.temp=1

if event.key == ord('f'):

win32api.MessageBox(0,"切换到对抗模式","提示",win32con.MB\_OK)

self.temp=2

if event.key == ord('c'):

win32api.MessageBox(0,"切换到人类玩家模式","提示",win32con.MB\_OK)

self.temp=3

if event.key == ord('0'):

feed\_back=2

self.fps=1

win32api.MessageBox(0,"极简模式","提示",win32con.MB\_OK)

if event.key == ord('1'):

feed\_back=2

self.fps=2

win32api.MessageBox(0,"简单模式","提示",win32con.MB\_OK)

if event.key == ord('2'):

feed\_back=1

self.fps=4

win32api.MessageBox(0,"中级模式","提示",win32con.MB\_OK)

if event.key == ord('3'):

self.fps=10

win32api.MessageBox(0,"高级模式","提示",win32con.MB\_OK)

if event.key == ord('4'):

self.fps=30

win32api.MessageBox(0,"终极模式","提示",win32con.MB\_OK)

if event.key == ord('5'):

self.fps=80

win32api.MessageBox(0,"闪电蛇","提示",win32con.MB\_OK)

if event.key == ord('z'):

acce=1

if event.key == K\_RIGHT:

self.changeDirection\_m = 'right'

if event.key == K\_LEFT:

self.changeDirection\_m = 'left'

if event.key == K\_UP:

self.changeDirection\_m = 'up'

if event.key == K\_DOWN:

self.changeDirection\_m = 'down'

if event.key == K\_ESCAPE:

pygame.event.post(pygame.event.Event(QUIT))

if self.direction\_m=='right':

# if data == '0':

# self.changeDirection\_m = 'up'

if data\_2 == '1':

self.changeDirection\_m = 'down'

if self.direction\_m=='left':

# if data == '0':

# self.changeDirection\_m = 'down'

if data\_2 == '1':

self.changeDirection\_m = 'up'

if self.direction\_m=='up':

# if data == '0':

# self.changeDirection\_m = 'left'

if data\_2 == '1':

self.changeDirection\_m = 'right'

if self.direction\_m=='down':

# if data == '0':

# self.changeDirection\_m = 'right'

if data\_2 == '1':

self.changeDirection\_m = 'left'

# print(data\_2)

# print (data)

# print("kkkk")

# if data == '0':

# self.changeDirection\_m = 'right'

# if data == '1':

# self.changeDirection\_m = 'left'

# if data == '2':

# self.changeDirection\_m = 'up'

# if data == '3':

# self.changeDirection\_m = 'down'

if data1=='4':

self.temp=1

#win32api.MessageBox(0,"切换到人工智能模式","提示",win32con.MB\_OK)

if data1=='5':

self.temp=2

#win32api.MessageBox(0,"切换到对抗模式","提示",win32con.MB\_OK)

if data1=='6':

self.temp=3

#win32api.MessageBox(0,"还原到最初模式","提示",win32con.MB\_OK)

if self.temp==0:

self.temp=3

# 判断是否输入了反方向

if len(self.snakeSegments\_m)!=1:

if self.changeDirection\_m == 'right' and not self.direction\_m == 'left':

self.direction\_m = self.changeDirection\_m

if self.changeDirection\_m == 'left' and not self.direction\_m == 'right':

self.direction\_m = self.changeDirection\_m

if self.changeDirection\_m == 'up' and not self.direction\_m == 'down':

self.direction\_m = self.changeDirection\_m

if self.changeDirection\_m == 'down' and not self.direction\_m == 'up':

self.direction\_m = self.changeDirection\_m

else:

if self.changeDirection\_m == 'right':

self.direction\_m = self.changeDirection\_m

if self.changeDirection\_m == 'left':

self.direction\_m = self.changeDirection\_m

if self.changeDirection\_m == 'up':

self.direction\_m = self.changeDirection\_m

if self.changeDirection\_m == 'down':

self.direction\_m = self.changeDirection\_m

# 根据方向移动蛇头的坐标

if self.temp==3 or self.temp==2:

if self.direction\_m == 'right':

if acce==1:

self.snakePosition\_m[0] += 40

for i in range(len(self.snakeSegments\_m)):

self.snakeSegments\_m[i][0]+=20

else:

self.snakePosition\_m[0] += 20

if self.direction\_m == 'left':

if acce==1:

self.snakePosition\_m[0] -= 40

for i in range(len(self.snakeSegments\_m)):

self.snakeSegments\_m[i][0]-=20

else:

self.snakePosition\_m[0] -= 20

#self.snakePosition\_m[0] -= 20

if self.direction\_m == 'up':

if acce==1:

self.snakePosition\_m[1] -= 40

for i in range(len(self.snakeSegments\_m)):

self.snakeSegments\_m[i][1]-=20

else:

self.snakePosition\_m[1] -= 20

#self.snakePosition\_m[1] -= 20

if self.direction\_m == 'down':

if acce==1:

self.snakePosition\_m[1] += 40

for i in range(len(self.snakeSegments\_m)):

self.snakeSegments\_m[i][1]+=20

else:

self.snakePosition\_m[1] += 20

#self.snakePosition\_m[1] += 20

if self.temp==1 or self.temp==2:

if sum(input\_actions) != 1:

raise ValueError('Multiple input actions!')

if input\_actions[0]==1:

self.changeDirection = 'right'

if input\_actions[1]==1:

self.changeDirection = 'left'

if input\_actions[2]==1:

self.changeDirection = 'up'

if input\_actions[3]==1:

self.changeDirection = 'down'

if input\_actions[0]==1 and not self.direction == 'left':

self.direction = self.changeDirection

if input\_actions[1]==1 and not self.direction == 'right':

self.direction = self.changeDirection

if input\_actions[2]==1 and not self.direction == 'down':

self.direction = self.changeDirection

if input\_actions[3]==1 and not self.direction == 'up':

self.direction = self.changeDirection

# 根据方向移动蛇头的坐标

if self.direction == 'right':

self.snakePosition[0] += 20

if self.direction == 'left':

self.snakePosition[0] -= 20

if self.direction == 'up':

self.snakePosition[1] -= 20

if self.direction == 'down':

self.snakePosition[1] += 20

# 增加蛇的长度

self.snakeSegments.insert(0,list(self.snakePosition))

# 判断是否吃掉了树莓

if self.snakePosition[0] == self.raspberryPosition[0] and self.snakePosition[1] == self.raspberryPosition[1]:

self.raspberrySpawned = 0

pygame.mixer.Sound('audio/point.wav').play()

else:

self.snakeSegments.pop()

# 如果吃掉树莓，则重新生成树莓

if self.raspberrySpawned == 0:

while(True):

x = random.randrange(0,15)

y = random.randrange(0,25)

self.raspberryPosition = [int(x\*20),int(y\*20)]

for position in self.snakeSegments:

if position==self.raspberryPosition:

q=1

if q==1:

q=0

continue

else:

break

self.raspberrySpawned = 1

q=0

# 增加蛇的长度

if self.temp==3 or self.temp==2:

self.snakeSegments\_m.insert(0,list(self.snakePosition\_m))

# 判断是否吃掉了树莓

if self.snakePosition\_m[0] == self.raspberryPosition[0] and self.snakePosition\_m[1] == self.raspberryPosition[1]:

self.raspberrySpawned = 0

pygame.mixer.Sound('audio/point.wav').play()

else:

self.snakeSegments\_m.pop()

# 如果吃掉树莓，则重新生成树莓

if self.raspberrySpawned == 0:

while(True):

x = random.randrange(0,15)

y = random.randrange(0,25)

self.raspberryPosition = [int(x\*20),int(y\*20)]

for position in self.snakeSegments\_m:

if position==self.raspberryPosition:

q=1

if q==1:

q=0

continue

else:

break

self.raspberrySpawned = 1

q=0

if self.temp==1 or self.temp==2:

#self.playSurface.fill(blackColour)

#playSurface=self.playSurface.subsurface(pygame.rect.Rect((0,0), (300, 500)))

#self.playSurface.subsurface(pygame.rect.Rect((0,0), (300, 500))).fill(blackColour)

playSurface.fill(blackColour)

for position in self.snakeSegments:

pygame.draw.rect(playSurface,whiteColour,Rect(position[0],position[1],20,20))

pygame.draw.rect(playSurface,redColour,Rect(self.raspberryPosition[0], self.raspberryPosition[1],20,20))

pygame.display.flip()

#image\_data = pygame.surfarray.array3d(pygame.display.get\_surface())

image\_data = pygame.surfarray.array3d(self.playSurface.subsurface(pygame.rect.Rect((0,0), (300, 500))))

#print (image\_data.shape)

if self.temp==1 or self.temp==2:

if self.snakePosition[0] > 280 or self.snakePosition[0] < 0:

terminal=True

pygame.mixer.Sound('audio/hit.wav').play()

if self.temp==1:

self.\_\_init\_\_(0,1,0)

else:

self.\_\_init\_\_(0,1,1)

if self.snakePosition[1] > 480 or self.snakePosition[1] < 0:

terminal=True

pygame.mixer.Sound('audio/hit.wav').play()

if self.temp==1:

self.\_\_init\_\_(0,1,0)

else:

self.\_\_init\_\_(0,1,1)

# 绘制pygame显示层

if self.temp==3 or self.temp==2:

#self.playSurface.fill(blackColour)

#self.playSurface.subsurface(pygame.rect.Rect((0,0), (300, 500))).fill(blackColour)

playSurface.fill(blackColour)

if self.temp==2:

for position in self.snakeSegments:

pygame.draw.rect(playSurface,whiteColour,Rect(position[0],position[1],20,20))

pygame.draw.rect(playSurface,redColour,Rect(self.raspberryPosition[0], self.raspberryPosition[1],20,20))

for position\_m in self.snakeSegments\_m:

pygame.draw.rect(playSurface,blueColour,Rect(position\_m[0],position\_m[1],20,20))

pygame.draw.rect(playSurface,redColour,Rect(self.raspberryPosition[0], self.raspberryPosition[1],20,20))

# 刷新pygame显示层

pygame.display.flip()

# 判断是否死亡

if self.snakePosition\_m[0] > 280 or self.snakePosition\_m[0] < 0:

if self.temp==3:

self.\_\_init\_\_(1,1,0)

else:

if terminal:

self.\_\_init\_\_(1,1,0)

pygame.mixer.Sound('audio/die.wav').play()

else:

self.\_\_init\_\_(1,1,1)

pygame.mixer.Sound('audio/die.wav').play()

if self.snakePosition\_m[1] > 480 or self.snakePosition\_m[1] < 0:

if self.temp==3:

self.\_\_init\_\_(1,1,0)

else:

if terminal:

self.\_\_init\_\_(1,1,0)

pygame.mixer.Sound('audio/die.wav').play()

else:

self.\_\_init\_\_(1,1,1)

pygame.mixer.Sound('audio/die.wav').play()

#pygame.display.update()

self.playSurface.set\_clip(300,0, 200, 500)

self.playSurface.fill(bakColour)

background0 = pygame.image.load('image/she2.jpg').convert\_alpha ()

background1 = pygame.image.load('image/ming3.jpg').convert\_alpha ()

background2 = pygame.image.load('image/zhuan3.jpg').convert\_alpha ()

background3 = pygame.image.load('image/zha.jpg').convert\_alpha ()

background4 = pygame.image.load('image/singal.jpg').convert\_alpha ()

background5 = pygame.image.load('image/zheng.jpg').convert\_alpha ()

background6 = pygame.image.load('image/ruo.jpg').convert\_alpha ()

background7 = pygame.image.load('image/lian.jpg').convert\_alpha ()

background8 = pygame.image.load('image/zhi.jpg').convert\_alpha ()

background9 = pygame.image.load('image/duan.jpg').convert\_alpha ()

background10 = pygame.image.load('image/mo.jpg').convert\_alpha ()

background11\_1 = pygame.image.load('image/nao.jpg').convert\_alpha ()

background11\_2 = pygame.image.load('image/nao\_h.jpg').convert\_alpha ()

background12\_1 = pygame.image.load('image/ren.jpg').convert\_alpha ()

background12\_2 = pygame.image.load('image/ren\_h.jpg').convert\_alpha ()

background13\_1 = pygame.image.load('image/dui2.jpg').convert\_alpha ()

background13\_2 = pygame.image.load('image/dui2\_h.jpg').convert\_alpha ()

# width,height = background.get\_size()

# background = pygame.transform.smoothscale(background,(width,height))

self.playSurface.blit(background0, (300,20))

self.playSurface.blit(background1, (420,380))

self.playSurface.blit(background2, (320,380))

self.playSurface.blit(background3, (320,200))

self.playSurface.blit(background4, (310,150))

self.playSurface.blit(background7, (310,100))

self.playSurface.blit(background10, (310,420))

if self.temp==1:

self.playSurface.blit(background12\_2, (370,450))

self.playSurface.blit(background11\_1, (360,420))

self.playSurface.blit(background13\_1, (440,420))

if self.temp==2:

self.playSurface.blit(background12\_1, (370,450))

self.playSurface.blit(background11\_1, (360,420))

self.playSurface.blit(background13\_2, (440,420))

if self.temp==3:

self.playSurface.blit(background12\_1, (370,450))

self.playSurface.blit(background11\_2, (360,420))

self.playSurface.blit(background13\_1, (440,420))

if data\_2=='8':

self.con=8

if data\_2=='9':

self.con=9

if self.con==8:

self.playSurface.blit(background6, (400,150))

self.playSurface.blit(background9, (400,100))

if self.con==9:

self.playSurface.blit(background5, (400,150))

self.playSurface.blit(background8, (400,100))

data1\_1=int(data3)

pygame.draw.rect(self.playSurface,redColour,Rect(340,350,20,-data1\_1))#专注度

data2\_1=int(data2)

pygame.draw.rect(self.playSurface,yellowColour,Rect(440,350,20,-data2\_1))#冥想度

if (data\_2=='1'):

pygame.draw.rect(self.playSurface,blueColour,Rect(440,200,20,20))

else:

pygame.draw.rect(self.playSurface,bakColour,Rect(440,200,20,20))

pygame.display.flip()

if data!='10':

mutex.release()

if data\_2!='11':

mutex2.release()

self.fpsClock.tick(self.fps)

return image\_data,terminal,feed\_back